EDÍCIA SKRÍPT

ZBIERKA RIEŠENÝCH ÚLOH Z PREDMETU AUTOMATY A FORMÁLNE JAZYKY

Viliam Hromada

STU FEI
SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

SPEKTRUM STU

ZBIERKA RIEŠENÝCH ÚLOH Z PREDMETU AUTOMATY A FORMÁLNE JAZYKY

Viliam Hromada

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
2023

Recenzenti: doc. Ing. Milan Vojvoda, PhD.
Mgr. Tomáš Fabšič, PhD.

Schválilo Vedenie Fakulty elektrotechniky a informatiky STU v Bratislave.

ISBN 978-80-227-5320-3

Obsah

Úvod

Vážené čitateľky, vážení čitatelia!

Skriptá, ktoré sa Vám dostali do rúk, slúžia ako doplnkový materiál k predmetu *Automaty a formálne jazyky*, vyučovaný v prvom ročníku inžinierskeho štúdia študijného programu *Aplikovaná informatika* na Fakulte elektrotechniky a informatiky STU v Bratislave. Obsahujú riešené úlohy doplnené o vysvetľujúci výklad. Úlohy pokrývajú prednášanú problematiku v rámci predmetu *Automaty a formálne jazyky*.

Úlohou skrípt je poskytnúť Vám zásobu úloh, ktorých štúdium a riešenie by Vám malo pomôcť lepšie porozumieť a oboznámiť sa s prednášanými konceptami a algoritmami. Skriptá zároveň neslúžia ako samostatná učebnica formálnych jazykov, automatov, gramatík či lexikálnej a syntaktickej analýzy, preto odporúčame venovať sa ich teoretickému štúdiu v rámci prednášok.

Rozdelenie úloh sa snaží odzrkadliť logickú následnosť konceptov na prednáškach, t. j. počnúc elementárnymi pojmami ako abeceda, jazyk či formálna gramatika. Ďalej sa skriptá venujú regulárnym jazykom a ich výpočtovému modelu — konečným automatom, či popisnému formalizmu — regulárnym výrazom. Za nimi nasledujú bezkontextové gramatiky a ich výpočtový model — zásobníkové automaty. Záver skrípt pozostáva z konceptov, ktoré sú praktickým využitím automatov a gramatík pri preklade počítačových programov — lexikálnej analýzy a rôznych druhov syntaktických analyzátorov.

Ak tieto skriptá pomôžu čo i len jedinej študentke či študentovi lepšie porozumieť príslušnej problematike, je autor skrípt rád, že ich nepísal nadarmo.

Kapitola 1

Formálne jazyky a gramatiky

1.1 Abeceda, ret'azce

Úloha č. 1.1.1 Je daná abeceda $A = \{a, b, c\}$. Uvažujme tri ret'azce nad touto abecedou, označíme si ich ret'azcovými premennými x, y, z. Nech ich konkrétne hodnoty sú $x = abac, y = aa, z = \varepsilon^{\dagger}$.

1. Určte výsledky zret'azení $xx, xy, xz, yx, yy, yz, zx, zy, zz$ a ich dĺžky.
2. Určte x^i, y^i, z^i pre $i \in \{0, 1, 2, 3\}$ a ich dĺžky.
3. Určte obrátené ret'azce x^R, y^R, z^R a ich dĺžky.
4. Uveďte, čo je A^* a čo je A^+.

Riešenie:

1. Zret'azenie dvoch ret'azcov $x = a_1a_2...a_m$ a $y = b_1b_2...b_n$ nad abecedou A je definované ako

$$xy = a_1a_2...a_mb_1b_2...b_n,$$

   t. j. zapíšeme za seba najprv ret'azec v premennej x a potom ret'azec v premennej y. Ak dĺžka $|x| = |a_1a_2...a_m| = m$ a $|y| = |b_1b_2...b_n| = n$, tak potom dĺžka zret'azenia $|xy| = |a_1a_2...a_mb_1b_2...b_n| = |a_1a_2...a_m| + |b_1b_2...b_n| = m + n$.

   - Pre ret'azec $x = abac$ dĺžky $|x| = 4$ máme $xx = abacabac$, $|xx| = |abacabac| = |x| + |x| = 4 + 4 = 8$.
   - Pre ret'azce $x = abac, |x| = 4$ a $y = aa, |y| = 2$ je ich zret'azením v poradí xy ret'azec $xy = abacaa$ dĺžky $|xy| = |x| + |y| = 4 + 2 = 6$.

---

†V týchto skriptách budeme pomocou ε reprezentovat' tzv. prázdny ret'azec. V inej literatúre sa môžete stretnút' s označením λ.

- Pre reťazce $x = abac, |x| = 4$ a $y = aa, |y| = 2$ je ich zreťazením v poradí yx reťazec $yx = aaabac$ dĺžky $|yx| = |y| + |x| = 2 + 4 = 6$. Všimnite si, že zreťazenie **nie je komutatívne**, teda záleží na poradí, v akom reťazce zreťazíme. Konkrétne tu vidíme že $xy \neq yx$, keďže $abacaa \neq aaabac$.
- Pre reťazce $x = abac, |x| = 4$, a $z = \varepsilon, |z| = 0$, je ich zreťazením v poradí xz reťazec $xz = abac\varepsilon = abac$ dĺžky $|xz| = |x|+|z| = 4+0 = 4$. Pre ľubovoľný reťazec x platí, že ak ho chceme zreťaziť s prázdnym reťazcom, dostávame znovu len reťazec x, t. j. $x\varepsilon = \varepsilon x = x$.
- Podobne dostávame aj zreťazenie zz, ktoré predstavuje zreťazenie 2 prázdnych reťazcov, $zz = \varepsilon\varepsilon = \varepsilon$. Dĺžka $|zz| = |z| + |z| = 0 + 0 = 0$.
- Ďalej platí $yy = aaaa$, $|yy| = 4$, $yz = aa\varepsilon = aa$, $|yz| = 2$, $zx = \varepsilon abac = abac$, $|zx| = 4$, $zy = \varepsilon aa = aa$, $|zy| = 2$.

2. Mocnina reťazca x^i je definovaná ako i-násobné zreťazenie reťazca x samého so sebou,
$$x^i = \underbrace{xx...x}_{i}$$
Dĺžka x^i je teda $|x^i| = i|x|$.
    - Pre $x^1 = x$, t. j. dostávame priamo pôvodný reťazec, $x^1 = x = abac$. Dĺžka $|x^1| = |x| = 4$.
    - Pre $x^2 = xx = abacabac$ dĺžky $|x^2| = 2|x| = 8$.
    - Pre $x^3 = xxx = abacabacabac$ dĺžky $|x^3| = 3|x| = 12$.
    - Reťazec x^0 je reťazec, ktorý vznikne nula zreťazeniami reťazca x. Logicky, ak teda nezreťazíme žiadnu kópiu reťazca x, dostali sme „nič“ a výsledkom je teda prázdny reťazec, $x^0 = \varepsilon$, dĺžky $|x^0| = 0|x| = 0$.
    - Analogicky $y^0 = \varepsilon, |y^0| = 0$; $y^1 = aa, |y^1| = 2$; $y^2 = aaaa, |y^2| = 2|y| = 4$; $y^3 = aaaaaa, |y^3| = 3|y| = 6$.
    - Analogicky $z^0 = \varepsilon, |z^0| = 0; z^1 = \varepsilon, |z^1| = 0$; $z^2 = \varepsilon\varepsilon = \varepsilon, |z^2| = 2|z| = 0$; $|z^3| = \varepsilon\varepsilon\varepsilon = \varepsilon, |z^3| = 0$.

3. K reťazcu $x = a_1a_2...a_{m-1}a_m$ nad abecedou A je obrátený reťazec x^R definovaný ako:
$$x^R = a_ma_{m-1}...a_2a_1,$$
teda ako reťazec, ktorý vznikne napísaním reťazca a „odzadu“. Logicky teda $|x^R| = |x|$.
    - Pre $x = abac$ obrátený reťazec $x^R = caba$, dĺžky $|x^R| = |x| = 4$.
    - Pre $y = aa$ obrátený reťazec $y^R = aa$. V tomto prípade teda $y^R = y$, keďže reťazec aa predstavuje tzv. palindróm (reťazec, ktorý sa číta rovnako spredu i odzadu). Dĺžka $|y^R| = |y| = 2$.

- Pre $z = \varepsilon$ obrátený ret'azec $z^R = \varepsilon$, $|z^R| = |z| = 0$.

4. Výrazom A^* označujeme všetky ret'azce konečnej dĺžky, ktoré sú zostavené zo symbolov abecedy A. V našom prípade budú teda A^* všetky ret'azce zložené zo symbolov $\{a, b, c\}$, kam patria:

   - ret'azec dĺžky 0: ε,
   - ret'azce dĺžky 1: a, b, c,
   - ret'azce dĺžky 2: $aa, ab, ac, ba, bb, bc, ca, cb, cc$,
   - ret'azce dĺžky 3: $aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc$,
   - ret'azce dĺžky 4, 5, 6, ...

   Celkovo je teda $A^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...\}$.

   Množina A^+ je množina všetkých **neprázdnych** ret'azcov konečnej dĺžky nad abecedou A, teda ret'azcov dĺžky aspoň 1, kam patria:

   - ret'azce dĺžky 1: a, b, c,
   - ret'azce dĺžky 2: $aa, ab, ac, ba, bb, bc, ca, cb, cc$,
   - ret'azce dĺžky 3: $aaa, aab, aac, aba, abb, abc, aca, acb, acc, baa, bab, bac, bba, bbb, bbc, bca, bcb, bcc, caa, cab, cac, cba, cbb, cbc, cca, ccb, ccc$,
   - ret'azce dĺžky 4, 5, 6, ...

   Celkovo je teda $A^+ = \{a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...\}$. Množina A^+ teda v porovnaní s A^* **neobsahuje** prázdny ret'azec.

Úloha č. 1.1.2 Je daná abeceda $A = \{\texttt{if}, \texttt{then}, \texttt{else}, \texttt{while}, \texttt{id}, \texttt{const}, =, +, -, <, >, ==\}$ a dva ret'azce nad touto abecedou, $x = \texttt{id} = \texttt{id} + \texttt{const} + \texttt{const}$ a $y = \texttt{if id} > \texttt{const then id} = \texttt{id} + \texttt{id else id} = \texttt{id}$ †.

1. Určte dĺžky ret'azcov x a y.
2. Určte obrátené ret'azce x^R, y^R.

Riešenie:

1. Dĺžka ret'azca je definovaná ako počet symbolov abecedy, ktoré sa v ret'azci vyskytujú. Keď ret'azec x rozdelíme na jednotlivé symboly abecedy, ktoré pre lepšiu ilustráciu očíslujeme dolnými indexami, dostávame:

$$\texttt{id}_1 \;=_2\; \texttt{id}_3 \;+_4\; \texttt{const}_5 \;+_6\; \texttt{const}_7$$

† Pre prehl'adnost' sme v uvedených ret'azcoch oddelili jednotlivé symboly medzerami. Ak by sme chceli byt' formálne presní, samotné ret'azce medzery neobsahujú, teda $x = \texttt{id} = \texttt{id} + \texttt{const} + \texttt{const}$ a $y = \texttt{ifid} > \texttt{constthenid} = \texttt{id} + \texttt{idelseid} = \texttt{id}$

Keďže id, $+$, const, $=$ sú v tomto prípade jednotlivé symboly abecedy, dĺžka reťazca x je $|x| = 7$, pretože je reťazec x tvorený postupnosťou siedmich symbolov abecedy.

Podobne vidíme, že dĺžka reťazca y je $|y| = 14$, pretože je výsledkom zreťazenia 14 symbolov z abecedy:

$$\texttt{if}_1\ \texttt{id}_2\ >_3\ \texttt{const}_4\ \texttt{then}_5\ \texttt{id}_6\ =_7\ \texttt{id}_8\ +_9\ \texttt{id}_{10}\ \texttt{else}_{11}\ \texttt{id}_{12}\ =_{13}\ \texttt{id}_{14}$$

2. Obrátený reťazec x^R k reťazcu $x = \texttt{id} = \texttt{id} + \texttt{const} + \texttt{const}$ je:

$$x^R = \texttt{const} + \texttt{const} + \texttt{id} = \texttt{id}$$

teda reťazec x zapíšeme v opačnom poradí použitých symbolov abecedy. **Pozor!** Samotné symboly nepíšeme odzadu, teda

$$x^R = \texttt{tsnoc} + \texttt{tsnoc} + \texttt{di} = \texttt{di}$$

nie je správne riešenie, keďže aj obrátený reťazec musí používať symboly z abecedy A.

Obrátený reťazec y^R má hodnotu:

$$y^R = \texttt{id} = \texttt{id}\ \texttt{else}\ \texttt{id} + \texttt{id} = \texttt{id}\ \texttt{then}\ \texttt{const} > \texttt{id}\ \texttt{if}$$

1.2 Formálne jazyky

Úloha č. 1.2.1 Je daná abeceda $A = \{a, b, c\}$ a nasledovné jazyky nad touto abecedou:

- $L_1 = \{c\}$,
- $L_2 = \{aa, ab, ba, bb\}$,
- $L_3 = \{aw \mid w \in \{a, b, c\}^*\}$,
- $L_4 = \{a^n cb^n \mid n \in \mathbb{N}_0\}^\dagger$.

1. Určte, ktoré jazyky z uvedených sú konečné a ktoré sú nekonečné. Popíšte slovne jazyky L_3, L_4.
2. Popíšte jazyky, ktoré vzniknú ako zreťazenie $L_1L_1, L_1L_2, L_2L_1, L_2L_2$.
3. Popíšte jazyky $L_1^2, L_1^3, L_1^*, L_1^+, L_2^*, L_3^C, L_4^2$.
4. Popíšte jazyky $L_2 \cap L_3, L_2 \cap L_4, L_3 \cap L_4, L_2 \setminus L_3, L_3 \setminus L_2, L_4 \setminus L_3, L_3 \setminus L_4, L_1 \cup L_2, L_1 \cup L_4, L_3 \cup L_3^C$.

$^\dagger\mathbb{N}_0$ označuje všetky nezáporné celé čísla, t. j. $\mathbb{N}_0 = \{0, 1, 2, 3, ...\}$

Riešenie:

1. Jazyky L_1 a L_2 sú konečné, pretože obsahujú konečný počet prvkov (reťazcov). L_1 obsahuje 1 reťazec, L_2 obsahuje 4 reťazce.

   Jazyk L_3 tvoria všetky reťazce zo symbolov $\{a, b, c\}$, ktoré začínajú symbolom a, t. j. $L_3 = \{a, aa, ab, ac, aaa, aab, aac, ...\}$. Keďže týchto reťazcov je nekonečne veľa, L_3 je nekonečný jazyk.

   Jazyk L_4 tvoria všetky také reťazce nad abecedou $\{a, b, c\}$, ktoré majú na začiatku n-krát symbol a, za ním jedenkrát symbol c a za ním n-krát symbol b. Napríklad pre $n = 0$ je taký reťazec c, pre $n = 1$ reťazec acb, pre $n = 2$ reťazec $a^2cb^2 = aacbb$, pre $n = 3$ reťazec $a^3cb^3 = aaacbbb$ atď. Teda $L_4 = \{c, acb, aacbb, aaacbbb, aaaacbbbb, ...\}$. Keďže hodnota n môže byť ľubovoľne veľká, takýchto reťazcov je nekonečne veľa a L_4 je nekonečný jazyk.

2. Zreťazenie jazykov L_iL_j je definované ako:

   $$L_iL_j = \{uv \mid u \in L_i, v \in L_j\},$$

   teda ako zreťazenie každého reťazca z jazyka L_i s každým reťazcom z jazyka L_j. Podobne ako pri klasickom zreťazení 2 reťazcov je potrebné dbať na poradie, t. j. vo všeobecnosti $L_iL_j \neq L_jL_i$.

   - $L_1L_1 = \{cc\}$.
   - $L_1L_2 = \{caa, cab, cba, cbb\}$.
   - $L_2L_1 = \{aac, abc, bac, bbc\}$.
   - $L_2L_2 = \{aaaa, aaab, aaba, aabb, abaa, abab, abba, abbb, baaa, baab, baba, babb, bbaa, bbab, bbba, bbbb\}$.

3. Mocnina jazyka L^i je jeho i-násobné zreťazenie samého so sebou, čo môžeme formálne zadefinovať ako:

   $$L^0 = \{\varepsilon\}$$
   $$L^i = LL^{i-1}$$

   - $L_1^2 = L_1L_1 = \{cc\}$.
   - $L_1^3 = L_1L_1L_1 = \{ccc\}$.
   - $L_4^2 = L_4L_4 = \{cc, cacb, acbc, caacbb, aacbbc, acbacb, aacbbacb, ...\}$, teda všetky reťazce, ktoré vzniknú zreťazením dvoch reťazcov v tvare a^mcb^m, a^ncb^n. To môžeme zapísať aj ako $L_4^2 = \{a^mcb^ma^ncb^n \mid m \in \mathbb{N}_0, n \in \mathbb{N}_0\}$.

   Iterácia jazyka L^* je definovaná ako

   $$L^* = \bigcup_{i=0}^{\infty} L^i,$$

   teda ako jazyk obsahujúci všetky reťazce, ktoré je možné zostrojiť ľubovoľným počtom zreťazení reťazcov z jazyka L (vrátane prázdneho reťazca).

- $L_1^* = L_1^0 \cup L_1^1 \cup L_1^2 \cup L_1^3 \cup ...$ kde
  - $L_1^0 = \{\varepsilon\}$,
  - $L_1^1 = L_1 = \{c\}$,
  - $L_1^2 = \{cc\}$,
  - $L_1^3 = \{ccc\}$,
  - ...

  teda $L_1^* = \{\varepsilon, c, cc, ccc, ...\}$, teda všetky reťazce, ktoré je možné zostrojiť z reťazca c, ktorý bol v pôvodnom jazyku L_1.

Podobne

- $L_2^* = L_2^0 \cup L_2^1 \cup L_2^2 \cup L_2^3 \cup ...$ kde
  - $L_2^0 = \{\varepsilon\}$,
  - $L_2^1 = L_2 = \{aa, ab, ba, bb\}$,
  - $L_2^2 = \{aaaa, aaab, aaba, aabb, abaa, abab, ..., bbbb\}$,
  - $L_2^3 = \{aaaaaa, aaaaab, aaaaba, aaaabb, aaabaa, ..., bbbbbb\}$,
  - ...

  teda $L_2^* = \{\varepsilon, aa, ab, ba, bb, aaaa, aaab, ..., bbbb, aaaaaa, ...\}$, teda všetky reťazce, ktoré je možné zostrojiť z reťazcov $\{aa, ab, ba, bb\}$, ktoré boli v pôvodnom jazyku L_2.

Pozitívna iterácia jazyka L^+ je definovaná ako:

$$L^+ = \bigcup_{i=1}^{\infty} L^i,$$

teda ako jazyk obsahujúci všetky reťazce, ktoré je možné zostrojiť ľubovoľným nenulovým počtom zreťazení reťazcov z jazyka L.

- $L_1^+ = L_1^1 \cup L_1^2 \cup L_1^3 \cup ...$ kde
  - $L_1^1 = L_1 = \{c\}$,
  - $L_1^2 = \{cc\}$,
  - $L_1^3 = \{ccc\}$,
  - ...

  teda $L_1^+ = \{c, cc, ccc, ...\}$ sú všetky reťazce, ktoré je možné zostrojiť z reťazca c, ktorý bol v pôvodnom jazyku L_1, pričom tento reťazec sa použije aspoň 1-krát.

Komplement jazyka L^C je definovaný pomocou rozdielu množín ako:

$$L^C = A^* \setminus L,$$

t. j. ako všetky také reťazce nad abecedou A, ktoré nepatria do jazyka L.

- $L_3^C = A^* \setminus L_3$. Keďže L_3 obsahuje všetky reťazce nad abecedou $\{a, b, c\}$, ktoré **začínajú** symbolom a, tak L_3^C sú všetky reťazce zo symbolov $\{a, b, c\}$, ktoré **nezačínajú** symbolom a. Teda prázdny reťazec ε, všetky reťazce začínajúce symbolom b a všetky reťazce začínajúce symbolom c, čo môžeme formálne popísať ako $L_3^C = \{\varepsilon\} \cup \{bw \mid w \in \{a, b, c\}^*\} \cup \{cw \mid w \in \{a, b, c\}^*\}$. Teda jazyk $L_3^C = \{\varepsilon, b, c, ba, bb, bc, ca, cb, cc, baa, ...\}$.

4. V prípade prieniku dvoch jazykov $L_i \cap L_j$ je výsledkom taký jazyk, ktorý obsahuje len tie reťazce, ktoré súčasne patria aj do jazyka L_i, aj do jazyka L_j:

$$L_i \cap L_j = \{w \mid w \in L_i \wedge w \in L_j\}$$

Pre riešenie tohto typu úlohy je teda potrebné hľadať tie reťazce, ktoré patria do oboch jazykov:

- $L_2 \cap L_3$ je jazyk tvorený prienikom jazykov L_2 a L_3, teda reťazcami, ktoré sú súčasne z jazyka $L_2 = \{aa, ab, ba, bb\}$ a súčasne z jazyka L_3, teda začínajú symbolom a. To spĺňajú 2 reťazce z L_2, konkrétne aa a ab, teda jazyk $L_2 \cap L_3 = \{aa, ab\}$.
- $L_2 \cap L_4$ budú tvoriť tie reťazce z L_2, ktoré sú zároveň reťazcami v tvare $a^n cb^n$, $n \in \mathbb{N}_0$, (jazyk L_4). Keďže ako vidíme, **žiaden** reťazec z jazyka L_2 neobsahuje symbol c, **neexistujú** také reťazce, ktoré by patrili súčasne do L_2 aj L_4, a teda $L_2 \cap L_4 = \emptyset$, teda tzv. prázdny jazyk (jazyk bez reťazcov).
- $L_3 \cap L_4$ tvoria tie reťazce, ktoré začínajú symbolom a (jazyk L_3) a súčasne sú v tvare $a^n cb^n$, $n \in \mathbb{N}_0$, (jazyk L_4). V podstate skoro všetky reťazce z jazyka L_4 začínajú symbolom a, s výnimkou reťazca c. Ten teda do prieniku $L_3 \cap L_4$ patriť nebude a jazyk $L_3 \cap L_4 = \{a^n cb^n \mid n \in \mathbb{N}\} = \{acb, aacbb, aaacbbb, ...\}$.

V prípade rozdielu dvoch jazykov $L_i \setminus L_j$ je výsledkom taký jazyk, ktorý obsahuje len tie reťazce, ktoré súčasne patria do jazyka L_i a nepatria do jazyka L_j:

$$L_i \setminus L_j = \{w \mid w \in L_i \wedge w \notin L_j\}$$

Pri rozdiele jazykov je potrebné dbať na poradie jazykov v zápise, pretože rozdiel nie je komutatívny, t. j. vo všeobecnosti $L_i \setminus L_j \neq L_j \setminus L_i$:

- $L_2 \setminus L_3$ predstavuje rozdiel množín L_2 a L_3, teda ide o jazyk, ktorý tvoria tie reťazce z jazyka L_2, ktoré zároveň nepatria do jazyka L_3. Znamená to, že z jazyka L_2 uvažujeme iba tie reťazce, ktoré **nezačínajú** symbolom a, teda $L_2 \setminus L_3 = \{ba, bb\}$.
- $L_3 \setminus L_2$ predstavuje rozdiel množín L_3 a L_2, teda ide o jazyk, ktorý tvoria tie reťazce z jazyka L_3, ktoré zároveň nepatria do jazyka L_2, teda nie sú aa, ab, ba alebo bb. Výsledný jazyk teda vznikne tak, že z L_3 odstránime aa a ab, teda $L_3 \setminus L_2 = \{a, ac, aaa, aab, aac, ...\}$.

- $L_4 \setminus L_3$ predstavuje rozdiel množín L_4 a L_3, teda ide o jazyk, ktorý tvoria tie ret'azce z jazyka L_4, ktoré zároveň nepatria do jazyka L_3. Znamená to, že z jazyka L_4 uvažujeme iba tie ret'azce, ktoré **nezačínajú** symbolom a, teda iba ret'azec c, ked'že všetky ostatné ret'azce z L_4 začínajú symbolom a. Výsledok je teda $L_4 \setminus L_3 = \{c\}$.
- $L_3 \setminus L_4$ predstavujú ret'azce, ktoré začínajú symbolom a, avšak súčasne **nie sú** ret'azcami tvaru $a^n cb^n$, $n \geq 0$. Čiže napríklad aj ked' ret'azce $acb, aacbb, aaacbbb$ pôvodne patrili do jazyka L_3, do jazyka $L_3 \setminus L_4$ už patrit' nebudú.

V prípade zjednotenia dvoch jazykov $L_i \cup L_j$ je výsledkom taký jazyk, ktorý obsahuje tie ret'azce, ktoré patria alebo do jazyka L_i, alebo do jazyka L_j:

$$L_i \cap L_j = \{w \mid w \in L_i \vee w \in L_j\}$$

Pre riešenie tohto typu úlohy je teda potrebné hl'adat' tie ret'azce, ktoré patria aspoň do jedného z jazykov L_i, L_j:

- $L_1 \cup L_2$ je jazyk, ktorý obsahuje všetky ret'azce z jazyka L_1 a z jazyka L_2, teda $L_1 \cup L_2 = \{c, aa, ab, ba, bb\}$.
- $L_1 \cup L_4$ je jazyk, ktorý obsahuje všetko z jazyka L_1, t. j. ret'azec c a všetky ret'azce tvaru $a^n cb^n, n \in \mathbb{N}_0$. Ked'že ret'azec z L_1 zároveň patrí aj do L_4, resp. L_1 je podmnožinou L_4, tak výsledkom zjednotenia je samotný jazyk L_4, $L_1 \cup L_4 = L_4$.
- $L_3 \cup L_3^C$ je jazyk, ktorý vznikne ako zjednotenie všetkých ret'azcov zo symbolov $\{a, b, c\}$ začínajúcich symbolom a (jazyk L_3) a všetkých ret'azcov zo symbolov $\{a, b, c\}$ nezačínajúcich symbolom a (jazyk L_3^C). Ked'že množiny L_3 a L_3^C sú vzájomnými doplnkami v rámci jazyka všetkých ret'azcov A^* nad abecedou A, tak zjednotenie $L_3 \cup L_3^C$ sú vlastne **všetky ret'azce** nad abecedou A^*, teda $L_3 \cup L_3^C = A^*$.

Úloha č. 1.2.2 Sú dané nasledovné jazyky nad abecedou $A = \{a, b\}$:

- $L_1 = \{ww^R \mid w \in \{a, b\}^*\}$
- $L_2 = \{ww \mid w \in \{a, b\}^*\}$
- $L_3 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv \sharp_b(w) \bmod 2\}$[†]
- $L_4 = \{w \in \{a, b\}^* \mid \sharp_a(w) = \sharp_b(w)\}$

Určte:

1. $L_1 \cap L_2, L_1 \setminus L_2, L_2 \setminus L_1$,
2. $L_3 \cap L_4, L_3 \setminus L_4, L_4 \setminus L_3$.

[†] $\sharp_a(w)$ označuje počet výskytov symbolu a v ret'azci w

Riešenie:

1. Jazyk L_1 tvoria palindrómy párnej dĺžky zo symbolov a, b, pretože ide o ret'azce tvorené predponou w, za ktorou nasleduje jej zrkadlový obraz w^R. Napríklad ret'azec $aabbaa$ vznikne zret'azením $w = aab$ a jeho zrkadlového obrazu $w^R = baa$. Patrí sem aj prázdny ret'azec, keď'že $\varepsilon = \varepsilon\varepsilon^R$. Príklady ret'azcov z jazyka $L_1 = \{\varepsilon, aa, bb, aaaa, abba, baab, bbbb, aaaaaa, aabbaa, abaaba, baaaab, ...\}$.

   Jazyk L_2 je známy ako tzv. kopírovací jazyk (angl. *copy language*), ktorý tvoria ret'azce (v tomto prípade zo symbolov $\{a, b\}$), ktoré je možné rozdelit' na 2 menšie identické ret'azce. Napríklad $abbabb$ je ret'azec, ktorý možno rozdelit' na 2 menšie identické kópie, $w = abb$. Patrí sem aj prázdny ret'azec, keď'že $\varepsilon = \varepsilon\varepsilon$. Ďalšie ret'azce $L_2 = \{\varepsilon, aa, bb, aaaa, abab, baba, bbbb, aaaaaa, aabaab, ...\}$.

   - Ich prienikom je teda jazyk, ktorý tvoria také ret'azce, ktoré pozostávajú z predpony w zret'azenej s jej zrkadlovým obrazom w^R, avšak zároveň platí, že predpona w je totožná so svojim zrkadlovým obrazom w^R (t. j. predpona w je zároveň palindrómom, $w = w^R$):

     $$L_1 \cap L_2 = \{ww^R \mid w \in \{a, b\}^* \land w = w^R\}$$

   - Konkrétne $L_1 \cap L_2 = \{\varepsilon, aa, bb, aaaa, bbbb, aaaaaa, abaaba, babbab, bbbbbb, ...\}$
   - Rozdiel $L_1 \setminus L_2$ budú tvorit' tie ret'azce, ktoré sú palindrómami párnej dĺžky zo symbolov a, b avšak zároveň sa **nedajú** rozdelit' na 2 identické menšie podret'azce:

     $$L_1 \setminus L_2 = \{ww^R \mid w \in \{a, b\}^* \land w \neq w^R\}$$

   - Konkrétne $L_1 \setminus L_2 = \{abba, baab, aabbaa, abbbba, baaaab, bbaabb, ...\}$
   - Rozdiel $L_2 \setminus L_1$ budú tvorit' tie ret'azce, ktoré sa dajú rozdelit' na 2 identické menšie podret'azce, avšak tieto menšie podret'azce nie sú vzájomnými zrkadlovými obrazmi:

     $$L_2 \setminus L_1 = \{ww \mid w \in \{a, b\}^* \land w \neq w^R\}$$

   - Konkrétne $L_2 \setminus L_1 = \{abab, baba, aabaab, abbabb, baabaa, bbabba, ...\}$

2. Jazyk L_3 tvoria ret'azce zo symbolov a, b, v ktorých je rovnaká parita počtu symbolov a a b. Inými slovami, počet a a počet b v ret'azci je alebo súčasne párny, alebo súčasne nepárny. Napríklad v ret'azci $abaa$ máme 3 symboly a, $\sharp_a(w) = 3$, a 1 symbol b, $\sharp_b(w) = 1$. Keď'že 3 a 1 sú obe nepárne čísla, ret'azec $abaa$ patrí do jazyka L_3, $abaa \in L_3$. Naopak, ret'azec aab by do tohto jazyka nepatril, pretože obsahuje párny počet symbolov a a nepárny počet symbolov b. Rovnako ret'azec aaa by do tohto jazyka nepatril, pretože obsahuje nepárny počet symbolov a (3) a párny počet symbolov b (0). Jazyk $L_3 = \{\varepsilon, ab, ba, aa, bb, aaab, aaba, abaa, baaa, ...\}$.

Jazyk L_4 tvoria reťazce zo symbolov a, b, v ktorých je rovnaký počet symbolov a a b. Teda napríklad reťazec ε patrí do jazyka L_4, pretože obsahuje 0-krát symbol a a 0-krát symbol b. Rovnako reťazce $aabb, abba, baab, bbaa, abab, baba$ patria do tohto jazyka, pretože všetky obsahujú 2-krát symbol a a 2-krát symbol b.

- Ich prienikom je teda jazyk, ktorý tvoria reťazce, v ktorých je identický počet symbolov a a b a zároveň má ich výskyt rovnakú paritu. Ak sa však nad tým zamyslíme, tak **každý** reťazec, ktorý má rovnaký počet symbolov a a b, má určite zároveň aj rovnakú paritu, teda každý reťazec z jazyka L_4 určite patrí aj do jazyka L_3, teda platí $L_4 \subset L_3$

$$L_3 \cap L_4 = L_4.$$

- Rozdiel $L_3 \setminus L_4$ budú tvoriť tie reťazce, ktoré majú síce rovnakú paritu počtu symbolov a a b, avšak tieto počty nebudú rovnaké. Napríklad reťazec ab, ktorý patrí do jazyka L_3, už nebude patriť do $L_3 \setminus L_4$, pretože súčasne obsahuje rovnaký počet a a b.

$$L_3 \setminus L_4 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv \sharp_b(w) \bmod 2 \wedge \sharp_a(w) \neq \sharp_b(w)\}$$

- Konkrétne $L_3 \setminus L_4 = \{aa, bb, aaaa, aaab, aaba, abaa, abbb, baaa, babb, bbab, bbba, ...\}$
- Rozdiel $L_4 \setminus L_3$ budú tvoriť tie reťazce, ktoré majú rovnaký počet symbolov a a b, avšak tieto počty nemajú rovnakú paritu. To je samozrejme logický nemožné, keďže neexistuje reťazec, ktorý má rovnaký počet symbolov a a b (t. j. súčasne majú alebo párny počet a a b, alebo nepárny počet a a b), avšak by ich parity boli rôzne. Preto:

$$L_4 \setminus L_3 = \emptyset.$$

1.3 Derivácie reťazcov v gramatikách

Úloha č. 1.3.1 Je daná formálna gramatika $G = (N, T, P, S)$, kde $N = \{S, A\}$, $T = \{0, 1\}$, S je počiatočný neterminál a pravidlá gramatiky P:

- $S \rightarrow 0A$
- $A \rightarrow 0A \mid 1A \mid 0 \mid 1$

1. Určte typ gramatiky (regulárna, bezkontextová, kontextová, frázová).
2. Zistite, či existuje, a ak áno, nájdite odvodenie slov $0101, 0111, 1000$ v danej gramatike.
3. Určte, aký jazyk $L(G)$ gramatika generuje (slovne, formálnym zápisom).

Riešenie:

1. Typ gramatiky určíme podľa tvaru pravidiel. Táto gramatika je regulárna, pretože všetky jej pravidlá spĺňajú jeden z tvarov:

   - $A \to x$, kde $A \in N, x \in T^*$ (vľavo neterminál, vpravo reťazec terminálov),
   - $A \to yB$, kde $A, B \in N, y \in T^+$ (vľavo neterminál, vpravo neprázdny reťazec terminálov nasledovaný neterminálom).

2. Derivácie reťazcov v tejto úlohe budeme hľadať skusmo.

   - Reťazec 0101 deriváciu v gramatike má, nájdeme ju postupnou aplikáciou pravidiel:
     $$S \Rightarrow 0A \Rightarrow 01A \Rightarrow 010A \Rightarrow 0101$$
   - Reťazec 0111 deriváciu v gramatike má, nájdeme ju postupnou aplikáciou pravidiel:
     $$S \Rightarrow 0A \Rightarrow 01A \Rightarrow 011A \Rightarrow 0111$$
   - Reťazec 1000 deriváciu v gramatike nemá. Ak by sme ju skúšali hľadať, vidíme, že každá derivácia v tejto gramatike musí ako prvé pravidlo použiť $S \to 0A$ (pretože nemáme k dispozícii iné pravidlo pre neterminál S), t. j.:

     $$S \Rightarrow 0A$$

     Avšak tým vyrobíme vetnú formu, ktorá začína nulou, ktorú nevieme odstrániť. Keďže reťazec, ktorého deriváciu hľadáme, 1000, začína jednotkou, vidíme, že takáto derivácia nemôže dospieť k reťazcu 1000:

     $$S \Rightarrow 0A \not\Rightarrow^* 1000$$

     preto v tomto prípade reťazec 1000 v danej gramatike nemá deriváciu.

3. Jazyk $L(G)$, ktorý gramatika generuje, zistíme v tomto prípade pohľadom na pravidlá gramatiky:

   - Ako prvé pravidlo sa vždy použije $S \to 0A$, pretože iné pravidlá pre neterminál S gramatika neobsahuje.
   - Toto pravidlo vyrobí na začiatku vetnej formy terminál 0, za ktorým nasleduje neterminál A. Všimnite si, že keďže gramatika je regulárna, terminál 0 na začiatku vetnej formy už nie je možné odstrániť / zmeniť na iný terminál.
   - Následne z neterminálu A vieme v tejto gramatike vyrobiť alebo 0, alebo 1, alebo $0A$, alebo $1A$. Teda z neterminálu A vieme takýmto rekurzívnym spôsobom generovať ľubovoľné reťazce z núl a jednotiek dĺžky aspoň 1, t. j. obsahujúce aspoň jednu nulu alebo jednotku.

- To znamená, že jazyk, ktorý táto gramatika generuje, vieme slovne popísať ako množinu reťazcov z núl a jednotiek, ktoré začínajú nulou, za ktorou nasleduje ľubovoľný neprázdny reťazec zložený z núl a jednotiek.
- Formálne by sme ho vedeli popísať ako $L(G) = \{0w \mid w \in \{0,1\}^+\}$.

Úloha č. 1.3.2 Je daná formálna gramatika $G = (N, T, P, S)$, kde $N = \{S, A, B\}$, $T = \{a, b, c\}$, S je počiatočný neterminál a pravidlá gramatiky P:

- $S \to AcB$
- $A \to aAb \mid \varepsilon$
- $B \to bBa \mid \varepsilon$

1. Určte typ gramatiky.
2. Zistite, či existujú odvodenia slov $abc, aabbcba, bac$ v danej gramatike.
3. Popíšte jazyk generovaný gramatikou $L(G)$ (slovne, formálnym popisom).

Riešenie:

1. Táto gramatika je bezkontextová, pretože všetky jej pravidlá spĺňajú tvar:
   - $A \to \alpha$, kde $A \in N, \alpha \in (N \cup T)^*$.
   - Teda, na ľavej strane každého pravidla je jeden neterminál a na pravej strane každého pravidla je ľubovoľný reťazec zložený zo symbolov gramatiky (symbolmi gramatiky rozumieme terminály a neterminály).
2. Deriváciu reťazca abc nájdeme nasledovným spôsobom:
   - Vidíme, že v gramatike máme na začiatku na výber len pravidlo $S \to AcB$:
   $$S \Rightarrow AcB$$
   - Ak chceme teraz na začiatku vetnej formy vyrobiť terminál a, ktorým začína reťazec abc, z pravidiel pre neterminál A si vyberieme to pravidlo, ktoré tam tento terminál vyrobí, t. j. $A \to aAb$:
   $$S \Rightarrow AcB \Rightarrow aAbcB$$
   - Na záver aplikujeme pravidlá $A \to \varepsilon, B \to \varepsilon$, aby sme z vetnej formy odstránili neterminály A, B a dostávame:
   $$S \Rightarrow AcB \Rightarrow aAbcB \Rightarrow abcB \Rightarrow abc$$

Podobne nájdeme deriváciu reťazca *aabbcba*:

$$S \Rightarrow AcB \Rightarrow aAbcB \Rightarrow aaAbbcB \Rightarrow aabbcB \Rightarrow aabbcbBa \Rightarrow aabbcba$$

Hľadanie derivácie reťazca *bac*:

- V prvom kroku máme na výber len pravidlo $S \rightarrow AcB$:

$$S \Rightarrow AcB$$

- Ak chceme teraz na začiatku vetnej formy vyrobiť terminál b, ktorým začína reťazec *bac*, vidíme, že ani jedno z pravidiel aplikovateľných na neterminál A nám tento terminál nedokáže vyrobiť! V prípade aplikácie pravidla $A \rightarrow aAb$ by sme dostali:

$$S \Rightarrow AcB \Rightarrow aAbcB$$

  teda vetnú formu $aAbcB$, ktorá začína terminálom a, ktorý v bezkontextovej gramatike nie je možné prepísať na iný terminál, a teda týmto spôsobom by sme nevedeli dostať reťazec *bac*,
- Ak by sme na neterminál A aplikovali druhé pravidlo, $A \rightarrow \varepsilon$:

$$S \Rightarrow AcB \Rightarrow cB$$

  dostávame vetnú formu cB, ktorá začína terminálom c, teda znovu niečo, čo nedokážeme upraviť na reťazec *bac*.
- Tým pádom v tejto gramatike nie je možné generovať reťazec *bac*, teda odvodiť ho z počiatočného neterminálu, $S \not\Rightarrow^* bac$.

3. Aby sme popísali jazyk, ktorý generuje táto gramatika, pozrime sa na jej pravidlá:

- Prvé pravidlo $S \rightarrow AcB$ vyrobí vetnú formu:

$$S \Rightarrow AcB$$

  Vidíme teda, že výsledné reťazce budú obsahovať terminál c, pred ktorým budú časti odvoditeľné z neterminálu A a za ktorým budú časti odvoditeľné z neterminálu B.
- Pravidlá $A \rightarrow aAb \mid \varepsilon$ nám v kombinácii m-aplikácií pravidla $A \rightarrow aAb$ a aplikácie pravidla $A \rightarrow \varepsilon$ dovoľujú generovať reťazce tvaru $a^m b^m$, $m \in \mathbb{N}_0$:

$$A \Rightarrow aAb \Rightarrow a^2Ab^2 \Rightarrow a^3Ab^3 \Rightarrow ... \Rightarrow a^mAb^m \Rightarrow a^mb^m$$

- Pravidlá $B \rightarrow bBa \mid \varepsilon$ nám v kombinácii n-aplikácií pravidla $B \rightarrow bBa$ a aplikácie pravidla $B \rightarrow \varepsilon$ dovoľujú generovať reťazce tvaru $b^n a^n$, $n \in \mathbb{N}_0$:

$$B \Rightarrow bBa \Rightarrow b^2Ba^2 \Rightarrow b^3Ba^3 \Rightarrow ... \Rightarrow b^nBa^n \Rightarrow b^na^n$$

- To znamená, že v uvedenej gramatike sú možné derivácie † nasledovného typu:

$$S \Rightarrow AcB \Rightarrow^m a^m A b^m cB \Rightarrow a^m b^m cB \Rightarrow^n a^m b^m cb^n Ba^n \Rightarrow a^m b^m cb^n a^n$$

Teda reťazce odvoditeľné v tejto gramatike sú vo všeobecnosti v tvare $a^m b^m cb^n a^n$, kde $m, n \in \mathbb{N}_0$.

Ide teda o reťazce, ktoré na začiatku obsahujú m-krát symbol a nasledovaný rovnakým počtom symbolov b, za ktorými sa nachádza symbol c, za ním sa nachádza n-krát symbol b nasledovaný rovnakým počtom symbolov a.

$L(G) = \{a^m b^m cb^n a^n \mid m, n \in \mathbb{N}_0\}$=$\{c, abc, cba, abcba, aabbc, aabbcba, aabbcbbaa, ...\}$.

Úloha č. 1.3.3 Je daná formálna gramatika $G = (N, T, P, S)$, kde $N = \{S, A\}$, $T = \{a, b, c\}$, S je počiatočný neterminál a pravidlá gramatiky P:

- $S \to aS \mid SbA \mid Aa$
- $A \to bAAa \mid bb$
- $aSb \to cSa$

1. Určte typ gramatiky.
2. Nájdite odvodenie slova *cbbaabb* v danej gramatike.

Riešenie:

1. Táto gramatika je kontextová, pretože všetky jej pravidlá spĺňajú tvar:
    - $\alpha \to \beta$, kde $\alpha \in (N \cup T)^* N (N \cup T)^*, \beta \in (N \cup T)^+$ a platí $|\alpha| \leq |\beta|$.
    - Teda, na ľavej strane každého pravidla je reťazec symbolov gramatiky s aspoň 1 neterminálom, na pravej strane každého pravidla je neprázdny reťazec symbolov gramatiky a zároveň pravá strana každého pravidla predstavuje reťazec symbolov gramatiky, ktorý je aspoň takej dĺžky ako ľavá strana pravidla.

    Len pre upozornenie, táto gramatika nie je bezkontextovou gramatikou, pretože obsahuje pravidlo $aSb \to cSa$, v ktorom sa na ľavej strane nenachádza len jeden neterminál, ale postupnosť viacerých symbolov gramatiky.

2. Deriváciu reťazca *cbbaabb* nájdeme nasledovným spôsobom:

† $\Rightarrow^m$ označuje skrátený zápis m-krokov derivácie

- Vidíme, že reťazec začína symbolom c. Pokúsime sa teda najprv vytvoriť symbol c na začiatku vetnej formy. Vidíme, že symbol c vo vetnej forme vie vyrobiť len pravidlo $aSb \to cSa$. Aby sme ho mohli použiť, musíme však mať najprv vo vetnej forme reťazec aSb. Ten si vieme vyrobiť napríklad nasledovnou deriváciou:
$$S \Rightarrow aS \Rightarrow aSbA$$
- Tým dostávame na začiatku vetnej formy reťazec aSb, ktorý teraz pravidlom $aSb \to cSa$ nahradíme reťazcom cSa:
$$S \Rightarrow aS \Rightarrow aSbA \Rightarrow cSaA$$
- Dostávame teda vetnú formu začínajúcu symbolom c a obsahujúcu terminál a. Reťazec, ktorého deriváciu hľadáme, $cbbaabb$ obsahuje dvakrát symboly a, pričom za druhým výskytom symbolu a obsahuje príponu bb — v našom prípade vieme odvodiť reťazec bb z neterminálu A použitím pravidla $A \to bb$:
$$S \Rightarrow aS \Rightarrow aSbA \Rightarrow cSaA \Rightarrow cSabb$$
- V ďalšej fáze z neterminálu S, ktorý nám zostal vo vetnej forme, odvodíme reťazec bba pomocou pravidiel $S \to Aa$ a $A \to bb$, čím dostávame výslednú deriváciu:
$$S \Rightarrow aS \Rightarrow aSbA \Rightarrow cSaA \Rightarrow cSabb \Rightarrow cAaabb \Rightarrow cbbaabb$$

Všimnite si teda, že v prípade kontextových (a aj frázových) gramatík môžeme pomocou pravidiel prepisovať terminálne symboly vo vetných formách.

1.4 Konštrukcie gramatík

Úloha č. 1.4.1 Je daný jazyk $L = \{xaby \mid x \in \{a,b\}^*, y \in \{a,b\}^*\}$ nad abecedou $A = \{a,b\}$. Nájdite formálnu gramatiku $G = (N,T,P,S)$, ktorá generuje jazyk L.

Riešenie: Pri zostrojení gramatiky G, ktorá generuje nejaký požadovaný jazyk L, musíme dbať na to, aby boli splnené 2 podmienky:

1. Každý reťazec, ktorý bude gramatika G generovať, musí byť zároveň reťazcom patriacim do jazyka L, teda musí platiť $L(G) \subseteq L$.

2. Každý reťazec z jazyka L musí mať v gramatike G deriváciu, teda musí platiť $L \subseteq L(G)$.

Ak sú tieto 2 podmienky splnené, potom platí $L(G) = L$, teda jazyk $L(G)$ generovaný gramatikou G je totožný s jazykom L.

Konštrukciu gramatiky je dobré začať tým, že si vymenujeme aspoň najkratšie reťazce patriace do jazyka L, aby sme videli, aké reťazce vlastne potrebujeme generovať. Zadaný jazyk $L = \{xaby \mid x \in \{a,b\}^*, y \in \{a,b\}^*\}$ tvoria také reťazce, ktoré sú zložené zo symbolov a, b a ktoré obsahujú ako podreťazec časť ab. Pred týmto reťazcom ab sa môže nachádzať ľubovoľná postupnosť symbolov a, b označená x a za týmto reťazcom ab sa môže nachádzať ľubovoľná postupnosť symbolov a, b označená y. Patria sem teda napríklad reťazce:

- ab, kde $x = \varepsilon, y = \varepsilon$
- aba, kde $x = \varepsilon, y = a$
- aab, kde $x = a, y = \varepsilon$
- $aaba$, kde $x = a, y = a$
- $baba$, kde $x = b, y = a$

Preto aj naša konštrukcia gramatiky začne tým, že pre nejaký počiatočný neterminál S pridáme pravidlo, ktoré nám zaručí vznik tohto podreťazca ab a zároveň si pripravíme 2 nové neterminály A a B, ktoré budú slúžiť na deriváciu predpony x, resp. prípony y:

- $S \to AabB$

Teraz potrebujeme zabezpečiť, aby sa z neterminálu A, resp. B, dal odvodiť ľubovoľný reťazec symbolov a, b. Napríklad:

- $A \to aA \mid bA \mid \varepsilon$
- $B \to aB \mid bB \mid \varepsilon$

Všimnite si, že pravidlá $A \to \varepsilon$ a $B \to \varepsilon$ nám dovolia odvodiť aj také reťazce, kde $x = \varepsilon$, resp. $y = \varepsilon$, čo je v súlade s daným jazykom.

Výsledná gramatika G, ktorá generuje jazyk L, bude teda obsahovať neterminály $N = \{S, A, B\}$, terminály $T = \{a, b\}$, počiatočný neterminál bude neterminál S a pravidlá P:

- $S \to AabB$
- $A \to aA \mid bA \mid \varepsilon$
- $B \to aB \mid bB \mid \varepsilon$

Skontrolujme, aspoň neformálne, či sú splnené nasledovné podmienky:

1. Platí $L(G) \subseteq L$?

- Zisťujeme, či každý reťazec, ktorý naša gramatika generuje, spĺňa zároveň podmienku jazyka L.
- V našej gramatike máme zaručené, že výsledné reťazce určite obsahujú podreťazec ab (vďaka pravidlu $S \to AabB$, ktoré sa použije v každej derivácii). Pred týmto / za týmto podreťazcom sú vo výslednom reťazci určite len postupnosti symbolov a, b, takže určite každý derivovaný reťazec zároveň spĺňa aj podmienku príslušnosti do jazyka L, teda platí $L(G) \subseteq L$.

2. Platí $L \subseteq L(G)$?

- Zisťujeme, či každý reťazec, ktorý patrí do jazyka L, má zároveň v našej gramatike G deriváciu.
- Uvažujme ľubovoľný reťazec patriaci do jazyka L, t. j. reťazec tvaru $xaby$, kde x, y sú ľubovoľné reťazce zložené zo symbolov a, b. V prvom kroku derivácie vieme vyrobiť spomínaný podreťazec ab:

$$S \Rightarrow AabB$$

- V našej gramatike je možné z neterminálu A odvodiť ľubovoľný reťazec symbolov a, b, teda nech by predpona x bola ľubovoľná, vždy ju budeme vedieť odvodiť z neterminálu A, teda

$$A \Rightarrow^* x, x \in \{a, b\}^*$$

- Podobne vieme z neterminálu B odvodiť ľubovoľný reťazec symbolov a, b, teda nech by prípona y bola ľubovoľná, vždy ju budeme vedieť odvodiť z neterminálu B, teda

$$B \Rightarrow^* y, y \in \{a, b\}^*$$

- Teda určite budeme vedieť pre ľubovoľný reťazec tvaru $xaby$, kde $x, y \in \{a, b\}^*$, nájsť v našej gramatike G deriváciu, a teda platí $L \subseteq L(G)$.

Keďže $L(G) \subseteq L$ a zároveň $L \subseteq L(G)$, tak určite platí $L(G) = L$, čo znamená, že jazyk $L(G)$ generovaný gramatikou G je totožný s jazykom L, a teda je naša gramatika správna.

Je dobré uviesť, že uvedená gramatika je **bezkontextová** a rozhodne nie je jedinou správnou gramatikou generujúcou jazyk $L = \{xaby \mid x \in \{a, b\}^*, y \in \{a, b\}^*\}$. Dá sa totižto nájsť aj nasledovná regulárna gramatika $G = (\{S, A\}, \{a, b\}, P, S)$, ktorá generuje ten istý jazyk:

- $S \to aS \mid bS \mid abA$
- $A \to aA \mid bA \mid \varepsilon$

Úloha č. 1.4.2 Sú dané nasledovné jazyky nad abecedou $A = \{a, b\}$. Nájdite gramatiky, ktoré tieto jazyky generujú.

1. $L_1 = \{abwba \mid w \in \{a, b\}^*\}$,
2. $L_2 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv 1 \bmod 3\}$,
3. $L_3 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv \sharp_b(w) \bmod 2\}$.

Riešenie:

1. Prvý jazyk tvoria reťazce zo symbolov a, b začínajúce predponou ab a končiace príponou ba. Je pomerne jednoduché nájsť bezkontextovú gramatiku, ktorá ho generuje. Nech jej neterminály sú $N = \{S, A\}$, terminály sú $T = \{a, b\}$, S bude počiatočný neterminál a pravidlá P:

   - $S \to abAba$
   - $A \to aA \mid bA \mid \varepsilon$

   Pre uvedený jazyk existuje aj regulárna gramatika, s neterminálmi $N = \{S, A\}$, terminálmi $T = \{a, b\}$, S bude počiatočný neterminál a pravidlá P:

   - $S \to abA$
   - $A \to aA \mid bA \mid ba$

2. Druhý jazyk je tvorený reťazcami zo symbolov a, b, v ktorých počet symbolov a po delení tromi dáva zvyšok 1, teda počet výskytov symbolov a v reťazcoch je 1, 4, 7, 10 atď. Riešením je napríklad gramatika s neterminálmi $N = \{S, A, B\}$, terminálmi $T = \{a, b\}$, S bude počiatočný neterminál a pravidlá P:

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

   Táto gramatika je regulárna, teda vetná forma vždy obsahuje len 1 neterminál, ako svoj posledný symbol. Navyše sme ju skonštruovali podľa nasledovného princípu:

   (a) Ak sa na konci aktuálnej vetnej formy nachádza neterminál S, tak sa priebežne vygeneroval počet symbolov a deliteľný tromi.

   (b) Ak sa na konci aktuálnej vetnej formy nachádza neterminál A, tak sa priebežne vygeneroval počet symbolov a, ktorý po delení tromi dáva zvyšok 1.

   (c) Ak sa na konci aktuálnej vetnej formy nachádza neterminál B, tak sa priebežne vygeneroval počet symbolov a, ktorý po delení tromi dáva zvyšok 2.

Keďže každá derivácia môže v tejto gramatike skončiť len ak sa aplikuje pravidlo $A \to \varepsilon$, tak to znamená, že všetky derivované reťazce budú obsahovať taký počet symbolov a, ktorý po delení tromi dáva zvyšok 1, teda že pred finálnou aplikáciou pravidla $A \to \varepsilon$ stál na konci vetnej formy neterminál A, teda $L(G) \subseteq L$.

Zároveň vidíme, že ľubovoľný reťazec, ktorý obsahuje taký počet symbolov a, že po delení 3 dáva zvyšok 1, má v gramatike deriváciu, pretože máme pre každý neterminál pravidlo, ktoré dokáže vyrobiť aj terminál a, aj terminál b, čiže bez ohľadu na to, ako reťazec vyzerá, bude mať v gramatike deriváciu, teda platí $L \subseteq L(G)$.

Gramatika G teda generuje jazyk $L_2 = \{w \in \{a,b\}^* \mid \sharp_a(w) \equiv 1 \bmod 3\}$.

3. Tretí jazyk je tvorený reťazcami zo symbolov a, b, v ktorých je súčasne počet symbolov a a počet symbolov b alebo párny, alebo súčasne nepárny. Takou gramatikou je napríklad gramatika s neterminálmi $N = \{S, A\}$, terminálmi $T = \{a, b\}$, S bude počiatočný neterminál a pravidlá P:

   - $S \to aA \mid bA \mid \varepsilon$
   - $A \to aS \mid bS$

   Táto gramatika je regulárna, teda vetná forma vždy obsahuje len 1 neterminál, ako svoj posledný symbol. Navyše sme ju skonštruovali podľa nasledovného princípu:

   (a) Ak sa na konci aktuálnej vetnej formy nachádza neterminál S, tak sa priebežne vygeneroval taký počet symbolov a a b, že ich parita je rovnaká.

   (b) Ak sa na konci aktuálnej vetnej formy nachádza neterminál A, tak sa priebežne vygeneroval taký počet symbolov a a b, že ich parita nie je rovnaká.

   Keďže derivácia môže končiť len pomocou pravidla $S \to \varepsilon$, teda ak sa na konci vetnej formy nachádza S, tak bude končiť práve v prípade, že sa odvodil počet a a počet b s rovnakou paritou.

Úloha č. 1.4.3 Sú dané nasledovné formálne jazyky s príslušnými abecedami. Nájdite gramatiky, ktoré tieto jazyky generujú.

1. $L_1 = \{a^n b^n c^k \mid n, k \in \mathbb{N}_0\}$ nad abecedou $A = \{a, b, c\}$.
2. $L_2 = \{ww^R \mid w \in \{0,1\}^*\}$ nad abecedou $A = \{0, 1\}$.
3. $L_3 = \{w \mid w \in \{a,b\}^*, \sharp_a(w) = \sharp_b(w)\}$ nad abecedou $A = \{a, b\}$.

Riešenie:

1. Prvý jazyk tvoria reťazce zo symbolov a, b, c ktoré si môžeme rozdeliť na 2 nezávislé podreťazce: predponu tvaru $a^n b^n, n \in \mathbb{N}_0$ a príponu tvaru $c^k, k \in \mathbb{N}_0$. Touto logikou zostrojíme aj príslušnú gramatiku $G = (\{S, A, B\}, \{a, b, c\}, P, S)$:

- $S \to AB$
- $A \to aAb \mid \varepsilon$
- $B \to cB \mid \varepsilon$

Neterminál A slúži na deriváciu predpony tvaru $a^n b^n$, neterminál B zase na deriváciu prípony tvaru c^k. Keďže n a k sú číselné hodnoty, ktoré sú na sebe nezávislé, neterminály A a B spolu nijako nesúvisia. Keďže výsledné reťazce musia byť zreťazením podreťazcov $a^n b^n$ a c^k, prvé pravidlo $S \to AB$ zabezpečí, že derivácia reťazcov $a^n b^n c^k$ bude obsahovať aj neterminál A, aj neterminál B.

2. Druhý jazyk je tvorený palindrómami nad abecedou $\{0, 1\}$ párnej dĺžky, teda do tohto jazyka patria napríklad reťazce: $\varepsilon, 00, 11, 0000, 0110, 1001, 1111, 000000, 001100$ atď. Pozostávajú teda z predpony w, ktorú tvorí nejaký reťazec núl a jednotiek a prípony w^R, ktorú tvorí zrkadlový obraz predpony w. Každej nule / jednotke v slove w teda zodpovedá nejaká nula / jednotka v zrkadlovom obraze w^R. Túto logiku použijeme aj pri zostrojení gramatiky a pravidlá navrhneme tak, že v jednom kroku bude možné odvodiť nulu/jednotku súčasne v slove w a súčasne v jeho zrkadlovom obraze w^R. Výsledná gramatika $G = (\{S\}, \{0, 1\}, P, S)$:

   - $S \to \varepsilon \mid 0S0 \mid 1S1$

   Všimnite si, že pravidlá $S \to 0S0$ a $S \to 1S1$ v každom kroku vytvoria nulu/jednotku vľavo od neterminálu S a súčasne vpravo od neterminálu S. Nula/jednotka generovaná pred neterminálom S je súčasťou slova w, nula/jednotka generovaná za neterminálom S je súčasťou slova w^R. Napríklad derivácia nuly vľavo/vpravo od neterminálu S:

$$S \Rightarrow 0S0$$

   Ak teraz znovu aplikujeme jedno z pravidiel, skôr generované terminály sa posunú smerom k začiatku/koncu vetnej formy a aktuálne generované terminály budú stáť pred/za neterminálom S. Napríklad, ak by sme v ďalšom kroku použili pravidlo $S \to 1S1$:

$$S \Rightarrow 0S0 \Rightarrow 01S10$$

   skôr vygenerované nuly boli odsunuté „ku krajom“ vetnej formy a vygenerované jednotky sú bližšie k stredu vetnej formy. Ak deriváciu ukončíme pravidlom $S \to \varepsilon$, máme zaručené, že výsledok je tvaru ww^R, kde $w \in \{0, 1\}$, napr.:

$$S \Rightarrow 0S0 \Rightarrow 01S10 \Rightarrow 0110$$

3. Tretí jazyk je tvorený reťazcami nad abecedou $\{a, b\}$, v ktorých je počet symbolov a a počet symbolov b rovnaký. Medzi takéto reťazce patria napríklad: $\varepsilon, ab, ba, aabb, abab, abba, baab, baba, bbaa$ atď. Pri zostrojení takejto gramatiky musíme mať na pamäti, že gramatika bude garantovať, že:

- Počet symbolov a a b bude rovnaký, teda že napríklad ku každému terminálu a existuje vo výslednom ret'azci terminál b a naopak.
- Že nie sú kladené žiadne požiadavky na tvar výsledných slov, teda že jazyk obsahuje **všetky** permutácie ret'azcov s nula výskytom a a b, s jedným symbolom a a b, s dvomi symbolmi a a b atd'. Napríklad pre 2 symboly a a b musí gramatika generovat' všetky takéto ret'azce, teda $aabb, abab, abba, baab, baba, bbaa$.

Takouto gramatikou je napríklad gramatika s neterminálom $N = \{S\}$, terminálmi $T = \{a, b\}$, S bude počiatočný neterminál a pravidlami P:

- $S \to SaSbS \mid SbSaS \mid \varepsilon$

Táto gramatika je založená na nasledovnej myšlienke:

- V každom ret'azci x, ktorý obsahuje rovnaký počet symbolov a a b, vieme vybrat' jeden symbol a a jeden symbol b tak, že:
  (a) Pred prvým z týchto symbolov je predpona ret'azca x, ret'azec x_1, ktorý znovu obsahuje rovnaký počet symbolov a a b,
  (b) Medzi týmito symbolmi je podret'azec ret'azca x, ret'azec x_2, ktorý znovu obsahuje rovnaký počet symbolov a a b,
  (c) Za druhým z týchto symbolov je prípona ret'azca x, ret'azec x_3, ktorý znovu obsahuje rovnaký počet symbolov a a b.
- Napríklad v ret'azci $x = aabbba$ vieme vybrat' symboly označené **tučným písmom**, $x = \mathbf{a}ab\mathbf{b}ba$, a príslušné segmenty x_1, x_2, x_3 sú:
  (a) $x_1 = \varepsilon$
  (b) $x_2 = ab$
  (c) $x_3 = ba$
- Vidíme, že vo všetkých 3 podret'azcoch je rovnaký počet symbolov a a b.
- Gramatika G je teda skonštruovaná rekurzívne. Ak je možné z neterminálu S odvodit' ret'azec s rovnakým počtom a a b, tak určite aj $SaSbS$, resp. $SbSaS$ bude obsahovat' rovnaký počet symbolov.
- Navyše oba tvary pravidla, t. j. $S \to SaSbS, S \to SbSaS$ zaručia, že je možné odvodit' alebo situáciu, že a sa nachádza v ret'azci niekde pred príslušným symbolom b, alebo naopak, symbol b sa nachádza v ret'azci niekde pred príslušným symbolom a.

Alternatívnym riešením by bola gramatika $G = (\{S, A, B\}, \{a, b\}, P, S)$ s pravidlami:

- $S \to aBS \mid bAS \mid \varepsilon$
- $A \to a \mid bAA$
- $B \to b \mid aBB$

Táto gramatika je založená na nasledovnej myšlienke:

- Z neterminálu S sa generujú ret'azce s rovnakým počtom a a b. Ak je na začiatku takéhoto ret'azca symbol a (pravidlo $S \to aBS$), tak sa zároveň vo vetnej forme vyrobí neterminál B, ktorý signalizuje chýbajúci terminál b vo vetnej forme, aby bola zaručená rovnost' počtu symbolov. Analogicky, ak chceme z neterminálu S odvodit' ret'azec začínajúci b, potom použijeme pravidlo $S \to bAS$, ktoré nám vo vetnej forme vyrobí neterminál A signalizujúci chýbajúci terminál a.
- V každom momente sa vo vetnej forme nachádza tol'ko neterminálov A a B, kol'ko terminálov a a b ešte potrebujeme vygenerovat', aby bol počet a a b vo vetnej forme rovnaký.

Napríklad derivácia:

$$S \Rightarrow aBS \Rightarrow aaBBS \Rightarrow aaBB$$

vyrobí 2 terminály a a zároveň vo vetnej forme vyrobí 2 neterminály B, ktoré signalizujú nepomer v počte symbolov. Aby mohla derivácia úspešne skončit' je potrebné tieto neterminály prepísat' na príslušné terminály:

$$S \Rightarrow aBS \Rightarrow aaBBS \Rightarrow aaBB \Rightarrow aabB \Rightarrow aabb$$

Úloha č. 1.4.4 Sú dané nasledovné formálne jazyky s príslušnými abecedami. Nájdite gramatiky, ktoré tieto jazyky generujú.

1. $L_1 = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ nad abecedou $A = \{a, b, c\}$.
2. $L_2 = \{ww \mid w \in \{a, b\}^*\}$ nad abecedou $A = \{a, b\}$.

Riešenie:

1. Jazyk L_1 tvoria ret'azce, ktoré na začiatku obsahujú n symbolov a, za ktorými nasleduje n symbolov b, za ktorými nasleduje n symbolov c, pričom n je celé číslo hodnoty aspoň 1. Teda príklady ret'azcov jazyka $L_1 = \{abc, aabbcc, aaabbbccc, aaaabbbbcccc, ...\}$. Tento jazyk patrí medzi typické kontextové jazyky a príslušná gramatika G by mohla vyzerat' napríklad nasledovne:

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

- $S \to aBC \mid aSBC$
- $CB \to BC$
- $aB \to ab$
- $bB \to bb$
- $bC \to bc$
- $cC \to cc$

Táto gramatika využíva vlastnosti kontextových gramatík, v ktorých je možné meniť vo vetných formách v jednom kroku derivácie skupiny symbolov gramatiky.

Jej princíp spočíva v tom, že pomocou pravidiel $S \to aBC, S \to aSBC$ sa vo vetnej forme vygeneruje toľko terminálov a, koľkými začína reťazec, ktorého deriváciu hľadáme. Tieto pravidlá zároveň vo vetnej forme vyrobia rovnaký počet neterminálov B a C ako terminálov a. Napríklad pre reťazec $aaabbbccc$ by sme vyrobili vetnú formu:

$$S \Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC$$

teda vieme dostať vetnú formu tvaru $a^n(BC)^n$. V ďalšej fáze opakovaním pravidla $CB \to BC$ dokážeme vetnú formu upraviť do tvaru $a^nB^nC^n$, teda usporiadame neterminály B a C:

$$\begin{aligned} S &\Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow aaaBCBBCC \Rightarrow \\ &\Rightarrow aaaBBCBCC \Rightarrow aaaBBBCCC \end{aligned}$$

a v záverečnej fáze využijeme zvyšné pravidlá, aby sme postupne zamenili neterminály B za terminály b a neterminály C za terminály c:

$$\begin{aligned} S &\Rightarrow aSBC \Rightarrow aaSBCBC \Rightarrow aaaBCBCBC \Rightarrow aaaBCBBCC \Rightarrow \\ &\Rightarrow aaaBBCBCC \Rightarrow aaaBBBCCC \Rightarrow aaabBBCCC \Rightarrow aaabbBCCC \Rightarrow \\ &\Rightarrow aaabbbCCC \Rightarrow aaabbbcCC \Rightarrow aaabbbccC \Rightarrow aaabbbccc \end{aligned}$$

Upozornenie! Prepis neterminálov B na terminály b (rovnako C na c) nemožno realizovať jednoducho pomocou pravidiel $B \to b$ (resp. $C \to c$), pretože v takom prípade by bolo možné napríklad v gramatike odvodiť aj reťazec $aaabcbcbc$ pomocou $S \Rightarrow^* aaaBCBCBC \Rightarrow^* aaabcbcbc$, čo nie je reťazec z jazyka L_1! Preto je nutné prepis neterminálov na terminály riešiť postupne, vzhľadom na **kontext** (t. j. okolité neterminály), v ktorom sa terminály nachádzajú.

2. Jazyk L_2 tvoria reťazce, ktoré pozostávajú z dvoch menších identických podreťazcov w zložených zo symbolov $\{a, b\}$, t. j. tzv. kopírovací jazyk. Jazyk

$L_2 = \{\varepsilon, aa, bb, aaaa, abab, baba, bbbb, aaaaaa, aabaab, ...\}$. Tento jazyk je ďalším predstaviteľom jazykov, pre ktoré neexistuje bezkontextová gramatika, ktorá by ich generovala. Príslušná (frázová) gramatika G by mohla vyzerať napríklad nasledovne: $G = (\{S, S_1, A, B, C, X, Y, Z\}, \{a, b\}, P, S)$, kde pravidlá P:

- $S \to S_1Z$
- $S_1 \to aS_1A \mid bS_1B \mid C$
- $AZ \to XZ$
- $AX \to XA$
- $BX \to XB$
- $CX \to aC$
- $BZ \to YZ$
- $AY \to YA$
- $BY \to YB$
- $CY \to bC$
- $CZ \to \varepsilon$

Gramatika je navrhnutá tak, že najprv vyrobí na konci vetnej formy špeciálny ukončovaní symbol - neterminál Z.

Následne sa pomocou pravidiel $S_1 \to aS_1A, S_1 \to bS_1B$ vygeneruje slovo w ako predpona vetnej formy a pomocou pravidla $S_1 \to C$ sa vo vetnej forme vyrobí neterminál C, ktorý bude označovať ukončenie tejto predpony. Napríklad pre reťazec $abbabb$, v ktorom $w = abb$:

$$S \Rightarrow S_1Z \Rightarrow aS_1AZ \Rightarrow abS_1BAZ \Rightarrow abbS_1BBAZ \Rightarrow abbCBBAZ$$

Tým vo vetnej forme vzniká predpona w a medzi symbolmi C a Z dostávame zrkadlový obraz slova w, v ktorom však nie sú terminálne symboly, ale im zodpovedajúce neterminálne symboly. V ďalšej fáze potrebujeme teraz tieto neterminálne symboly presunúť od konca vetnej formy (t. j. od neterminálu Z) k neterminálu C a nahradiť ich za príslušné terminálne symboly. Na to slúžia neterminály X resp. Y, ktoré označujú presun terminálu a, resp. b:

$$\begin{aligned} S &\Rightarrow^* abbCBBAZ \Rightarrow abbCBBXZ \Rightarrow abbCBXBZ \Rightarrow abbCXBBZ \Rightarrow \\ &\Rightarrow abbaCBBZ \Rightarrow abbaCBYZ \Rightarrow abbaCYBZ \Rightarrow abbabCBZ \Rightarrow \\ &\Rightarrow abbabCYZ \Rightarrow abbabbCZ \end{aligned}$$

Na záver, po utriedení a zámene všetkých neterminálov A a B za terminály a a b dostávame na konci vetnej formy príponu CZ, ktorá signalizuje, že vetná forma je v tvare $wwCZ$, kde w je požadovaný prefix, a teda môžeme skupinu CZ odstrániť pomocou pravidla $CZ \to \varepsilon$.

Kapitola 2

Konečné automaty

2.1 Deterministické konečné automaty

Úloha č. 2.1.1 Je daný deterministický konečný automat (DKA) $M = (Q, \Sigma, \delta, q_0, F)$, kde stavy konečného automatu $Q = \{q_0, q_1, q_2, q_3, q_4\}$, vstupná abeceda $\Sigma = \{a, b\}$, počiatočný stav automatu je q_0, akceptačné stavy sú $F = \{q_3, q_4\}$ a prechodová funkcia δ je daná tabuľkou 2.1:

δ	a	b
q_0	q_1	q_2
q_1	q_4	q_3
q_2	q_4	q_3
q_3	q_4	q_3
q_4	q_4	q_3

Tabuľka 2.1: Prechodová funkcia DKA z úlohy 2.1.1.

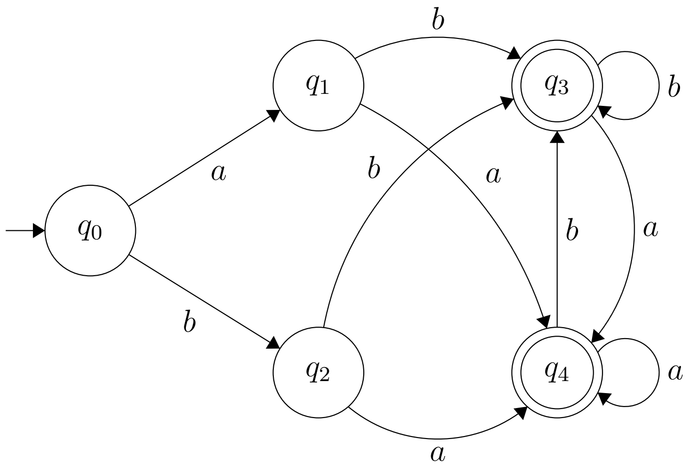
1. Nakreslite grafickú reprezentáciu daného DKA pomocou prechodového diagramu.
2. Zistite, či daný DKA akceptuje reťazce: $aa, ab, a, b, \varepsilon$.
3. Určte, aký jazyk $L(M)$ akceptuje daný DKA.

Riešenie:

1. Prechodový diagram reprezentujúci DKA predstavuje nasledovný orientovaný graf:
   - Každý vrchol tohto grafu predstavuje jeden stav DKA.
   - Ak je v DKA možný prechod zo stavu q_i do stavu q_j na symbol c, t. j. v prechodovej funkcii $\delta(q_i, c) = q_j$, potom v grafe vedie orientovaná hrana z vrcholu q_i do vrcholu q_j ohodnotená symbolom c.

- Do vrcholu predstavujúceho počiatočný stav vedie neohodnotená hrana.
- Vrcholy predstavujúce akceptačné stavy sú označené dvojitou kružnicou.

Príslušný prechodový diagram je znázornený na obrázku 2.1:



Obr. 2.1: Prechodový diagram DKA z úlohy 2.1.1

2. Výpočty daného DKA pre jednotlivé reťazce:

    - Každý výpočet začína v počiatočnom stave, pričom na vstupe je celý vstupný reťazec. Výpočet zapisujeme pomocou tzv. konfigurácií, ktoré predstavujú dvojicu: (aktuálny stav, neprečítaná časť vstupného slova).
    - Ak DKA **úspešne prečítal celý reťazec**, teda na vstupe zostal už len prázdny reťazec ε, a zároveň DKA **skončil** v jednom z **akceptačných** stavov, hovoríme, že DKA **akceptuje** vstupný reťazec.
    - V prípade, že DKA **nedočítal** celý reťazec, alebo v prípade, že po jeho kompletnom prečítaní **neskončil** v akceptačnom stave, hovoríme, že DKA vstupný reťazec **neakceptuje**.
    - Reťazec aa : $(q_0, aa) \vdash (q_1, a) \vdash (q_4, \varepsilon)$. Vidíme, že vstupný reťazec bol celý spracovaný, pretože na vstupe zostal už len prázdny reťazec. Výpočet zároveň skončil v stave q_4, ktorý je akceptačným stavom, teda DKA slovo aa akceptuje.
    - Reťazec ab : $(q_0, ab) \vdash (q_1, b) \vdash (q_3, \varepsilon)$. Vstupný reťazec bol celý spracovaný a výpočet skončil v stave q_3, ktorý je akceptačný, teda DKA slovo ab akceptuje.
    - Reťazec a : $(q_0, a) \vdash (q_1, \varepsilon)$. Vstupný reťazec bol celý spracovaný a výpočet skončil v stave q_1, ktorý nie je akceptačným stavom. DKA teda reťazec a neakceptuje.

- Ret'azec b : $(q_0, b) \vdash (q_2, \varepsilon)$. Vstupný ret'azec bol celý spracovaný a výpočet skončil v stave q_2, ktorý nie je akceptačným stavom. DKA teda ret'azec b neakceptuje.
- Ret'azec ε : (q_0, ε). Ak je na vstupe prázdny ret'azec, výpočet DKA končí. Teda vstupný ret'azec ε považujeme za spracovaný hneď na začiatku výpočtu. Ked'že v tomto prípade skončil výpočet v stave q_0, ktorý nie je akceptačným stavom, DKA ret'azec ε neakceptuje.

3. Jazyk $L(M)$ akceptovaný konečným automatom tvoria všetky ret'azce nad vstupnou abecedou konečného automatu, ktoré tento automat akceptuje.

   - Aby sme určili, aký jazyk automat akceptuje, položme si otázku, ako musia vyzerat' ret'azce, pre ktoré výpočet začínajúci v počiatočnom stave q_0 dospeje do niektorého z akceptačných stavov — v tomto prípade q_3 alebo q_4.
   - Vidíme, že na to, aby sme sa dostali zo stavu q_0 do stavu q_3 alebo q_4 potrebujeme prečítat' na vstupe l'ubovol'nú postupnost' 2 symbolov: aa, ab, ba, bb. Všetky 4 vedú do jedného z akceptačných stavov.
   - Ďalej vidíme, že ak sa daný DKA dostane do jedného z akceptačných stavov q_3, q_4, tak bez ohl'adu na to, aké budú d'alšie symboly na vstupe, sa DKA bude nachádzat' alebo v stave q_3, alebo v stave q_4.
   - To znamená, že ak bude na vstupe **l'ubovol'ný** ret'azec zo symbolov $\{a, b\}$, ktorý je dĺžky aspoň 2, tak ho automat dokáže celý spracovat' a skončí alebo v stave q_3, alebo v stave q_4, teda bude daný ret'azec akceptovat'.
   - Preto jazyk tohto automatu tvorí množina všetkých ret'azcov zo symbolov $\{a, b\}$, ktoré sú dĺžky aspoň 2, t. j. $L(M) = \{w \in \{a, b\}^* \mid |w| \geq 2\}$.

Úloha č. 2.1.2 Je daný neúplný deterministický konečný automat (DKA) $M = (Q, \Sigma, \delta, q_0, F)$, kde stavy konečného automatu $Q = \{q_0, q_1, q_2\}$, vstupná abeceda $\Sigma = \{0, 1\}$, počiatočný stav automatu je q_0, akceptačný stav je $F = \{q_0\}$ a prechodová funkcia δ je daná tabul'kou 2.2:

δ	0	1
q_0	q_1	
q_1	q_0	q_2
q_2		q_0

Tabul'ka 2.2: Prechodová funkcia DKA z úlohy 2.1.2

1. Doplňte DKA na úplný deterministický konečný automat.
2. Zistite, či DKA akceptuje ret'azce: $\varepsilon, 00, 011, 100$.
3. Určte, aký jazyk $L(M)$ akceptuje uvedený DKA.

Riešenie:

1. Daný DKA je neúplný, pretože jeho prechodová funkcia δ neobsahuje prechody pre všetky kombinácie stavov a vstupných symbolov. Konkrétne chýbajú definované prechody $\delta(q_0, 1)$ a $\delta(q_2, 0)$.

   Doplnenie neúplného DKA na úplný DKA je možné vykonať nasledovným spôsobom:

   - Do množiny stavov daného DKA sa pridá nový neakceptačný stav — tzv. pasca, q_p.
   - Všetky doteraz nedefinované prechody sa definujú ako prechody do pasce q_p.
   - Rovnako sa doplnia prechody z pasce q_p na všetky vstupné symboly vedúce znovu do pasce q_p.

   Teda príslušný úplný DKA by v danom prípade bol deterministický konečný automat so stavmi $Q = \{q_0, q_1, q_2, q_p\}$ a prechodovou funkciou δ:

δ	0	1
q_0	q_1	q_p
q_1	q_0	q_2
q_2	q_p	q_0
q_p	q_p	q_p

2. Výpočty daného (pôvodného neúplného) DKA pre jednotlivé reťazce:

   - $\varepsilon : (q_0, \varepsilon)$. Automat reťazec ε akceptuje.
   - $00 : (q_0, 00) \vdash (q_1, 0) \vdash (q_0, \varepsilon)$. Automat reťazec 00 akceptuje.
   - $011 : (q_0, 011) \vdash (q_1, 11) \vdash (q_2, 1) \vdash (q_0, \varepsilon)$. Automat reťazec 011 akceptuje.
   - $100 : (q_0, 100)$. Automat nemá definovaný prechod pre kombináciu stavu q_0 a vstupného symbolu 1. Preto sa v danej konfigurácii výpočet zastaví. Keďže vstupný reťazec 100 sa nepodarilo celý spracovať, automat reťazec neakceptuje.

   V úplnej verzii DKA by bol výpočet reťazca 100 nasledovný:

   - $100 : (q_0, 100) \vdash (q_p, 00) \vdash (q_p, 0) \vdash (q_p, \varepsilon)$. V tomto prípade sa reťazec 100 podarilo celý spracovať a výpočet skončil v stave q_p. Keďže q_p nie je akceptačný stav, automat reťazec 100 neakceptuje.

3. Keďže tento automat bude akceptovať len tie reťazce, po ktorých spracovaní skončí v akceptačnom stave q_0, skúmajme, pre aké reťazce sa automat vie do tohto stavu dostať z počiatočného stavu q_0:

- Keďže q_0 je zároveň aj počiatočný stav, tento automat určite akceptuje prázdny reťazec ε.
- V automate existuje cesta zo stavu q_0 cez stav q_1 do stavu q_0 pre postupnosť symbolov 00.
- Podobne existuje cesta zo stavu q_0 cez stavy q_1 a q_2 do stavu q_0 pre postupnosť symbolov 011.
- Iné cesty z počiatočného stavu do akceptačného stavu v tomto automate nie sú.
- To znamená, že každý reťazec, ktorý bude automat akceptovať, bude pozostávať z podreťazcov $\{00, 011\}$.
- Výsledný jazyk akceptovaný automatom je teda v tomto prípade $L(M) = \{00, 011\}^*$ (do tohto jazyka patrí aj spomínaný ε).

Úloha č. 2.1.3 Je daný jazyk $L = \{xaby \mid x \in \{a,b\}^*, y \in \{a,b\}^*\}$ nad abecedou $A = \{a, b\}$. Nájdite deterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$, ktorý akceptuje jazyk L.

Riešenie: Pri zostrojení automatu M, ktorý akceptuje nejaký požadovaný jazyk L, musíme, analogicky s konštrukciou gramatiky, dbať na to, aby boli splnené 2 podmienky:

1. Každý reťazec, ktorý bude automat M akceptovať, musí byť zároveň reťazcom patriacim do jazyka L, teda musí platiť $L(M) \subseteq L$.
2. Každý reťazec z jazyka L musí mať v automate M akceptačný výpočet, teda musí platiť $L \subseteq L(M)$.

Ak sú tieto 2 podmienky splnené, potom platí $L(M) = L$, teda jazyk $L(M)$ akceptovaný automatom M je totožný s jazykom L.

Podobne, ako tomu bolo pri konštrukcii gramatiky, aj pri konštrukcii automatu je dobré začať tým, že si vymenujeme aspoň najkratšie reťazce patriace do jazyka L, aby sme videli, aké reťazce vlastne potrebujeme akceptovať. Do zadaného jazyka $L = \{xaby \mid x \in \{a,b\}^*, y \in \{a,b\}^*\}$ patria napríklad reťazce:

- ab, kde $x = \varepsilon, y = \varepsilon$
- aba, kde $x = \varepsilon, y = a$
- aab, kde $x = a, y = \varepsilon$
- $aaba$, kde $x = a, y = a$
- $baba$, kde $x = b, y = a$

Ako je zo zadaného jazyka zrejmé, každý ret'azec, ktorý obsahuje ab ako svoj podret'azec, by mal DKA akceptovat'. Ciel'om bude teda zostrojit' DKA tak, aby v prípade, že bude na vstupe detegovaná postupnost' symbolov ab, prešiel do akceptačného stavu, v ktorom už len dočíta zvyšok vstupu.

Ked'že zadaný jazyk L obsahuje len ret'azce zložené zo symbolov $\{a, b\}$, aj automat zostrojíme tak, že jeho vstupnou abecedou budú len tieto symboly, $\Sigma = \{a, b\}$.

Na začiatku je automat v počiatočnom stave q_0. Stav q_0 bude predstavovat' situáciu, že sme zatial' na vstupe nerozpoznali ani hl'adanú postupnost' ab, ani jej predponu a. Ak sa teda automat nachádza v stave q_0, tak v závislosti na aktuálnom vstupnom symbole môžu nastat' 2 situácie:

- Ak je na vstupe symbol a, **môže** íst' o čast' hl'adaného podret'azca ab. Automat sa teda presunie do nejakého nového stavu, označíme ho q_a, ktorý bude signalizovat', že posledný čítaný symbol bol a, teda sme potenciálne rozpoznali predponu a hl'adanej sekvencie ab. Teda v prechodovej funkcii $\delta(q_0, a) = q_a$.
- V prípade, že na vstupe je symbol b, určite nemôže íst' o symbol z hl'adanej postupnosti ab, ked'že sme v stave q_0, teda sme zatial' nerozpoznali na vstupe ani predponu a hl'adanej postupnosti ab. Preto v danej situácii zostávame v stave q_0, teda $\delta(q_0, b) = q_0$.

Ak sa automat ocitne v stave q_a, znamená to, že posledný symbol čítaný zo vstupu bol a. V závislosti na aktuálnom vstupnom symbole môžu nastat' 2 situácie:

- Ak je na vstupe symbol a, znamená to, že predchádzajúci symbol a nebol súčast'ou hl'adanej sekvencie a, avšak aktuálny symbol a **môže** byt' predponou hl'adaného podret'azca ab. Automat teda zostane v stave q_a, čo je v korešpodencii s tým, že stav q_a znamená, že „posledný čítaný symbol na vstupe bolo a“, $\delta(q_a, a) = q_a$.
- V prípade, že na vstupe je symbol b, tak to znamená, že sme museli narazit' na postupnost' ab na vstupe, pretože a nás dostalo do stavu q_a a aktuálne čítame symbol b. V takom prípade sa automat prepne do stavu, ktorý označíme ako q_{ab}, ktorý bude signalizovat', že sme niekde v rámci čítania vstupu rozpoznali postupnost' ab, $\delta(q_a, b) = q_{ab}$.

Ak sa automat ocitne v stave q_{ab}, znamená to, že v rámci vstupu bola rozpoznaná postupnost' ab, teda celý vstup je potrebné akceptovat', ked'že ide o slovo z jazyka L. Bez ohl'adu na to, aký je nasledovný vstupný symbol, zostane automat v stave q_{ab}:

- $\delta(q_{ab}, a) = q_{ab}$.
- $\delta(q_{ab}, b) = q_{ab}$.

Skontrolujme, aspoň neformálne, či sú splnené nasledovné podmienky:

1. Platí $L(M) \subseteq L$?
   - Zisťujeme, či každý reťazec, ktorý náš automat akceptuje, spĺňa zároveň podmienku jazyka L.
   - Aby nami zostrojený automat akceptoval vstupný reťazec, musí dôjsť k prechodu zo stavu q_0 do stavu q_{ab}. K takémuto prechodu môže dôjsť len cez stav q_a. Ak sa automat nachádza v stave q_a, znamená to, že posledný čítaný symbol na vstupe bol a. Ak následne automat prejde zo stavu q_a do stavu q_{ab}, znamená to, že nasledujúci symbol na vstupe bol b, teda že aktuálne posledne čítané 2 symboly na vstupe boli ab. Teda celý vstupný reťazec obsahuje ab ako podreťazec, čiže vstupné slovo zároveň patrí do jazyka L, teda platí $L(M) \subseteq L$.
2. Platí $L \subseteq L(M)$?
   - Zisťujeme, či každý reťazec, ktorý patrí do jazyka L, má zároveň v automate akceptačný výpočet.
   - Uvažujme ľubovoľný reťazec w patriaci do jazyka L, t. j. reťazec tvaru $w = xaby$, $x, y \in \{a, b\}^*$. Zároveň môžeme predpokladať, že ak reťazec w obsahuje podreťazec ab viackrát, tak rozhodujúcim je jeho prvý výskyt, teda že predpona x neobsahuje ab ako svoj podreťazec.
   - Ak je teda w slovo z jazyka L, potom sa dá ukázať, že predpona x musí byť tvaru $x = b^*a^*$.
     - V prípade, že predpona x neobsahuje symbol/symboly a, tak potom platí: $(q_0, xaby) \vdash^* (q_0, aby)$, teda po spracovaní tejto predpony zostáva automat v stave q_0. Keďže následne sa na vstupe nachádza postupnosť ab, automat po jej spracovaní dospeje do stavu q_{ab}, v ktorom už len zostane spracovaním prípony y.
     - V prípade, že predpona x obsahuje symbol/symboly a, tak potom tvoria jej príponu a platí: $(q_0, xaby) \vdash^* (q_a, aby)$, teda spracovaním tejto predpony sa automat dostane do stavu q_a. Keďže následne sa na vstupe nachádza postupnosť ab, automat po jej spracovaní dospeje do stavu q_{ab}, v ktorom už len zostane spracovaním prípony y.
   - V oboch uvedených prípadoch automat dospeje do akceptačného stavu q_{ab}, teda platí, že ak je na vstupe automatu ľubovoľný reťazec tvaru $xaby$, kde $x, y \in \{a, b\}^*$, tak v automate existuje jeho akceptačný výpočet a teda platí $L \subseteq L(M)$.

Keďže $L(M) \subseteq L$ a zároveň $L \subseteq L(M)$, tak určite platí $L(M) = L$, čo znamená, že jazyk $L(M)$ akceptovaný automatom M je totožný s jazykom L, a teda je náš automat správny.

Len pre úplnosť, zostrojili sme deterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$, ktorého stavy $Q = \{q_0, q_a, q_{ab}\}$, vstupná abeceda $\Sigma = \{a, b\}$, q_0 je počiatočný stav, množina akceptačných stavov $F = \{q_{ab}\}$ a prechodová funkcia δ je daná tabuľkou 2.3:

δ	a	b
q_0	q_a	q_0
q_a	q_a	q_{ab}
q_{ab}	q_{ab}	q_{ab}

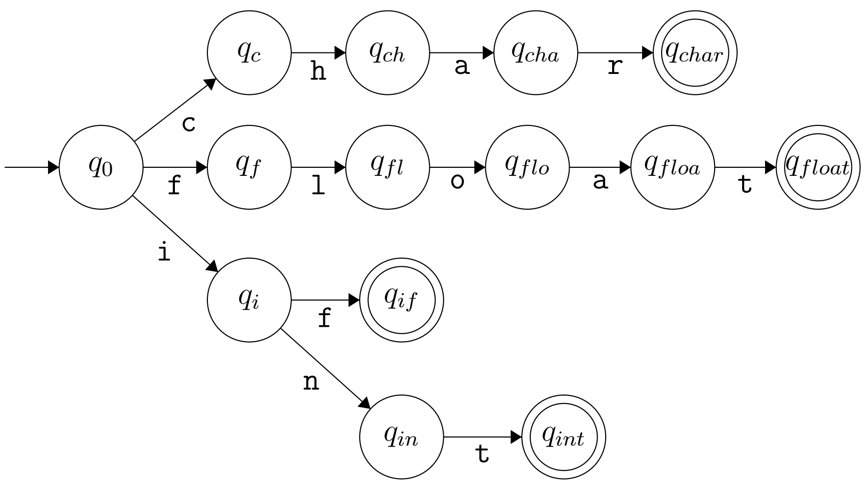
Tabuľka 2.3: Prechodová funkcia DKA z úlohy 2.1.3

Úloha č. 2.1.4 Sú dané nasledovné jazyky nad príslušnými abecedami. Nájdite deterministické konečné automaty, ktoré akceptujú príslušné jazyky.

1. $L_1 = \{\texttt{char}, \texttt{float}, \texttt{if}, \texttt{int}\}$ nad abecedou $A = \{\texttt{a}, \texttt{b}, ..., \texttt{z}\}$.
2. $L_2 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv 0 \bmod 3\}$ nad abecedou $\{a, b\}$.
3. $L_3 = \{w \in \{0, 1\}^* \mid$ prvý symbol w je iný ako posledný symbol $w\}$ nad abecedou $\{0, 1\}$.
4. $L_4 = \{w \in \{0, 1\}^* \mid w$ je binárny rozvoj nezáporného čísla deliteľného 3$\}$ nad abecedou $\{0, 1\}$.

Riešenie:

1. Jazyk L_1 je konečný jazyk tvorený 4 reťazcami, $\texttt{char}, \texttt{float}, \texttt{if}, \texttt{int}$. Stačí zostrojiť DKA tak, aby pre každý reťazec existovala samostatná vetva v automate, ktorá dokáže akceptovať príslušný reťazec. Riešením by mohol byť neúplný deterministický konečný automat na obrázku 2.2.



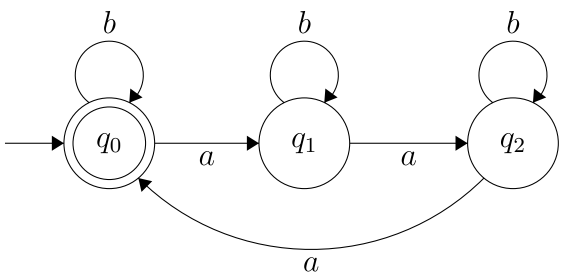
Obr. 2.2: Prechodový diagram DKA pre jazyk L_1 z úlohy 2.1.4

Pre úplnosť dodávame, že ide o deterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$, ktorého množina stavov $Q = \{q_0, q_c, q_{ch}, q_{cha}, q_{char}, q_f, q_{fl}, q_{flo}, q_{floa}, q_{float}, q_i, q_{if}, q_{in}, q_{int}\}$, vstupnou abecedou je $\Sigma = \{\mathtt{a}, \mathtt{b}, ..., \mathtt{z}\}$, q_0 je počiatočný stav, akceptačné stavy sú $F = \{q_{char}, q_{float}, q_{if}, q_{int}\}$ a prechodová funkcia δ je znázornená prechodovým diagramom na obrázku 2.2.

2. Jazyk L_2 je nekonečný jazyk tvorený reťazcami zo symbolov $\{a, b\}$, v ktorých je počet symbolov a deliteľný tromi. Medzi takéto reťazce patria:

   - Reťazce neobsahujúce symbol a: $\varepsilon, b, bb, bbb, bbbb, ...$
   - Reťazce obsahujúce 3 symboly a: $aaa, baaa, abaa, aaba, aaab,$
     $bbaaa, babaa, baaba, baaab, abbaa, ababa, abaab, ...$
   - Reťazce obsahujúce 6 symbolov a, 9 symbolov a atď.

   Príkladom takéhoto deterministického konečného automatu je automat $M = (Q, \Sigma, \delta, q_0, F)$ so stavmi $Q = \{q_0, q_1, q_2\}$, vstupnými symbolmi $\Sigma = \{a, b\}$, počiatočným stavom q_0, množinou akceptačných stavov $F = \{q_0\}$ a prechodovou funkciou δ zobrazenou na obrázku 2.3:



Obr. 2.3: Prechodový diagram DKA pre jazyk L_2 z úlohy 2.1.4

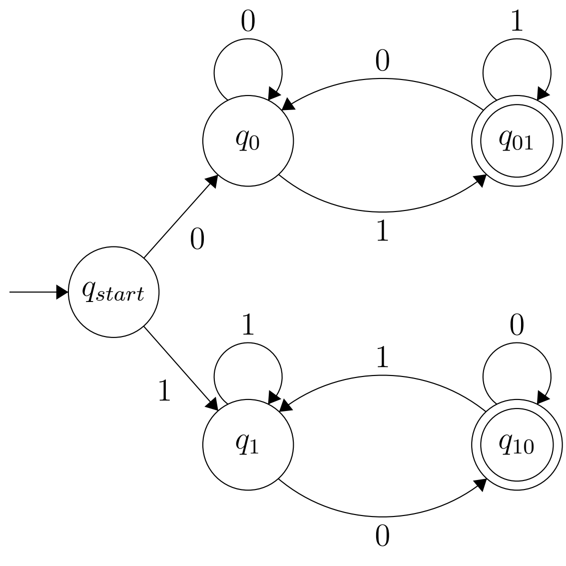
Tento DKA je zostrojený podľa nasledovného princípu:

   - DKA obsahuje 3 stavy: q_0, q_1, q_2, ktoré predstavujú, aký je zvyšok po delení 3 doteraz prečítaného počtu symbolov a vo vstupnom slove. Ak bol prečítaný počet symbolov a deliteľný tromi, automat sa nachádza v stave q_0, ak je zvyšok po delení 3 rovný jednej, je v stave q_1, resp. ak je zvyšok po delení 3 rovný dvom, je v stave q_2.
   - Na začiatku sa DKA nachádza v stave q_0, keďže ešte nebol prečítaný žiaden vstupný symbol, teda logicky bolo doteraz prečítaných nula symbolov a, čo je číslo deliteľné tromi.
   - Ak sa na vstupe číta symbol b, počet symbolov a sa nemení, preto sú v jednotlivých stavoch slučky pre symbol b.

- Ak sa na vstupe číta symbol a, zvyšok po delení 3 počtu symbolov a sa zvýši o 1. Samozrejme, ak bol doteraz prečítaný taký počet symbolov a, že po delení 3 dáva zvyšok 2 (stav q_2), tak ak sa prečíta ďalšie a, dostávame zvyšok po delení 3 nula (stav q_0).
- Keďže chceme akceptovať tie reťazce, ktoré obsahujú počet symbolov a deliteľný 3 (stav q_0), tak budeme akceptovať tie reťazce, po ktorých prečítaní skončí automat v stave q_0. Preto je stav q_0 akceptačným stavom.

3. Jazyk L_3 je nekonečný jazyk tvorený reťazcami zo symbolov $\{0, 1\}$, v ktorých je prvý symbol iný ako posledný. Medzi takéto reťazce patria:
   - Reťazce začínajúce nulou a končiace jednotkou: $01, 001, 011, 0001, 0011, 0101, 0111, ...$
   - Reťazce začínajúce jednotkou a končiace nulou: $10, 100, 110, 1000, 1010, 1100, 1110, ...$

Príkladom takéhoto deterministického konečného automatu je automat $M = (Q, \Sigma, \delta, q_{start}, F)$ so stavmi $Q = \{q_{start}, q_0, q_1, q_{01}, q_{10}\}$, vstupnými symbolmi $\Sigma = \{0, 1\}$, počiatočným stavom q_{start}, množinou akceptačných stavov $F = \{q_{01}, q_{10}\}$, prechodová funkcia δ je znázornená na obrázku 2.4.



Obr. 2.4: Prechodový diagram DKA pre jazyk L_3 z úlohy 2.1.4

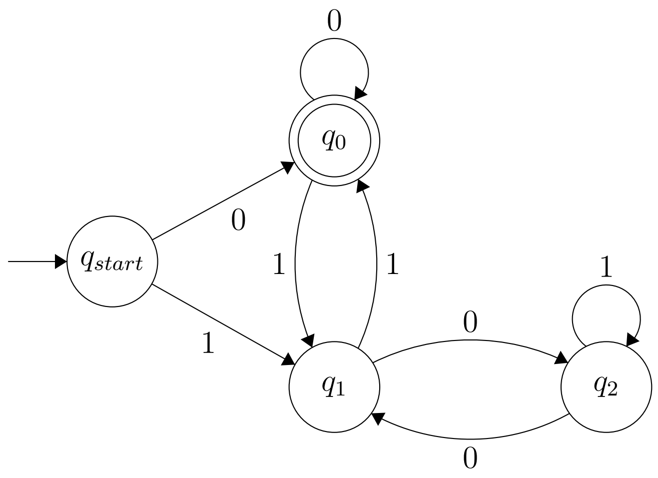
Tento DKA je zostrojený podľa nasledovného princípu:

- Na začiatku sa výpočet automatu vetví podľa prvého symbolu. Ak bol prvý symbol vstupu nula, automat sa prepne do stavu q_0, ak bol prvý symbol vstupu 1, automat sa prepne do stavu q_1.

- Následne sa v jednotlivých vetvách pokračuje v čítaní vstupných symbolov, pričom pre vetvu so stavom q_0 platí:
  - Ak bol posledný čítaný symbol 0, automat je v stave q_0.
  - Ak bol posledný čítaný symbol 1, automat je v stave q_{01}.
- Pre vetvu so stavom q_1 platí:
  - Ak bol posledný čítaný symbol 1, automat je v stave q_1.
  - Ak bol posledný čítaný symbol 0, automat je v stave q_{10}.
- Ak sa teda automat po prečítaní celého vstupu nachádza v stave q_{01} muselo to znamenať', že prvý symbol vstupu bola 0 a posledný symbol vstupu bola 1, teda vstup bol v tvare $0w1$, kde $w \in \{0,1\}^*$, teda prvý a posledný symbol sú rôzne.
- Analogicky, ak sa automat po prečítaní celého vstupu nachádza v stave q_{10} muselo to znamenať', že prvý symbol vstupu bola 1 a posledný symbol vstupu bola 0, teda vstup bol v tvare $1w0$, kde $w \in \{0,1\}^*$, teda prvý a posledný symbol sú rôzne.
- Ak sa automat po prečítaní celého vstupu ocitne v stave q_{start}, q_0 alebo q_1 znamená to:
  - q_{start} — automat mal na vstupe len prázdny ret'azec. Ten nemá rôzny prvý a posledný symbol, teda ho DKA nesmie akceptovat'.
  - q_0 — prvý a posledný symbol vstupu bola 0 (teda rovnaký symbol). DKA vstup nebude akceptovat'.
  - q_1 — prvý a posledný symbol vstupu bola 1 (teda rovnaký symbol). DKA vstup nebude akceptovat'.
- Stavy q_{01} a q_{10} budú teda akceptačnými stavmi.

4. Jazyk L_4 je nekonečný jazyk, tvorený ret'azcami zo symbolov $\{0,1\}$, ktoré predstavujú binárny rozvoj nezáporných čísiel delitel'ných 3:
   - Binárna reprezentácia čísla 0: 0
   - Binárna reprezentácia čísla 3: 11
   - Binárna reprezentácia čísla 6: 110
   - Binárna reprezentácia čísla 9: 1001
   - atd'.
   - Navyše navrhneme automat tak, aby akceptoval aj ret'azce s bezvýznamnými nulami zl'ava, t. j. napríklad akceptovaná binárna reprezentácia čísla 3 bude nielen 11, ale aj všetky ostatné binárne reprezentácie, ako napr. $011, 0011, 00011$ atd'.

Príkladom takéhoto deterministického konečného automatu je automat $M = (Q, \Sigma, \delta, q_{start}, F)$ so stavmi $Q = \{q_{start}, q_0, q_1, q_2\}$, vstupnými symbolmi



Obr. 2.5: Prechodový diagram DKA pre jazyk L_4 z úlohy 2.1.4

$\Sigma = \{0, 1\}$, počiatočným stavom q_{start}, množinou akceptačných stavov $F = \{q_0\}$, ktorého prechodová funkcia δ je znázornená na obrázku 2.5.

Tento DKA je zostrojený podľa nasledovného princípu počítania s binárnymi číslami:

- Predstavme si, že máme postupnosť bitov u reprezentujúcu nejaké dekadické číslo U. Ak vezmeme postupnosť bitov v, ktorá vznikne pridaním nuly k postupnosti u, t. j. $v = u0$, potom pre príslušné dekadické číslo V platí, že $V = 2U$. Napríklad pre $u = 110$ reprezentujúce $U = 6$ je reťazec $v = u0 = 1100$ binárny rozvoj čísla $V = 12 = 2U$.
- Ak k postupnosti u pridáme jednotku, t. j. $v = u1$, potom v predstavuje binárny rozvoj dekadického čísla V, pre ktoré platí $V = 2U + 1$. Napríklad pre $u = 110$ reprezentujúce $U = 6$ je reťazec $v = u1 = 1101$ binárny rozvoj čísla $V = 13 = 2U + 1$.

Použijúc tento princíp sme zostrojili uvedený DKA podľa nasledovnej logiky:

- DKA postupne načítava zo vstupu postupnosť bitov (núl a jednotiek). Po každom načítanom bite sa nachádza v takom stave, ktorý predstavuje **aktuálny zvyšok po delení 3** pre doteraz prečítanú postupnosť bitov.
- Na začiatku automat neprečítal žiaden bit zo vstupu, teda počiatočný stav q_{start} slúži na to, že sa z neho DKA prepne alebo do stavu q_0, ak prvý čítaný bit má hodnotu 0, pretože 0 má po delení 3 zvyšok 0, alebo do stavu q_1, ak prvý čítaný bit má hodnotu 1, pretože 1 má po delení 3 zvyšok 1.
- Ak doteraz prečítaná postupnosť bitov tvorí binárny rozvoj čísla deliteľného 3, DKA sa nachádza v stave q_0.

- Ak doteraz prečítaná postupnosť bitov tvorí binárny rozvoj čísla, ktoré po delení 3 dáva zvyšok 1 (resp. 2), DKA sa nachádza v stave q_1 (resp. q_2).
- Po prečítaní posledného bitu vstupu sa DKA nachádza v stave, ktorý predstavuje zvyšok po delení tromi čísla, ktorého binárny rozvoj bol na vstupe. Keďže chceme akceptovať len reťazce predstavujúce binárny rozvoj čísiel deliteľných 3, akceptačným stavom bude stav q_0.
- Prechody medzi stavmi q_0, q_1, q_2 reprezentujú zmenu aktuálne uvažovaného zvyšku po delení 3 podľa nasledovných vzťahov:
  - Ak $U \equiv 0 \bmod 3$, potom $2U \equiv 0 \bmod 3$. Teda $\delta(q_0, 0) = q_0$ (slovne: ak doteraz čítaná postupnosť bitov u predstavuje číslo U deliteľné tromi, aj $v = u0$, teda číslo $V = 2U$ predstavuje číslo deliteľné tromi).
  - Ak $U \equiv 0 \bmod 3$, potom $2U + 1 \equiv 1 \bmod 3$. Teda $\delta(q_0, 1) = q_1$. (slovne: ak doteraz čítaná postupnosť bitov u predstavuje číslo U deliteľné tromi, tak $v = u1$, teda číslo $V = 2U + 1$ predstavuje číslo, ktoré má po delení 3 zvyšok 1).
  - Ak $U \equiv 1 \bmod 3$, potom $2U \equiv 2 \bmod 3$. Teda $\delta(q_1, 0) = q_2$.
  - Ak $U \equiv 1 \bmod 3$, potom $2U + 1 \equiv 0 \bmod 3$. Teda $\delta(q_1, 1) = q_0$.
  - Ak $U \equiv 2 \bmod 3$, potom $2U \equiv 1 \bmod 3$. Teda $\delta(q_2, 0) = q_1$.
  - Ak $U \equiv 2 \bmod 3$, potom $2U + 1 \equiv 2 \bmod 3$. Teda $\delta(q_2, 1) = q_2$.

2.2 Nedeterministické konečné automaty

Úloha č. 2.2.1 Je daný nedeterministický konečný automat (NKA) $M = (Q, \Sigma, \delta, q_0, F)$, ktorého stavy sú $Q = \{q_0, q_1, q_2, q_3\}$, vstupná abeceda $\Sigma = \{a, b\}$, q_0 je počiatočný stav, akceptačné stavy sú $F = \{q_1, q_2\}$ a prechodová funkcia je daná tabuľkou:

δ	a	b	ε
q_0	$\{q_2\}$	$\emptyset$	$\{q_1, q_3\}$
q_1	$\{q_1\}$	$\emptyset$	$\{q_3\}$
q_2	$\emptyset$	$\{q_2, q_3\}$	$\emptyset$
q_3	$\{q_0, q_1\}$	$\emptyset$	$\emptyset$

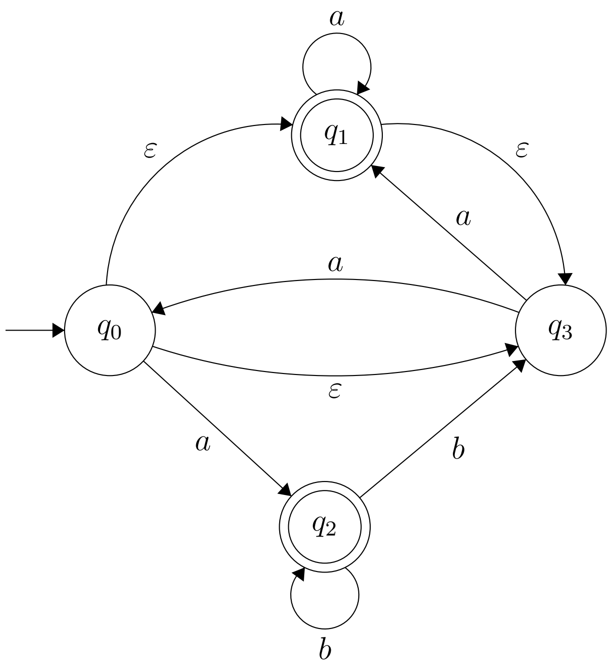
1. Nakreslite grafickú reprezentáciu daného NKA pomocou prechodového diagramu.
2. Zistite, či uvedený NKA akceptuje reťazce: ε, a, b, aba.

Riešenie:

1. Prechodový diagram reprezentujúci NKA zostrojíme rovnako ako pre DKA:
   - Každý vrchol tohto grafu predstavuje jeden stav NKA.

- Ak je v NKA možný prechod zo stavu q_i do stavov $\{q_{j_1}, q_{j_2}, ..., q_{j_i}\}$ na symbol c alebo prázdny ret'azec ε, t. j. v prechodovej funkcii $\delta(q_i, c) = \{q_{j_1}, q_{j_2}, ..., q_{j_i}\}$, resp. $\delta(q_i, \varepsilon) = \{q_{j_1}, q_{j_2}, ..., q_{j_i}\}$, potom v grafe vedú orientované hrany z vrcholu q_i do vrcholov $\{q_{j_1}, q_{j_2}, ..., q_{j_i}\}$ ohodnotené symbolom c, resp. ε.
- Do vrcholu predstavujúceho počiatočný stav vedie neohodnotená hrana.
- Vrcholy predstavujúce akceptačné stavy sú označené dvojitou kružnicou.

Prechodový diagram daného NKA je uvedený na obrázku 2.6.



Obr. 2.6: Prechodový diagram NKA z úlohy 2.2.1

2. Výpočty daného NKA pre jednotlivé ret'azce:

- Výpočet NKA zapisujeme podobne ako výpočet DKA, pomocou konfigurácií. Nedeterminizmus NKA sa prejavuje dvomi spôsobmi:
  - Prechodová funkcia obsahuje tzv. ε-prechody. V takom prípade sa NKA môže prepnút' do príslušného nasledovného stavu bez čítania vstupného symbolu.
  - Prechodová funkcia obsahuje prechody na ten istý vstupný symbol (prípadne ε) do viacerých nasledujúcich stavov. V takom prípade sa môže NKA po spracovaní vstupného symbolu (prípadne jeho ignorovaní ak ide o ε-prechod), nachádzat' v l'ubovol'nom z týchto stavov.
  - Dôsledkom nedeterminizmu je, že pre ten istý vstupný ret'azec môže existovat' viacero vetiev výpočtu.

- Ak NKA **úspešne prečítal celý reťazec**, teda na vstupe zostal už len prázdny reťazec ε, a zároveň NKA **skončil** v jednom z **akceptačných** stavov, hovoríme, že NKA **akceptuje** vstupný reťazec.
- Keďže v prípade NKA môže existovať viacero výpočtov pre ten istý vstup, NKA akceptuje reťazec vtedy, ak **existuje aspoň jeden** akceptačný výpočet.
- V prípade, že pre daný reťazec **neexistuje** akceptačný výpočet, t. j. alebo automat vždy reťazec celý spracuje a skončí v nejakom neakceptačnom stave, alebo sa zasekne pred jeho úplným spracovaním, tak hovoríme, že NKA vstupný reťazec **neakceptuje**.
- Reťazec ε: Pre tento reťazec existuje viacero výpočtov:
  - (q_0, ε), t. j. výpočet by skončil v počiatočnom stave q_0, ktorý je neakceptačný a vstup bol celý spracovaný.
  - $(q_0, \varepsilon) \vdash (q_3, \varepsilon)$, t. j. výpočet by skončil v neakceptačnom stave q_3, vstup bol celý spracovaný. Prechod zo stavu q_0 do stavu q_3 sme uskutočnili vďaka príslušnému ε-prechodu v NKA.
  - $(q_0, \varepsilon) \vdash (q_1, \varepsilon)$, t. j. výpočet by skončil v **akceptačnom** stave q_1, vstup bol celý spracovaný.

  Pre reťazec ε sme v NKA našli výpočet, ktorý tento reťazec akceptuje. Reťazec ε by teda uvedený NKA **akceptoval**, $\varepsilon \in L(M)$.
- Reťazec a: Pre tento reťazec existuje viacero výpočtov, napríklad:
  - $(q_0, a) \vdash (q_2, \varepsilon)$, t. j. výpočet by skončil v **akceptačnom** stave q_2, vstup bol celý spracovaný. Tento výpočet je teda **akceptačným výpočtom** reťazca a.
  - $(q_0, a) \vdash (q_1, a) \vdash (q_1, \varepsilon)$, t. j. výpočet by skončil v **akceptačnom** stave q_1, vstup bol celý spracovaný. Tento výpočet je teda **ďalším akceptačným výpočtom** reťazca a. Všimnite si, že prvý krok výpočtu $(q_0, a) \vdash (q_1, a)$ sme uskutočnili pomocou ε-prechodu zo stavu q_0 do stavu q_1, teda vstupný symbol a zostal po tomto kroku výpočtu nespracovaný na vstupe a spracoval sa až v nasledovnom kroku.
  - $(q_0, a) \vdash (q_1, a) \vdash (q_1, \varepsilon) \vdash (q_3, \varepsilon)$, t. j. výpočet by skončil v **neakceptačnom** stave q_3, vstup bol celý spracovaný. Tento výpočet je teda **neakceptačným výpočtom** reťazca a. Všimnite si, že posledný krok výpočtu $(q_1, \varepsilon) \vdash (q_3, \varepsilon)$ sme uskutočnili pomocou ε-prechodu zo stavu q_1 do stavu q_3.

  Pre reťazec a sme v NKA našli výpočet, ktorý tento reťazec akceptuje — dokonca prvé 2 uvedené výpočty sú akceptačné. Pre akceptáciu reťazca stačí, aby existoval jeden akceptačný výpočet, teda reťazec a by uvedený NKA **akceptoval**, $a \in L(M)$.
- Reťazec b: Aby sme zistili, či NKA akceptuje reťazec b, hľadáme akceptačný výpočet:

  - $(q_0, b) \vdash (q_1, b) \vdash (q_3, b)$. Zo stavu q_3 už neexistuje ďalší krok výpočtu pre symbol b na vstupe. Automat sa teda v tejto výpočtovej vetve zasekol v stave q_3 a na vstupe zostala neprečítaná časť vstupu. Tento výpočet teda nie je akceptačný.
  - $(q_0, b) \vdash (q_3, b)$. Formálne ide o iný výpočet ako ten predchádzajúci, ale znovu sa NKA zasekol v stave q_3 a nespracoval celý vstup. Ani tento výpočet nie je akceptačný.
  - Iné výpočty pre reťazec neexistujú.

  Vidíme, že pre reťazec b sme **nenašli** ani jeden akceptačný výpočet. V tomto NKA teda **neexistuje spôsob**, ako akceptovať reťazec b, NKA teda reťazec b neakceptuje, $b \notin L(M)$.

- Reťazec aba:
  - $(q_0, aba) \vdash (q_2, ba) \vdash (q_2, a)$. V stave q_2 sa výpočet zasekne, pretože nie je v tomto stave možné ani spracovať symbol a, ani sa prepnúť do iného stavu bez jeho spracovania, pretože zo stavu q_2 nevedú žiadne ε-prechody. Tento výpočet teda nie je akceptačný.
  - $(q_0, aba) \vdash (q_2, ba) \vdash (q_3, a) \vdash (q_1, \varepsilon)$. Keďže vstup sme celý spracovali a q_1 patrí medzi akceptačné stavy, tento výpočet je akceptačným.

  Vidíme, že pre reťazec aba sme **našli** aspoň jeden akceptačný výpočet. Tento NKA teda akceptuje reťazec aba, $aba \in L(M)$.

Úloha č. 2.2.2 Je daný jazyk $L = \{w \in \{0,1\}^* \mid \text{ tretí symbol od konca } w \text{ je nula}\}$ nad abecedou $A = \{0, 1\}$. Nájdite nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$, ktorý akceptuje jazyk L.

Riešenie: Pri konštrukcii NKA akceptujúceho nejaký jazyk L postupujeme rovnako, ako pri konštrukcii DKA:

1. Každý reťazec, ktorý bude automat M akceptovať, musí byť zároveň reťazcom patriacim do jazyka L, teda musí platiť $L(M) \subseteq L$.

2. Každý reťazec z jazyka L musí mať v automate M akceptačný výpočet, teda musí platiť $L \subseteq L(M)$.

Ak sú tieto 2 podmienky splnené, potom platí $L(M) = L$, teda jazyk $L(M)$ akceptovaný automatom M je totožný s jazykom L.

Znovu si na začiatok vymenujeme aspoň najkratšie reťazce patriace do jazyka L, aby sme videli, aké reťazce vlastne potrebujeme akceptovať. Do zadaného jazyka L patria všetky reťazce, ktorých tretí symbol od konca je nula, teda napríklad:

- $000, 001, 010, 011$ — to sú najkratšie reťazce, u ktorých má zmysel uvažovať, či ich tretí symbol od konca je nula alebo nie.

- $0000, 0001, 0010, 0011, 1000, 1001, 1010, 1011$ atď.

Ide teda o reťazce, ktoré je možné alternatívne popísať ako jazyk $L = \{w0\{00, 01, 10, 11\} \mid w \in \{0, 1\}^*\}$, čiže ako množinu reťazcov, ktoré pozostávajú zo zreťazenia:

- ľubovoľného reťazca z núl a jednotiek w,
- nuly,
- ľubovoľného reťazca z množiny $\{00, 01, 10, 11\}$.

NKA zostrojíme podľa nasledovnej logiky:

- Počiatočný stav q_0 bude slúžiť na 2 účely:
  - Počiatočný stav q_0 bude slúžiť na spracovanie prefixu w slov z jazyka L, teda tej postupnosti núl a jednotiek, ktorá predchádza nule, ktorá sa nachádza na treťom mieste sprava. To znamená, že ak v tomto stave NKA číta 0/1, ostáva v stave q_0, pretože bude predpokladať, že ide o symboly z prefixu w.
  - Zároveň, ak NKA v stave q_0 číta nulu, môže ísť práve o hľadanú nulu, ktorá sa nachádza na treťom mieste sprava. Preto zároveň zo stavu q_0 bude automat môcť prejsť do ďalšieho stavu q_1, ktorý bude označovať situáciu, že sme práve na vstupe našli hľadanú nulu, ktorá je tretím symbolom sprava.
  - Keďže dopredu nevieme, ktorá situácia nastala, teda či aktuálne čítaná nula predstavuje súčasť slova w alebo nulu, ktorá je tretia sprava, práve nedeterministické správanie NKA nám zaručí, že určite bude existovať akceptačný výpočet, teda, že jeden z výpočtov bude správny, pretože v rámci neho sa NKA správne rozhodne pre tretiu nulu sprava pre prechod do stavu q_1.
- Teda prechody zo stavu $q_0 : \delta(q_0, 0) = \{q_0, q_1\}, \delta(q_0, 1) = \{q_0\}$.
- Stav q_1 :
  - Ak sa NKA nachádza v stave q_1, predpokladáme, že sme práve na vstupe prečítali symbol nula, ktorý predstavuje tretí symbol vstupu sprava a na vstupe sa už len nachádza dvojica symbolov $\{00, 01, 10, 11\}$. Preto v stave q_1 čítaním jedného symbolu, 0 alebo 1, prejdeme do stavu q_2, ktorý bude reprezentovať situáciu, že nám zostáva už len jeden symbol na vstupe, 0 alebo 1.
- Teda prechody zo stavu $q_1 : \delta(q_1, 0) = \{q_2\}, \delta(q_1, 1) = \{q_2\}$.
- Stav q_2 :
  - Ak sa NKA nachádza v stave q_2, predpokladáme, že máme na vstupe už len jeden symbol, 0 alebo 1. Preto v stave q_2 čítaním symbolu 0 alebo 1 prejdeme do stavu q_f, ktorý bude reprezentovať situáciu, že sme práve dočítali slovo v tvare $w0\{00, 01, 10, 11\}$, teda slovo z jazyka L.

- Teda prechody zo stavu $q_2 : \delta(q_2, 0) = \{q_f\}, \delta(q_2, 1) = \{q_f\}$.
- Stav q_f :
  - Ak sa NKA nachádza v stave q_f, predpokladáme, že sme práve na vstupe rozpoznali ret'azec, ktorý obsahoval nulu ako tretí symbol sprava, teda slovo z jazyka L. Preto zo stavu q_f už nevedú žiadne prechody, pretože ak bolo na vstupe slovo z jazyka L, tak v momente, keď sa NKA dostal do stavu q_f, musel byt' už vstup celý prečítaný.
  - Stav q_f bude teda zároveň akceptačným stavom.

Dostávame teda nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$ so stavmi $Q = \{q_0, q_1, q_2, q_f\}$, vstupnou abecedou $\Sigma = \{0, 1\}$, počiatočným stavom q_0, množinou akceptačných stavov $F = \{q_f\}$ a prechodová funkcia δ je daná tabuľkou 2.4.

δ	0	1	ε
q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\emptyset$
q_1	$\{q_2\}$	$\{q_2\}$	$\emptyset$
q_2	$\{q_f\}$	$\{q_f\}$	$\emptyset$
q_f	$\emptyset$	$\emptyset$	$\emptyset$

Tabuľka 2.4: Prechodová funkcia NKA z úlohy 2.2.2

Skontrolujme, aspoň neformálne, či sú splnené nasledovné podmienky:

1. Platí $L(M) \subseteq L$?
   - Zist'ujeme, či každý ret'azec, ktorý náš automat akceptuje, spĺňa zároveň podmienku jazyka L.
   - Aby nami zostrojený automat akceptoval vstupný ret'azec, musí existovat' akceptačný výpočet, t. j. musí počas neho dôjst' k prechodu zo stavu q_0 do stavu q_f. K takémuto prechodu môže dôjst' len cez stavy q_1 a q_2. Z konštrukcie NKA je zrejmé, že ak sa automat dostane do stavu q_f, tak posledné 3 symboly vstupného slova museli tvorit' ret'azec z množiny $\{000, 001, 010, 011\}$, teda na vstupe bol ret'azec,v ktorom bol tretí symbol od konca 0. Teda každé slovo, ktorý automat akceptuje, zároveň patrí do jazyka L a platí $L(M) \subseteq L$.
2. Platí $L \subseteq L(M)$?
   - Zist'ujeme, či každý ret'azec, ktorý patrí do jazyka L, má zároveň v automate akceptačný výpočet.
   - Uvažujme l'ubovoľný ret'azec patriaci do jazyka L, t. j. ret'azec tvaru $w0\{00, 01, 10, 11\}$, $w \in \{0, 1\}^*$.

- Predpokladajme, že ide o reťazec tvaru $w000$. Keďže NKA sa môže počas jeho spracovania v stave q_0 rozhodnúť, či pri čítaní nuly prejde do stavu q_0 alebo q_1, určite existuje nasledovný výpočet:

$$(q_0, w000) \vdash^* (q_0, 000) \vdash (q_1, 00) \vdash (q_2, 0) \vdash (q_f, \varepsilon)$$

- Podobne sa dá ukázať, že existuje výpočet aj pre vstupy v tvare $w001, w010, w011$.
- Teda vidíme, že pre ľubovoľné slovo z jazyka L existuje v automate akceptačný výpočet a teda $L \subseteq L(M)$.

Keďže $L(M) \subseteq L$ a zároveň $L \subseteq L(M)$, tak určite platí $L(M) = L$, čo znamená, že jazyk $L(M)$ akceptovaný automatom M je totožný s jazykom L, a teda je náš automat správny.

Úloha č. 2.2.3 Sú dané nasledovné jazyky nad príslušnými abecedami. Nájdite nedeterministické konečné automaty, ktoré akceptujú príslušné jazyky.

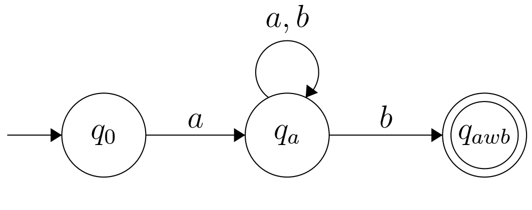
1. $L_1 = \{awb \mid w \in \{a, b\}^*\}$ nad abecedou $A = \{a, b\}$.
2. $L_2 = \{aw \mid w \in \{a, b\}^* \wedge |w| \equiv 0 \bmod 2\} \cup \{aw \mid w \in \{a, b\}^* \wedge |w| \leq 3\}$ nad abecedou $A = \{a, b\}$.
3. $L_3 = \{w \in \{0, 1, 2\}^* \mid$ posledný symbol reťazca w sa v ňom už vyskytol aj skôr$\}$ nad abecedou $A = \{0, 1, 2\}$.

Riešenie:

1. Jazyk L_1 je množina reťazcov nad abecedou $\{a, b\}$, ktoré začínajú symbolom a a končia symbolom b. Pre takýto jazyk je pomerne jednoduché zostrojiť aj DKA, aj NKA. Výhodou NKA vo všeobecnosti je, že je ich často jednoduchšie navrhnúť, než navrhnúť ekvivalentný deterministický konečný automat. Konkrétne v tomto jazyku je potrebné rozhodnúť, či aktuálne čítaný symbol b je posledným symbolom vstupu, alebo je súčasťou podreťazca w. Ak sa zostrojí NKA, je toto rozhodnutie možné urobiť nedeterministicky, čím sa zjednoduší fáza návrhu automatu. Riešením by mohol byť napríklad nedeterministický konečný automat uvedený na obrázku 2.7.

   NKA bol zostrojený nasledovným spôsobom:

   - Keďže prvý symbol reťazca, ktorý chceme akceptovať je a, z počiatočného stavu vedie prechod na symbol a do stavu q_a, ktorý predstavuje situáciu, že vstupný reťazec začínal symbolom a, a teda je adeptom na akceptáciu.
   - Ak vstupný reťazec začína symbolom b, výpočet sa zasekne v počiatočnom stave q_0, čo korešponduje so situáciou, že takéto reťazce by automat akceptovať nemal, pretože nepatria do jazyka L_1.



Obr. 2.7: Prechodový diagram NKA pre jazyk L_1 z úlohy 2.2.3

- V stave q_a sú slučky pre symboly a a b, ktoré sa použijú na spracovanie podreťazca w.
- Zo stavu q_a vedie prechod na symbol b do stavu q_{awb}. Ak sa tento prechod použije pri spracovaní posledného symbolu vstupného reťazca, potom vstupný reťazec bol určite v tvare awb, kde $w \in \{a, b\}^*$.
- Preto je stav q_{awb} akceptačným. Zo stavu navyše nevychádzajú žiadne prechody, pretože predpokladáme, že do tohto stavu sa výpočet dostane čítaním symbolu b v stave q_a, ktorý bol posledným symbolom vstupu.
- Je zrejmé, že v stave q_a je nedeterminizmus vzhľadom na symbol b, teda NKA si môže vybrať z 2 situácií:
  - Ostať v stave q_a — táto situácia je žiadúca, ak čítaný symbol b nie je posledným symbolom vstupu, ale súčasťou podreťazca w.
  - Prejsť do stavu q_{awb} — táto situácia je žiadúca, ak čítaný symbol b je posledným symbolom vstupu.
  - Keďže automat sa môže nedeterministicky rozhodnúť, do množiny všetkých vetiev výpočtu patrí aj situácia, že zostane v stave q_a, aj situácia, že prejde do stavu q_{awb}.
  - Teda určite bude existovať aj správny akceptačný výpočet, v rámci ktorého sa automat korektne rozhodne pre všetky symboly b, ktoré nie sú posledným symbolom vstupu zostať v stave q_a a až pre posledný symbol b prejsť do stavu q_{awb}.

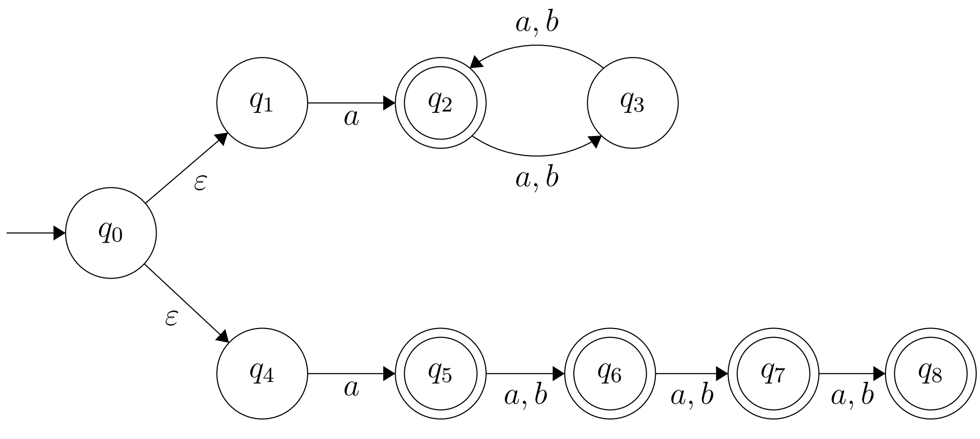
Pre úplnosť dodávame, že ide o nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$, ktorého množina stavov $Q = \{q_0, q_a, q_{awb}\}$, vstupnou abecedou je $\Sigma = \{a, b\}$, q_0 je počiatočný stav, množina akceptačných stavov $F = \{q_{awb}\}$ a prechodová funkcia δ je znázornená prechodovým diagramom na obrázku 2.7.

2. Jazyk L_2 je jazyk tvorený reťazcami zo symbolov $\{a, b\}$, ktorý obsahuje 2 typy reťazcov:

   - Reťazce začínajúce symbolom a, ktorých zvyšok za týmto symbolom má párnu dĺžku: $a, aaa, aab, aba, abb, aaaaa, aaaab, ...$, t. j. množina $\{aw \mid w \in \{a, b\}^* \wedge |w| \equiv 0 \text{ mod } 2\}$ nad abecedou $A = \{a, b\}$.

- Ret'azce začínajúce symbolom a, ktorých zvyšok za týmto symbolom má dĺžku najviac 3: $a, aa, ab, aaa, aab, aba, abb, aaaa, aaab, aaba, aabb, abaa, abab, abba, abbb$, t. j. množina $\{aw \mid w \in \{a,b\}^* \wedge |w| \leq 3\}$ nad abecedou $A = \{a, b\}$.

Príkladom takéhoto nedeterministického konečného automatu je automat $M = (Q, \Sigma, \delta, q_0, F)$ so stavmi $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8\}$, vstupnými symbolmi $\Sigma = \{a, b\}$, počiatočným stavom q_0, množinou akceptačných stavov $F = \{q_2, q_5, q_6, q_7, q_8\}$ a prechodovou funkciou δ danou prechodovým diagramom na obrázku 2.8:



Obr. 2.8: Prechodový diagram NKA pre jazyk L_2 z úlohy 2.2.3

Tento NKA je zostrojený podľa nasledovného princípu:

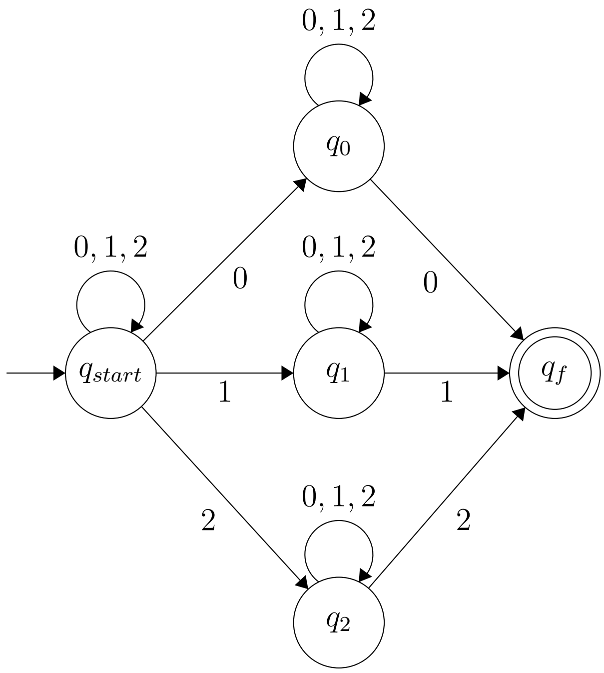
- V podstate je rozdelený na 2 menšie časti — časť vychádzajúcu zo stavu q_1, ktorej úlohou bude akceptácia reťazcov z množiny $\{aw \mid w \in \{a,b\}^* \wedge |w| \equiv 0 \bmod 2\}$ a časť vychádzajúcu zo stavu q_4, ktorej úlohou bude akceptácia reťazcov z množiny $\{aw \mid w \in \{a,b\}^* \wedge |w| \leq 3\}$.
- Na začiatku sa NKA nedeterministicky rozhodne pre jednu z týchto častí pomocou ε-prechodov a následne sa pokúsi spracovať vstupný reťazec.
- Ak je teda na vstupe reťazec, ktorý patrí do množiny $\{aw \mid w \in \{a,b\}^* \wedge |w| \equiv 0 \bmod 2\}$, tak existuje jeho akceptačný výpočet — v prípade, že sa NKA rozhodne prejsť do stavu q_1. Podobne, ak je na vstupe reťazec z množiny $\{aw \mid w \in \{a,b\}^* \wedge |w| \leq 3\}$, tak jeho akceptačný výpočet začína tak, že sa NKA rozhodne prepnúť do stavu q_4.
- Samotné menšie časti sú už potom de facto deterministické konečné automaty, ktoré v oboch prípadoch najprv prečítajú prvý symbol, ktorým musí byť a a následne skontrolujú:
  - Či je zvyšné slovo w párnej dĺžky (časť vychádzajúca zo stavu q_2),

– alebo či je zvyšné slovo w dĺžky najviac 3 (časť vychádzajúca zo stavu q_5).

3. Jazyk L_3 je jazyk tvorený reťazcami zo symbolov $\{0, 1, 2\}$, v ktorých sa posledný symbol vyskytuje v danom reťazci aspoň dvakrát.

   - Do jazyka L_3 patria reťazce: $00, 11, 22, 010, 100, 020, 200, 101, 111, 121, 200, 202, 0121$ atď., teda vo všetkých prípadoch sa posledný symbol vyskytuje v reťazci aj niekde skôr.
   - Do jazyka L_3 nepatria reťazce: $\varepsilon, 0, 1, 2, 01, 02, 12, 001, 002, 012, 01112$ atď., teda ide o reťazce, ktoré alebo nemajú posledný symbol (ε), alebo sa v nich posledný symbol vyskytuje len jedenkrát.

   Príkladom takéhoto nedeterministického konečného automatu je automat $M = (Q, \Sigma, \delta, q_{start}, F)$ so stavmi $Q = \{q_{start}, q_0, q_1, q_2, q_f\}$, vstupnými symbolmi $\Sigma = \{0, 1, 2\}$, počiatočným stavom q_{start}, množinou akceptačných stavov $F = \{q_f\}$ a prechodovou funkciou δ znázornenou prechodovým diagramom na obrázku 2.9.



Obr. 2.9: Prechodový diagram NKA pre jazyk L_3 z úlohy 2.2.3

Tento NKA je zostrojený podľa nasledovného princípu:

- Reťazce, pre ktoré platí, že ich posledný symbol sa v nich vyskytuje minimálne dvakrát, sa dajú rozdeliť do 3 skupín podľa posledného symbolu:

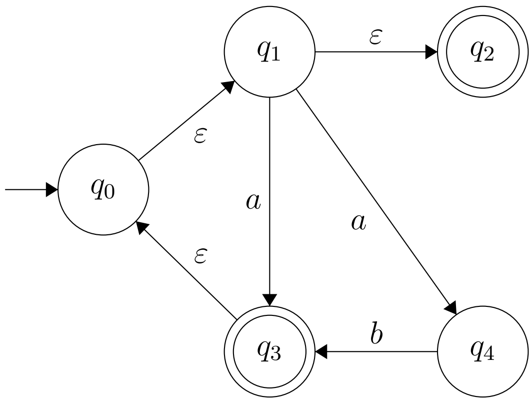
- $x0y0,\ x, y \in \{0, 1, 2\}^*$
- $x1y1,\ x, y \in \{0, 1, 2\}^*$
- $x2y2,\ x, y \in \{0, 1, 2\}^*$

- Uvedený zápis odzrkadľuje fakt, že posledný symbol sa musí niekde v reťazci nachádzať ešte minimálne jedenkrát, pričom zvyšné časti reťazca (označené x, y) sú v princípe ľubovoľné postupnosti symbolov $\{0, 1, 2\}$.
- Ak si uvedomíme vyššie uvedenú skutočnosť, je pomerne jednoduché navrhnúť stavy a činnosť NKA.
- Stav q_{start} je počiatočným stavom, ktorý sa použije na spracovanie predpony x, t. j. v stave q_{start} je slučka pre čítanie symbolov $\{0, 1, 2\}$.
- Keďže posledný symbol sa pre reťazce z jazyka L_3 musí v reťazci nachádzať ešte raz, predpokladajme, že NKA narazí na tento opakovaný výskyt posledného symbolu. V prípade, že ide o symbol 0 sa automat prepne do stavu q_0, ak ide o symbol 1 do stavu q_1, ak ide o symbol 2 do stavu q_2.
- Ak sa NKA nachádza v stave q_0, znamená to, že predpokladá, že prečítaná 0 na vstupe bola opakovaním posledného symbolu, ktorým bude znovu nula. Preto je v stave q_0 slučka pre symboly $\{0, 1, 2\}$, ktorej úlohou je spracovať časť y a zároveň je v stave q_0 prechod do stavu q_f, ktorý sa použije v prípade čítania posledného symbolu vstupu, nuly.
- Analogicky pracujú stavy q_1, resp. q_2, ktoré predpokladajú, že symbol, ktorý je na konci vstupu je 1, resp. 2.
- Je zrejmé, že ak je na vstupe NKA reťazec, ktorý obsahuje posledný symbol aspoň dvakrát, tak v závislosti na tom, či je posledný symbol 0, 1 alebo 2 bude existovať akceptačný výpočet cez stav q_0, q_1, resp. q_2.
- Znovu sa teda využíva nedeterminizmus NKA, ktorý predpokladá, že NKA sa sám „rozhodne“, ktorú výpočtovú cestu má zvoliť. To je dané tým, že keďže NKA uvažuje všetky možné výpočtové vetvy, tak keďže jedna z nich je určite správna, tak potom bude existovať akceptačný výpočet pre slová z jazyka L_3.
- Automat zároveň určite nebude akceptovať reťazce, ktorých posledný symbol sa v nich vyskytuje iba jedenkrát, pretože pre takéto reťazce sa NKA dostane maximálne do stavov q_0, q_1, q_2 a nie až do stavu q_f.

2.3 Determinizácia nedeterministických konečných automatov

Úloha č. 2.3.1 Je daný nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$ s množinou stavov $Q = \{q_0, q_1, q_2, q_3, q_4\}$, vstupnými symbolmi $\Sigma = \{a, b\}$, počiatočným stavom q_0, množinou akceptačných stavov $F = \{q_2, q_3\}$ a prechodovou funkciou danou prechodovým diagramom na obrázku 2.10. Nájdite ε-uzáver $CLOSURE_\varepsilon$ pre množiny stavov:

1. $\{q_2\}$
2. $\{q_3\}$
3. $\{q_0\}$
4. $\{q_1, q_3\}$
5. $\{q_0, q_1, q_2\}$
6. $\{q_0, q_1, q_4\}$



Obr. 2.10: Prechodový diagram NKA z úlohy 2.3.1

Riešenie:

Pri hľadaní ε-uzáveru množiny stavov NKA vezmeme danú množinu a pridáme do nej všetky také stavy NKA, do ktorých sa vieme dostať zo stavov pôvodnej množiny bez čítania vstupných symbolov, t. j. len pomocou ε-prechodov.

1. $\{q_2\}$
   - Zo stavu q_2 sa nevieme dostať pomocou ε-prechodov do žiadneho iného stavu, preto je ε-uzáverom množiny $\{q_2\}$ znovu len množina $\{q_2\}$, $CLOSURE_\varepsilon(\{q_2\}) = \{q_2\}$.

2. $\{q_3\}$
   - Zo stavu q_3 sa vieme dostať pomocou ε-prechodov do stavu q_0, $(q_3, \varepsilon) \vdash (q_0, \varepsilon)$ teda do výsledku $CLOSURE_\varepsilon(\{q_3\})$ bude okrem q_3 určite patriť aj q_0.
   - Keďže stav q_0 patrí do $CLOSURE_\varepsilon(\{q_3\})$, tak aj tie stavy, do ktorých sa vieme dostať z q_0 pomocou ε-prechodov, konkrétne q_1, budú patriť do $CLOSURE_\varepsilon(\{q_3\})$, pretože $(q_3, \varepsilon) \vdash (q_0, \varepsilon) \vdash (q_1, \varepsilon)$.
   - Keďže stav q_1 patrí do $CLOSURE_\varepsilon(\{q_3\})$, tak aj tie stavy, do ktorých sa vieme dostať z q_1 pomocou ε-prechodov, konkrétne q_2, budú patriť do $CLOSURE_\varepsilon(\{q_3\})$, pretože $(q_3, \varepsilon) \vdash (q_0, \varepsilon) \vdash (q_1, \varepsilon) \vdash (q_2, \varepsilon)$.
   - Dostávame množinu stavov $\{q_3, q_0, q_1, q_2\}$, z ktorých sa pomocou ε-prechodov už vieme dostať znovu len do stavov z tejto množiny, t. j. dostávame množinu uzavretú na ε-prechody.
   - Výsledkom operácie $CLOSURE_\varepsilon$ pre množinu $\{q_3\}$ teda je $CLOSURE_\varepsilon(\{q_3\}) = \{q_0, q_1, q_2, q_3\}$.
3. $\{q_0\}$
   - $CLOSURE_\varepsilon(\{q_0\}) = \{q_0, q_1, q_2\}$.
4. $\{q_1, q_3\}$
   - Zo stavu q_1 sa pomocou ε-prechodov vieme dostať (priamo) do stavu q_2, teda do výslednej množiny budú určite patriť q_1, q_2.
   - Zo stavu q_3 sa pomocou ε-prechodov vieme dostať (priamo) do stavu q_0. Ďalej sa zo stavu q_3 vieme dostať do stavu q_1 (cez stav q_0) a do stavu q_2 (cez stavy q_0, q_1). Teda do výslednej množiny budú určite patriť q_3, q_0, q_1, q_2.
   - Výsledok: $CLOSURE_\varepsilon(\{q_1, q_3\}) = \{q_0, q_1, q_2, q_3\}$.
5. $\{q_0, q_1, q_2\}$
   - $CLOSURE_\varepsilon(\{q_0, q_1, q_2\}) = \{q_0, q_1, q_2\}$.
6. $\{q_0, q_1, q_4\}$
   - $CLOSURE_\varepsilon(\{q_0, q_1, q_4\}) = \{q_0, q_1, q_2, q_4\}$.

Úloha č. 2.3.2 Je daný nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$ s množinou stavov $Q = \{q_0, q_1, q_2, q_3, q_4\}$, vstupnými symbolmi $\Sigma = \{a, b\}$, počiatočným stavom q_0, množinou akceptačných stavov $F = \{q_2, q_3\}$ a prechodovou funkciou danou prechodovým diagramom na obrázku 2.10, teda NKA z úlohy č. 2.3.1. Nájdite k nemu ekvivalentný deterministický konečný automat $\acute{M} = (\acute{Q}, \Sigma, \acute{\delta}, \acute{q_0}, \acute{F})$.

Riešenie:

1. Hľadanie deterministického konečného automatu (DKA) ekvivalentného k nejakému nedeterministickému konečnému automatu (NKA) začíname hľadaním počiatočného stavu $\acute{q_0}$ v DKA. Tento stav predstavuje množinu všetkých stavov, v ktorých sa môže nachádzať NKA predtým, ako prečíta prvý vstupný symbol vstupného reťazca, t. j. počiatočný stav v DKA nájdeme ako ε-uzáver počiatočného stavu q_0 NKA:

$$\acute{q_0} = CLOSURE_\varepsilon(\{q_0\}) = \{q_0, q_1, q_2\}$$

   Množina $\{q_0, q_1, q_2\}$ teda tvorí počiatočný stav $\acute{q_0}$ DKA ekvivalentného k zadanému NKA.

2. Po získaní nového stavu DKA, v tomto prípade $\{q_0, q_1, q_2\}$, následne musíme nájsť prechody z tohto stavu na všetky prípustné symboly vstupnej abecedy NKA, $\Sigma = \{a, b\}$, teda hľadáme hodnoty prechodovej funkcie v DKA: $\acute{\delta}(\{q_0, q_1, q_2\}, a)$ a $\acute{\delta}(\{q_0, q_1, q_2\}, b)$. Vo všeobecnosti, ak hľadáme DKA ekvivalentný k nejakému NKA a v DKA nám vznikne stav $\{q_1, q_2, ..., q_i\}$, potom stav, do ktorého prejde DKA zo stavu $\{q_1, q_2, ..., q_i\}$ na symbol a sa vypočíta podľa vzťahu:

$$\acute{\delta}(\{q_1, q_2, ..., q_i\}, a) = CLOSURE_\varepsilon\left(\bigcup_{k=1}^{i} \delta(q_k, a)\right)$$

   Prechod v DKA $\acute{\delta}(\{q_0, q_1, q_2\}, a)$:

   - Podľa predpisu najprv určíme, aká je hodnota $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a)$, teda množinu stavov, do ktorých sa potenciálne vie NKA dostať priamo pomocou symbolu a z ľubovoľného zo stavov $\{q_0, q_1, q_2\}$:
     - $\delta(q_0, a) = \emptyset$
     - $\delta(q_1, a) = \{q_3, q_4\}$
     - $\delta(q_2, a) = \emptyset$
     - $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) = \{q_3, q_4\}$
   - Následne ešte nájdeme ε-uzáver tejto množiny, pretože NKA sa môže teoreticky zo stavov $\{q_3, q_4\}$ prepnúť do ďalších stavov bez čítania vstupných symbolov pomocou ε-prechodov, teda stále ide o prechod z jedného z pôvodných stavov $\{q_0, q_1, q_2\}$ čítaním symbolu a na vstupe:
     - $CLOSURE_\varepsilon(\{q_3, q_4\}) = \{q_3, q_4, q_0, q_1, q_2\}$.
   - Teda v DKA bude prechod zo stavu $\{q_0, q_1, q_2\}$ na symbol a do stavu $\{q_0, q_1, q_2, q_3, q_4\}$:

$$\acute{\delta}(\{q_0, q_1, q_2\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$$

- Tento výsledok znamená, že ak sa NKA potenciálne nachádza v jednom zo stavov $\{q_0, q_1, q_2\}$, tak potenciálne sa čítaním vstupného symbolu a vie dostať do jedného zo stavov $\{q_0, q_1, q_2, q_3, q_4\}$.

Prechod v DKA $\acute{\delta}(\{q_0, q_1, q_2\}, b)$:

- Podľa predpisu najprv určíme, aká je hodnota $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b)$, teda množinu stavov, do ktorých sa potenciálne vie NKA dostať priamo pomocou symbolu b z ľubovoľného zo stavov $\{q_0, q_1, q_2\}$:
  - $\delta(q_0, b) = \emptyset$
  - $\delta(q_1, b) = \emptyset$
  - $\delta(q_2, b) = \emptyset$
  - $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) = \emptyset$
- Vidíme, že ak by sa NKA potenciálne nachádzal v jednom zo stavov $\{q_0, q_1, q_2\}$, tak čítaním symbolu b sa nevie dostať do žiadneho stavu, t. j. s istotou by sa NKA zasekol.
- V takom prípade nemusíme hľadať ε-uzáver množiny $\emptyset$, avšak ak by sme ho hľadali, tak by sme dostali znovu len prázdnu množinu:
  - $CLOSURE_\varepsilon(\emptyset) = \emptyset$.
- Teda v DKA bude prechod zo stavu $\{q_0, q_1, q_2\}$ na symbol b do stavu označeného ako $\emptyset$:
$$\acute{\delta}(\{q_0, q_1, q_2\}, b) = \emptyset$$
- Tento výsledok znamená, že ak sa NKA potenciálne nachádza v jednom zo stavov $\{q_0, q_1, q_2\}$, tak pri čítaní symbolu b sa zasekne a nevie sa dostať do žiadneho stavu.

3. Zároveň vidíme, že sme zistili, že v DKA sa budú okrem počiatočného stavu $\{q_0, q_1, q_2\}$ nachádzať aj stavy označené ako $\{q_0, q_1, q_2, q_3, q_4\}$ a $\emptyset$. Preto následne musíme vyšetriť prechody z týchto stavov na vstupné symboly a a b.

   Prechod v DKA $\acute{\delta}(\{q_0, q_1, q_2, q_3, q_4\}, a)$:

   - Najprv určíme, aká je hodnota $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a)$, teda množinu stavov, do ktorých sa potenciálne vie NKA dostať priamo pomocou symbolu a z ľubovoľného zo stavov $\{q_0, q_1, q_2, q_3, q_4\}$:
     - $\delta(q_0, a) = \emptyset$
     - $\delta(q_1, a) = \{q_3, q_4\}$
     - $\delta(q_2, a) = \emptyset$
     - $\delta(q_3, a) = \emptyset$
     - $\delta(q_4, a) = \emptyset$
     - $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a) = \{q_3, q_4\}$

- Následne ešte nájdeme ε-uzáver tejto množiny:
  - $CLOSURE_\varepsilon(\{q_3, q_4\}) = \{q_3, q_4, q_0, q_1, q_2\}$.
- Teda v DKA bude prechod zo stavu $\{q_0, q_1, q_2, q_3, q_4\}$ na symbol a do stavu $\{q_0, q_1, q_2, q_3, q_4\}$, t. j. v tomto stave bude „slučka“ na symbol a:

$$\acute{\delta}(\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$$

Prechod v DKA $\acute{\delta}(\{q_0, q_1, q_2, q_3, q_4\}, b)$:

- Najprv určíme, aká je hodnota $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) \cup \delta(q_4, b)$, teda množinu stavov, do ktorých sa potenciálne vie NKA dostať priamo pomocou symbolu b z ľubovoľného zo stavov $\{q_0, q_1, q_2, q_3, q_4\}$:
  - $\delta(q_0, b) = \emptyset$
  - $\delta(q_1, b) = \emptyset$
  - $\delta(q_2, b) = \emptyset$
  - $\delta(q_3, b) = \emptyset$
  - $\delta(q_4, b) = \{q_3\}$
  - $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) \cup \delta(q_4, b) = \{q_3\}$
- Následne ešte nájdeme ε-uzáver tejto množiny:
  - $CLOSURE_\varepsilon(\{q_3\}) = \{q_3, q_0, q_1, q_2\}$.
- Teda v DKA bude prechod zo stavu $\{q_0, q_1, q_2, q_3, q_4\}$ na symbol b do stavu $\{q_0, q_1, q_2, q_3\}$:

$$\acute{\delta}(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_0, q_1, q_2, q_3\}$$

4. Zistili sme, že v DKA bude v stave $\{q_0, q_1, q_2, q_3, q_4\}$ „slučka“ pre symbol a a prechod do stavu $\{q_0, q_1, q_2, q_3\}$ pre symbol b. Stav $\{q_0, q_1, q_2, q_3\}$ je zároveň novým stavom, ktorý sme doteraz v DKA nemali. Pokračujeme vo vyšetrovaní prechodov v DKA na vstupné symboly pre stavy, ktoré sme doteraz nevyšetrili, $\emptyset$ a $\{q_0, q_1, q_2, q_3\}$.

5. Prechody v DKA $\acute{\delta}(\emptyset, a)$, $\acute{\delta}(\emptyset, b)$:

   - Ak pri konštrukcii DKA nastane situácia, že nám vznikne stav označený prázdnou množinou, platí, že pre prázdnu množinu budú v DKA „slučky“ pre všetky vstupné symboly, t. j.:

$$\acute{\delta}(\emptyset, a) = \emptyset$$
$$\acute{\delta}(\emptyset, b) = \emptyset$$

6. Zostal nám nevyšetrený stav $\{q_0, q_1, q_2, q_3\}$. Vyšetrime najprv prechod na symbol a, $\acute{\delta}(\{q_0, q_1, q_2, q_3\}, a)$:

- $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a)$:
  - $\delta(q_0, a) = \emptyset$
  - $\delta(q_1, a) = \{q_3, q_4\}$
  - $\delta(q_2, a) = \emptyset$
  - $\delta(q_3, a) = \emptyset$
  - $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) = \{q_3, q_4\}$
- ε-uzáver tejto množiny:
  - $CLOSURE_\varepsilon(\{q_3, q_4\}) = \{q_3, q_4, q_0, q_1, q_2\}$.
- Teda v DKA bude prechod zo stavu $\{q_0, q_1, q_2, q_3\}$ na symbol a do stavu $\{q_0, q_1, q_2, q_3, q_4\}$:

$$\acute{\delta}(\{q_0, q_1, q_2, q_3\}, a) = \{q_0, q_1, q_2, q_3, q_4\}$$

Prechod na symbol b, $\acute{\delta}(\{q_0, q_1, q_2, q_3\}, b)$:

- $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b)$:
  - $\delta(q_0, b) = \emptyset$
  - $\delta(q_1, b) = \emptyset$
  - $\delta(q_2, b) = \emptyset$
  - $\delta(q_3, b) = \emptyset$
  - $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) = \emptyset$
- ε-uzáver tejto množiny:
  - $CLOSURE_\varepsilon(\emptyset) = \emptyset$.
- Teda v DKA bude prechod zo stavu $\{q_0, q_1, q_2, q_3\}$ na symbol b do stavu $\emptyset$:

$$\acute{\delta}(\{q_0, q_1, q_2, q_3\}, b) = \emptyset$$

7. Nedostali sme žiadny nový stav. Zistili sme teda, že výsledný DKA bude mať 4 stavy: $\acute{Q} = \{\{q_0, q_1, q_2\}, \{q_0, q_1, q_2, q_3, q_4\}, \emptyset, \{q_0, q_1, q_2, q_3\}\}$, pričom jeho počiatočným stavom bude $\acute{q_0} = \{q_0, q_1, q_2\}$ a prechodová funkcia tohto DKA je uvedená v tabuľke 2.5:

$\acute{\delta}$	a	b
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2, q_3, q_4\}$	$\emptyset$
$\{q_0, q_1, q_2, q_3, q_4\}$	$\{q_0, q_1, q_2, q_3, q_4\}$	$\{q_0, q_1, q_2, q_3\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3, q_4\}$	$\emptyset$

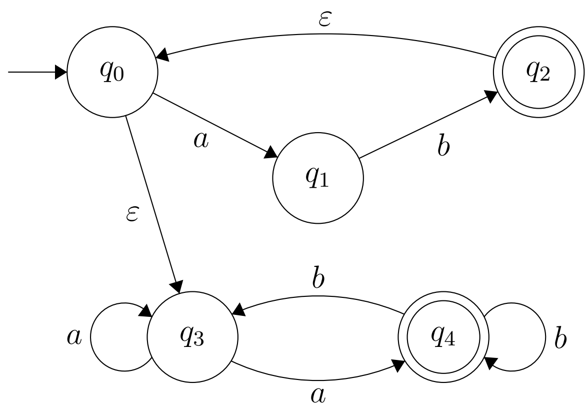
Tabuľka 2.5: Prechodová funkcia $\acute{\delta}$ DKA $\acute{M}$ zostrojeného v úlohe 2.3.2

8. Na záver nám zostáva určiť akceptačné stavy deterministického konečného automatu $\acute{M}$. Platí, že akceptačnými stavmi budú tie stavy, ktoré sú označené množinami, v ktorých sa nachádza aspoň jeden akceptačný stav pôvodného nedeterministického konečného automatu.

   - V našom prípade máme stavy:
     - $\{q_0, q_1, q_2\}$ — obsahuje akceptačný stav q_2 z NKA, teda stav $\{q_0, q_1, q_2\}$ bude akceptačným stavom DKA.
     - $\{q_0, q_1, q_2, q_3, q_4\}$ — obsahuje akceptačné stavy q_2, resp. q_3 z NKA, teda stav $\{q_0, q_1, q_2, q_3, q_4\}$ bude akceptačným stavom DKA.
     - $\emptyset$ — neobsahuje žiadne akceptačné stavy z NKA, teda stav $\emptyset$ bude neakceptačným stavom.
     - $\{q_0, q_1, q_2, q_3\}$ — obsahuje akceptačné stavy q_2, resp. q_3 z NKA, teda stav $\{q_0, q_1, q_2, q_3\}$ bude akceptačným stavom DKA.

   Výsledná množina akceptačných stavov DKA $\acute{F} = \{\{q_0, q_1, q_2\}, \{q_0, q_1, q_2, q_3, q_4\}, \{q_0, q_1, q_2, q_3\}\}$.

Úloha č. 2.3.3 Je daný nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$ s množinou stavov $Q = \{q_0, q_1, q_2, q_3, q_4\}$, vstupnou abecedou $\Sigma = \{a, b\}$, q_0 je počiatočný stav, množina akceptačných stavov $F = \{q_2, q_4\}$ a prechodová funkcia δ je daná prechodovým diagramom na obrázku 2.11. Nájdite ekvivalentný deterministický konečný automat $\acute{M}$.



Obr. 2.11: Prechodový diagram NKA z úlohy 2.3.3

Riešenie:

1. Počiatočný stav DKA $\acute{q_0} = CLOSURE(\{q_0\}) = \{q_0, q_3\}$.

   Tento stav nebude patriť medzi akceptačné stavy DKA, pretože príslušná množina $\{q_0, q_3\}$ neobsahuje ani 1 z akceptačných stavov pôvodného NKA.

Vyšetríme prechody zo stavu $\{q_0, q_3\}$ na jednotlivé symboly:

- $\acute{\delta}(\{q_0, q_3\}, a) = \{q_1, q_3, q_4\}$ pretože:
  - $\delta(q_0, a) = \{q_1\}$
  - $\delta(q_3, a) = \{q_3, q_4\}$
  - $\delta(q_0, a) \cup \delta(q_3, a) = \{q_1, q_3, q_4\}$
  - $CLOSURE_\varepsilon(\{q_1, q_3, q_4\}) = \{q_1, q_3, q_4\}$
- $\acute{\delta}(\{q_0, q_3\}, b) = \emptyset$ pretože:
  - $\delta(q_0, b) = \emptyset$
  - $\delta(q_3, b) = \emptyset$
  - $\delta(q_0, b) \cup \delta(q_3, b) = \emptyset$
  - $CLOSURE_\varepsilon(\emptyset) = \emptyset$
- Dostali sme teda 2 nové stavy: $\{q_1, q_3, q_4\}$ a $\emptyset$.

2. Stav DKA $\{q_1, q_3, q_4\}$.

   Tento stav bude patrit' medzi akceptačné stavy DKA, pretože príslušná množina $\{q_1, q_3, q_4\}$ obsahuje aspoň 1 z akceptačných stavov (q_4) pôvodného NKA.

   Vyšetríme prechody zo stavu $\{q_1, q_3, q_4\}$ na jednotlivé symboly:

   - $\acute{\delta}(\{q_1, q_3, q_4\}, a) = \{q_3, q_4\}$ pretože:
     - $\delta(q_1, a) = \emptyset$
     - $\delta(q_3, a) = \{q_3, q_4\}$
     - $\delta(q_4, a) = \emptyset$
     - $\delta(q_1, a) \cup \delta(q_3, a) \cup \delta(q_4, a) = \{q_3, q_4\}$
     - $CLOSURE_\varepsilon(\{q_3, q_4\}) = \{q_3, q_4\}$
   - $\acute{\delta}(\{q_1, q_3, q_4\}, b) = \{q_0, q_2, q_3, q_4\}$ pretože:
     - $\delta(q_1, b) = \{q_2\}$
     - $\delta(q_3, b) = \emptyset$
     - $\delta(q_4, b) = \{q_3, q_4\}$
     - $\delta(q_1, b) \cup \delta(q_3, b) \cup \delta(q_4, b) = \{q_2, q_3, q_4\}$
     - $CLOSURE_\varepsilon(\{q_2, q_3, q_4\}) = \{q_0, q_2, q_3, q_4\}$
   - Dostali sme teda 2 nové stavy: $\{q_3, q_4\}$ a $\{q_0, q_2, q_3, q_4\}$.

3. Stav DKA $\emptyset$.

   Tento stav nemôže patrit' medzi akceptačné stavy DKA, ked'že určite príslušná množina $\emptyset$ neobsahuje žiaden z akceptačných stavov pôvodného NKA.

   V stave $\emptyset$ máme vždy slučku na jednotlivé symboly:

- $\acute{\delta}(\emptyset, a) = \emptyset$
- $\acute{\delta}(\emptyset, b) = \emptyset$
- Tu sme nový stav nedostali.

4. Stav DKA $\{q_3, q_4\}$.

   Tento stav bude patriť medzi akceptačné stavy DKA, pretože príslušná množina $\{q_3, q_4\}$ obsahuje aspoň 1 z akceptačných stavov (q_4) pôvodného NKA.

   Vyšetríme prechody zo stavu $\{q_3, q_4\}$ na jednotlivé symboly:

   - $\acute{\delta}(\{q_3, q_4\}, a) = \{q_3, q_4\}$ pretože:
     - $\delta(q_3, a) = \{q_3, q_4\}$
     - $\delta(q_4, a) = \emptyset$
     - $\delta(q_3, a) \cup \delta(q_4, a) = \{q_3, q_4\}$
     - $CLOSURE_\varepsilon(\{q_3, q_4\}) = \{q_3, q_4\}$
   - $\acute{\delta}(\{q_3, q_4\}, b) = \{q_3, q_4\}$ pretože:
     - $\delta(q_3, b) = \emptyset$
     - $\delta(q_4, b) = \{q_3, q_4\}$
     - $\delta(q_3, b) \cup \delta(q_4, b) = \{q_3, q_4\}$
     - $CLOSURE_\varepsilon(\{q_3, q_4\}) = \{q_3, q_4\}$
   - Tu sme nový stav nedostali.

5. Stav DKA $\{q_0, q_2, q_3, q_4\}$.

   Tento stav bude patriť medzi akceptačné stavy DKA, pretože príslušná množina $\{q_0, q_2, q_3, q_4\}$ obsahuje aspoň 1 z akceptačných stavov (q_2, q_4) pôvodného NKA.

   Vyšetríme prechody zo stavu $\{q_0, q_2, q_3, q_4\}$ na jednotlivé symboly:

   - $\acute{\delta}(\{q_0, q_2, q_3, q_4\}, a) = \{q_1, q_3, q_4\}$ pretože:
     - $\delta(q_0, a) = \{q_1\}$
     - $\delta(q_2, a) = \emptyset$
     - $\delta(q_3, a) = \{q_3, q_4\}$
     - $\delta(q_4, a) = \emptyset$
     - $\delta(q_0, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a) = \{q_1, q_3, q_4\}$
     - $CLOSURE_\varepsilon(\{q_1, q_3, q_4\}) = \{q_1, q_3, q_4\}$
   - $\acute{\delta}(\{q_0, q_2, q_3, q_4\}, b) = \{q_3, q_4\}$ pretože:
     - $\delta(q_0, b) = \emptyset$
     - $\delta(q_2, b) = \emptyset$
     - $\delta(q_3, b) = \emptyset$

- $\delta(q_4, b) = \{q_3, q_4\}$
- $\delta(q_0, b) \cup \delta(q_2, b) \cup \delta(q_3, b) \cup \delta(q_4, b) = \{q_3, q_4\}$
- $CLOSURE_\varepsilon(\{q_3, q_4\}) = \{q_3, q_4\}$

• Tu sme nový stav nedostali.

6. Vyšetrili sme všetky stavy, ktoré nám počas determinizácie vznikli. Dostávame teda DKA $\acute{M}$, ktorý obsahuje 5 stavov, $\acute{Q} = \{\{q_0, q_3\}, \{q_1, q_3, q_4\}, \emptyset, \{q_3, q_4\}, \{q_0, q_2, q_3, q_4\}\}$, jeho počiatočným stavom je $\acute{q_0} = \{q_0, q_3\}$, akceptačné stavy sú $\acute{F} = \{\{q_1, q_3, q_4\}, \{q_3, q_4\}, \{q_0, q_2, q_3, q_4\}\}$ a jeho prechodová funkcia $\acute{\delta}$ je uvedená v tabuľke 2.6.

$\acute{\delta}$	a	b
$\{q_0, q_3\}$	$\{q_1, q_3, q_4\}$	$\emptyset$
$\{q_1, q_3, q_4\}$	$\{q_3, q_4\}$	$\{q_0, q_2, q_3, q_4\}$
$\emptyset$	$\emptyset$	$\emptyset$
$\{q_3, q_4\}$	$\{q_3, q_4\}$	$\{q_3, q_4\}$
$\{q_0, q_2, q_3, q_4\}$	$\{q_1, q_3, q_4\}$	$\{q_3, q_4\}$

Tabuľka 2.6: Prechodová funkcia $\acute{\delta}$ DKA $\acute{M}$ zostrojeného v úlohe 2.3.3

Úloha č. 2.3.4 Je daný nedeterministický konečný automat $M = (Q, \Sigma, \delta, q_0, F)$ s množinou stavov $Q = \{q_0, q_1, q_2, q_3, q_4\}$, vstupnou abecedou $\Sigma = \{a, b\}$, q_0 je počiatočný stav, množina akceptačných stavov $F = \{q_1, q_4\}$ a prechodová funkcia δ je daná tabuľkou 2.7. Nájdite ekvivalentný deterministický konečný automat $\acute{M}$.

δ	a	b	ε
q_0		$\{q_3\}$	$\{q_1\}$
q_1	$\{q_1, q_2\}$		
q_2		$\{q_2, q_4\}$	$\{q_0\}$
q_3	$\{q_1, q_2\}$		
q_4	$\{q_3\}$		

Tabuľka 2.7: Prechodová funkcia δ NKA z úlohy 2.3.4

Riešenie:

1. Počiatočný stav DKA $\acute{q_0} = CLOSURE(\{q_0\}) = \{q_0, q_1\}$.

   Počiatočný stav $\{q_0, q_1\}$ bude zároveň aj akceptačným stavom, keďže množina, ktorá ho tvorí, obsahuje aspoň 1 z akceptačných stavov, konkrétne q_1, pôvodného NKA.

   Vyšetríme prechody zo stavu $\{q_0, q_1\}$ na jednotlivé symboly:

- $\acute{\delta}(\{q_0, q_1\}, a) = \{q_0, q_1, q_2\}$ pretože:
  - $\delta(q_0, a) = \emptyset$
  - $\delta(q_1, a) = \{q_1, q_2\}$
  - $\delta(q_0, a) \cup \delta(q_1, a) = \{q_1, q_2\}$
  - $CLOSURE_\varepsilon(\{q_1, q_2\}) = \{q_0, q_1, q_2\}$
- $\acute{\delta}(\{q_0, q_1\}, b) = \{q_3\}$ pretože:
  - $\delta(q_0, b) = \{q_3\}$
  - $\delta(q_1, b) = \emptyset$
  - $\delta(q_0, b) \cup \delta(q_1, b) = \{q_3\}$
  - $CLOSURE_\varepsilon(\{q_3\}) = \{q_3\}$
- Dostali sme teda 2 nové stavy: $\{q_0, q_1, q_2\}$ a $\{q_3\}$.

2. Stav DKA $\{q_0, q_1, q_2\}$.

   Tento stav bude patriť medzi akceptačné stavy DKA, pretože príslušná množina $\{q_0, q_1, q_2\}$ obsahuje aspoň 1 z akceptačných stavov (q_1) pôvodného NKA.

   Vyšetríme prechody zo stavu $\{q_0, q_1, q_2\}$ na jednotlivé symboly:

   - $\acute{\delta}(\{q_0, q_1, q_2\}, a) = \{q_0, q_1, q_2\}$ pretože:
     - $\delta(q_0, a) = \emptyset$
     - $\delta(q_1, a) = \{q_1, q_2\}$
     - $\delta(q_2, a) = \emptyset$
     - $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) = \{q_1, q_2\}$
     - $CLOSURE_\varepsilon(\{q_1, q_2\}) = \{q_0, q_1, q_2\}$
   - $\acute{\delta}(\{q_0, q_1, q_2\}, b) = \{q_0, q_1, q_2, q_3, q_4\}$ pretože:
     - $\delta(q_0, b) = \{q_3\}$
     - $\delta(q_1, b) = \emptyset$
     - $\delta(q_2, b) = \{q_2, q_4\}$
     - $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) = \{q_2, q_3, q_4\}$
     - $CLOSURE_\varepsilon(\{q_2, q_3, q_4\}) = \{q_0, q_1, q_2, q_3, q_4\}$
   - Dostali sme teda 1 nový stav: $\{q_0, q_1, q_2, q_3, q_4\}$.

3. Stav DKA $\{q_3\}$.

   Tento stav nebude patriť medzi akceptačné stavy DKA, pretože príslušná množina $\{q_3\}$ neobsahuje ani 1 z akceptačných stavov pôvodného NKA.

   Vyšetríme prechody zo stavu $\{q_3\}$ na jednotlivé symboly:

   - $\acute{\delta}(\{q_3\}, a) = \{q_0, q_1, q_2\}$ pretože:

  - $\delta(q_3, a) = \{q_1, q_2\}$
  - $CLOSURE_\varepsilon(\{q_1, q_2\}) = \{q_0, q_1, q_2\}$
- $\acute{\delta}(\{q_3\}, b) = \emptyset$ pretože:
  - $\delta(q_3, b) = \emptyset$
  - $CLOSURE_\varepsilon(\emptyset) = \emptyset$
- Dostali sme teda 1 nový stav: $\emptyset$.

4. Stav DKA $\{q_0, q_1, q_2, q_3, q_4\}$.

   Tento stav bude patriť medzi akceptačné stavy DKA, pretože príslušná množina $\{q_0, q_1, q_2, q_3, q_4\}$ obsahuje aspoň 1 z akceptačných stavov (q_1, q_4) pôvodného NKA.

   Vyšetríme prechody zo stavu $\{q_0, q_1, q_2, q_3, q_4\}$ na jednotlivé symboly:

   - $\acute{\delta}(\{q_0, q_1, q_2, q_3, q_4\}, a) = \{q_0, q_1, q_2, q_3\}$ pretože:
     - $\delta(q_0, a) = \emptyset$
     - $\delta(q_1, a) = \{q_1, q_2\}$
     - $\delta(q_2, a) = \emptyset$
     - $\delta(q_3, a) = \{q_1, q_2\}$
     - $\delta(q_4, a) = \{q_3\}$
     - $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a) = \{q_1, q_2, q_3\}$
     - $CLOSURE_\varepsilon(\{q_1, q_2, q_3\}) = \{q_0, q_1, q_2, q_3\}$
   - $\acute{\delta}(\{q_0, q_1, q_2, q_3, q_4\}, b) = \{q_0, q_1, q_2, q_3, q_4\}$ pretože:
     - $\delta(q_0, b) = \{q_3\}$
     - $\delta(q_1, b) = \emptyset$
     - $\delta(q_2, b) = \{q_2, q_4\}$
     - $\delta(q_3, b) = \emptyset$
     - $\delta(q_4, b) = \emptyset$
     - $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) \cup \delta(q_4, b) = \{q_2, q_3, q_4\}$
     - $CLOSURE_\varepsilon(\{q_2, q_3, q_4\}) = \{q_0, q_1, q_2, q_3, q_4\}$
   - Dostali sme teda 1 nový stav: $\{q_0, q_1, q_2, q_3\}$.

5. Stav DKA $\emptyset$.

   Tento stav nemôže patriť medzi akceptačné stavy DKA, keďže určite príslušná množina $\emptyset$ neobsahuje žiaden z akceptačných stavov pôvodného NKA.

   V stave $\emptyset$ máme vždy slučku na jednotlivé symboly:

   - $\acute{\delta}(\emptyset, a) = \emptyset$

- $\acute{\delta}(\emptyset, b) = \emptyset$
- Tu sme nový stav nedostali.

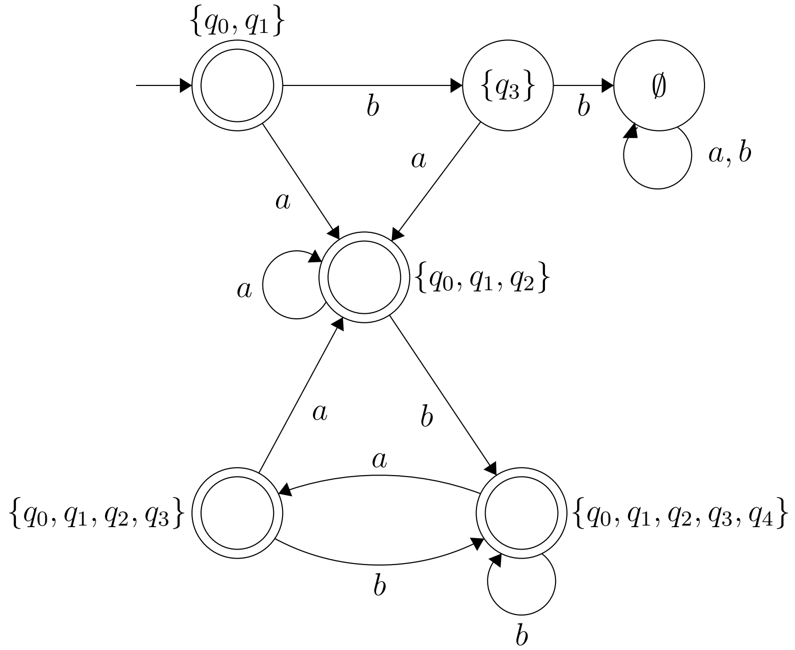
6. Stav DKA $\{q_0, q_1, q_2, q_3\}$.

   Tento stav bude patriť medzi akceptačné stavy DKA, pretože príslušná množina $\{q_0, q_1, q_2, q_3\}$ obsahuje aspoň 1 z akceptačných stavov (q_1) pôvodného NKA.

   Vyšetríme prechody zo stavu $\{q_0, q_1, q_2, q_3\}$ na jednotlivé symboly:

   - $\acute{\delta}(\{q_0, q_1, q_2, q_3\}, a) = \{q_0, q_1, q_2\}$ pretože:
     - $\delta(q_0, a) = \emptyset$
     - $\delta(q_1, a) = \{q_1, q_2\}$
     - $\delta(q_2, a) = \emptyset$
     - $\delta(q_3, a) = \{q_1, q_2\}$
     - $\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) = \{q_1, q_2\}$
     - $CLOSURE_\varepsilon(\{q_1, q_2\}) = \{q_0, q_1, q_2\}$
   - $\acute{\delta}(\{q_0, q_1, q_2, q_3\}, b) = \{q_0, q_1, q_2, q_3, q_4\}$ pretože:
     - $\delta(q_0, b) = \{q_3\}$
     - $\delta(q_1, b) = \emptyset$
     - $\delta(q_2, b) = \{q_2, q_4\}$
     - $\delta(q_3, b) = \emptyset$
     - $\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) = \{q_2, q_3, q_4\}$
     - $CLOSURE_\varepsilon(\{q_2, q_3, q_4\}) = \{q_0, q_1, q_2, q_3, q_4\}$
   - Nedostali sme žiadne nové stavy.

7. Vyšetrili sme všetky stavy, ktoré nám počas determinizácie vznikli. Dostávame teda DKA $\acute{M}$, ktorý obsahuje 6 stavov, $\acute{Q} = \{\{q_0, q_1\}, \{q_0, q_1, q_2\}, \{q_3\}, \{q_0, q_1, q_2, q_3, q_4\}, \emptyset, \{q_0, q_1, q_2, q_3\}\}$, jeho počiatočným stavom je $\acute{q_0} = \{q_0, q_1\}$, akceptačné stavy sú $\acute{F} = \{\{q_0, q_1\}, \{q_0, q_1, q_2\}, \{q_0, q_1, q_2, q_3, q_4\}, \{q_0, q_1, q_2, q_3\}\}$ a jeho prechodová funkcia $\acute{\delta}$ je daná prechodovým diagramom na obrázku 2.12.



Obr. 2.12: Prechodový diagram DKA $\acute{M}$ zostrojeného v úlohe 2.3.4

Kapitola 3

Regulárne výrazy

3.1 Popis jazykov regulárnymi výrazmi

Úloha č. 3.1.1 Slovne popíšte jazyky popísané uvedenými regulárnymi výrazmi:

1. $R_1 = 0(0 \mid 1)^*1$,
2. $R_2 = (((b \mid B)(e \mid E)(g \mid G)(i \mid I)(n \mid N)) \mid ((e \mid E)(n \mid N)(d \mid D)))$,
3. $R_3 = (a \mid b \mid ... \mid z)^*(nos \mid pin)(a \mid b \mid ... \mid z)^*$, kde $(a \mid b \mid ... \mid z)$ je skrátený zápis zjednotenia všetkých malých písmen anglickej abecedy,
4. $R_4 = ((ab \mid ba)^*(\varepsilon \mid c \mid d)(aba)(aba)^*)^*$,
5. $R_5 = ((\_ \mid a \mid ... \mid z \mid A \mid ... \mid Z)(\_ \mid a \mid ... \mid z \mid A \mid ... \mid Z \mid 0 \mid ... \mid 9)^*$, kde $(\_ \mid a \mid ... \mid z \mid A \mid ... \mid Z)$ je skrátený zápis zjednotenia malých/veľkých písmen anglickej abecedy a podčiarkovníka a $(\_ \mid a \mid ... \mid z \mid A \mid ... \mid Z \mid 0 \mid ... \mid 9)$ je skrátený zápis zjednotenia malých/veľkých písmen anglickej abecedy, podčiarkovníka a číslic.

Riešenie:

1. $R_1 = 0(0 \mid 1)^*1$:

   regulárny výraz popisuje reťazce nad abecedou $\{0, 1\}$, ktoré začínajú nulou a končia jednotkou.

2. $R_2 = (((b \mid B)(e \mid E)(g \mid G)(i \mid I)(n \mid N)) \mid ((e \mid E)(n \mid N)(d \mid D)))$:

   regulárny výraz popisuje reťazce, ktoré sú tvorené všetkými verziami slov *begin* a *end* z hľadiska veľkosti jednotlivých písmen, t. j. reťazce $\{begin, Begin, bEgin, beGin, begIn, begiN, end, End, eNd, enD, BEgin, ENd, ...\}$.

3. $R_3 = (a \mid b \mid ... \mid z)^*(nos \mid pin)(a \mid b \mid ... \mid z)^*$,

   regulárny výraz popisuje všetky ret'azce nad abecedou malých písmen anglickej abecedy, ktoré obsahujú *nos* alebo *pin* ako svoj podret'azec, t. j. napríklad $\{nos, podnos, nostradamus, pin, pinelka, camping, aminoskupina, ...\}$.

4. $R_4 = ((ab \mid ba)^*(\varepsilon \mid c \mid d)(aba)(aba)^*)^*$,

   regulárny výraz popisuje všetky ret'azce nad abecedou $\{a, b, c, d\}$, ktoré sú alebo prázdny ret'azec, alebo sa dajú rozdelit' na podret'azce, pričom každý podret'azec je v nasledovnom tvare:

   - na začiatku **môže** obsahovat' predponu tvorenú l'ubovol'nými kombináciami ret'azcov ab, ba,
   - za touto predponou sa **môže** nachádzat' symbol c alebo d,
   - na konci obsahuje príponu, ktorú tvorí **aspoň jedno** opakovanie ret'azca aba.

5. $R_5 = ((\_ \mid a \mid ... \mid z \mid A \mid ... \mid Z)(\_ \mid a \mid ... \mid z \mid A \mid ... \mid Z \mid 0 \mid ... \mid 9)^*$:

   regulárny výraz popisuje všetky **neprázdne** ret'azce zložené z malých/vel'kých písmen anglickej abecedy, číslic a podčiarkovníka, ktoré **nezačínajú** číslicou.

   Mimochodom, tento regulárny výraz popisuje platné identifikátory v programovacom jazyku C.

Úloha č. 3.1.2 Sú dané nasledovné jazyky nad príslušnými abecedami. Popíšte uvedené jazyky pomocou regulárnych výrazov:

1. $L_1 = \{w \in \{a, b, c\}^* \mid w$ obsahuje ako podret'azec aba a začína $c\}$ nad abecedou $A = \{a, b, c\}$.
2. $L_2 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv \sharp_b(w) \bmod 2\}$ nad abecedou $A = \{a, b\}$.
3. $L_3 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv 0 \bmod 3\}$ nad abecedou $A = \{a, b\}$.
4. $L_4 = \{w \in \{a, b\}^* \mid w$ má nepárny počet symbolov $b\}$ nad abecedou $A = \{a, b\}$.
5. $L_5 = \{a^*b^*\} \cup \{b^*a^*\}$ nad abecedou $A = \{a, b\}$.
6. $L_6 = \{a^*b^*\} \cap \{b^*a^*\}$ nad abecedou $A = \{a, b\}$.
7. $L_7 = \{b^*ab^*\} \cap \{a^*bab\}$ nad abecedou $A = \{a, b\}$.
8. $L_8 = \{w \in \{a, b\}^* \mid \sharp_a(w) < 3\}$ nad abecedou $A = \{a, b\}$.
9. $L_9 = \{w \in \{a, b\}^* \mid \sharp_a(w) < 3\}^C$ nad abecedou $A = \{a, b\}$.
10. $L_{10} = \{aw \mid w \in \{a, b\}^*\} \setminus \{wa \mid w \in \{a, b\}^*\}$ nad abecedou $A = \{a, b\}$.

Riešenie:

1. $L_1 = \{w \in \{a, b, c\}^* \mid w$ obsahuje ako podreťazec aba a začína $c\}$ nad abecedou $A = \{a, b, c\}$.

   $R_1 = c(a \mid b \mid c)^*(aba)(a \mid b \mid c)^*$

2. $L_2 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv \sharp_b(w) \bmod 2\}$ nad abecedou $A = \{a, b\}$.

   $R_2 = ((a \mid b)(a \mid b))^*$

   Do jazyka patria reťazce, ktoré majú súčasne alebo párny počet symbolov a a párny počet symbolov b, alebo súčasne nepárny počet symbolov a a nepárny počet symbolov b. Po malom zamyslení sa dá prísť na to, že takéto reťazce sú **práve tie reťazce**, ktoré sú **párnej dĺžky**. A túto vlastnosť vieme pomocou regulárneho výrazu zapísať jednoducho, ako potenciálne opakovanie sa dvojice symbolov $(a \mid b)(a \mid b)$, t. j. $((a \mid b)(a \mid b))^*$

3. $L_3 = \{w \in \{a, b\}^* \mid \sharp_a(w) \equiv 0 \bmod 3\}$ nad abecedou $A = \{a, b\}$.

   $R_3 = (b^*) \mid (b^*ab^*ab^*ab^*)^*$

   Do jazyka patria reťazce, ktoré majú počet symbolov a deliteľný tromi. Takéto reťazce sa teda dajú rozdeliť na 2 skupiny:

   - Reťazce zložené len zo symbolov b, tie popisuje regulárny výraz b^*.
   - Reťazce zložené z kombinácií takých podreťazcov, v ktorých sa nachádzajú 3 symboly a, ktoré môžu byť obklopené symbolmi b, t. j. podreťazce sú popísateľné regulárnym výrazom $b^*ab^*ab^*ab^*$ a ich ľubovoľné kombinácie výrazom $(b^*ab^*ab^*ab^*)^*$.

   Zjednotením vyššie uvedených teda dostávame výsledný regulárny výraz $(b^*) \mid (b^*ab^*ab^*ab^*)^*$.

   Alternatívne je tento jazyk možné popísať aj jednoduchším regulárnym výrazom: $R_3 = (b \mid ab^*ab^*a)^*$.

4. $L_4 = \{w \in \{a, b\}^* \mid w$ má nepárny počet symbolov $b\}$ nad abecedou $A = \{a, b\}$.

   $R_4 = (a^*ba^*)(a^*ba^*ba^*)^*$

   Do jazyka patria reťazce, ktoré majú nepárny počet symbolov b (1, 3, 5, 7, ...).

   - Reťazce určite obsahujú aspoň 1 symbol b, ktorý môže byť obklopený postupnosťou symbolov a. Nech tento symbol b je prvý symbol v reťazci, t. j. na začiatku reťazce z jazyka obsahujú predponu popísanú regulárnym výrazom a^*ba^*.

- Následne ret'azce môžu obsahovat' l'ubovol'né kombinácie podret'azcov tvorených dvojicou symbolov b pred ktorými/za ktorými/medzi ktorými môžu byt' postupnosti symbolov a, t. j. $(a^*ba^*ba^*)^*$.

5. $L_5 = \{a^*b^*\} \cup \{b^*a^*\}$ nad abecedou $A = \{a, b\}$.

   $R_5 = (a^*b^*) \mid (b^*a^*)$

   Ked'že už pôvodná špecifikácia jazyka obsahovala len operácie zjednotenia, zret'azenia a iterácie, jej prepis do regulárneho výrazu je pomerne priamočiary.

6. $L_6 = \{a^*b^*\} \cap \{b^*a^*\}$ nad abecedou $A = \{a, b\}$.

   $R_6 = a^* \mid b^*$

   Stačí si uvedomit', že jazyk je prienikom ret'azcov spĺňajúcich tvar a^*b^* alebo b^*a^*, teda musí íst' alebo o ret'azce typu a^*, alebo o ret'azce typu b^*.

7. $L_7 = \{b^*ab^*\} \cap \{a^*bab\}$ nad abecedou $A = \{a, b\}$.

   $R_7 = bab$

   Stačí si uvedomit', že uvedeným prienikom je jediný ret'azec, bab.

8. $L_8 = \{w \in \{a, b\}^* \mid \sharp_a(w) < 3\}$ nad abecedou $A = \{a, b\}$.

   $R_8 = b^* \mid b^*ab^* \mid b^*ab^*ab^*$

   Stačí si uvedomit', že sem patria 3 typy ret'azcov nad abecedou $A = \{a, b\}$:

   - Bez symbolov a, teda typu b^*.
   - S jedným výskytom symbolu a, teda typu b^*ab^*.
   - S dvomi výskytmi symbolu a, teda typu $b^*ab^*ab^*$.

   Alternatívne by sme napríklad mohli použit' aj regulárny výraz $b^*(\varepsilon \mid a \mid ab^*a)b^*$

9. $L_9 = \{w \in \{a, b\}^* \mid \sharp_a(w) < 3\}^C$ nad abecedou $A = \{a, b\}$.

   $R_9 = (b^*ab^*ab^*ab^*)(a \mid b)^*$

   Stačí si uvedomit', že sem patria také ret'azce nad abecedou $A = \{a, b\}$, ktoré majú aspoň 3 symboly a, pretože tento jazyk tvorí doplnok jazyka L_8:

   - Teda tieto ret'azce určite budú mat' ako prefix tri symboly a, pred ktorými/medzi ktorými/za ktorými sa nachádzajú l'ubovol'né postupnosti symbolov b, t. j. $(b^*ab^*ab^*ab^*)$
   - Za týmto prefixom môže byt' l'ubovol'ný ret'azec zložený zo symbolov $\{a, b\}$, t. j. $(a \mid b)^*$.

10. $L_{10} = \{aw \mid w \in \{a,b\}^*\} \setminus \{wa \mid w \in \{a,b\}^*\}$ nad abecedou $A = \{a,b\}$.

    $R_{10} = a(a \mid b)^*b$

    Stačí si uvedomiť, že sem patria tie reťazce začínajúce symbolom a nad abecedou $\{a,b\}$, ktoré zároveň nekončia symbolom a, keďže uvažujeme rozdiely týchto dvoch množín.

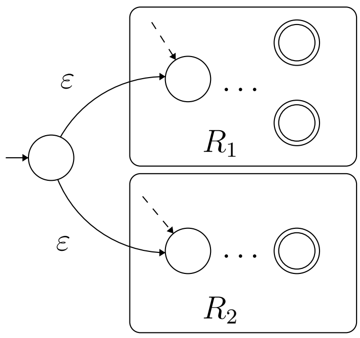
3.2 Konštrukcia nedeterministických konečných automatov ekvivalentných k regulárnym výrazom

Úloha č. 3.2.1 Nájdite nedeterministický konečný automat ekvivalentný k regulárnemu výrazu $R = (01 \mid 10)^*(\varepsilon \mid 0 \mid 1)$ nad abecedou $A = \{0,1\}$.

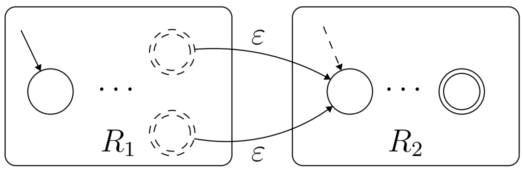
Riešenie: Pri hľadaní ekvivalentného NKA k zadanému regulárnemu výrazu hľadáme taký nedeterministický konečný automat M, pre ktorý platí, že akceptuje práve taký jazyk, ktorý popisuje regulárny výraz R.

Štandardne sa na riešenie tejto úlohy používa tzv. Thompsonova konštrukcia (nazývaná aj ako algoritmus McNaughton-Yamada-Thompson). Táto konštrukcia spočíva v tom, že sa zadaný regulárny výraz rozdelí na aplikácie príslušných operácií na menšie regulárne výrazy podľa uvedených schém†:

- zjednotenie 2 regulárnych výrazov R_1 a R_2, $R_1 \mid R_2$

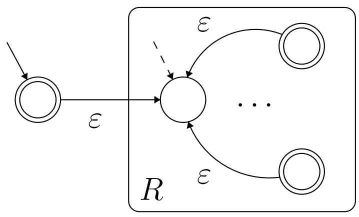


- zreťazenie 2 regulárnych výrazov R_1 a R_2, R_1R_2



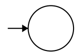
†Schémy sú prevzaté z [2].

- iterácia regulárneho výrazu R, R^*

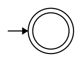


a postupne sa konštruujú NKA pre výsledky jednotlivých operácií, počnúc elementárnymi regulárnymi výrazmi, pre ktoré uvádzame aj príslušné elementárne NKA:

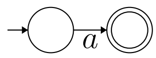
- prázdny jazyk, $R = \emptyset$,



- jazyk s prázdnym reťazcom, $R = \varepsilon$,



- jazyk s jedným symbolom a abecedy, $R = a$.

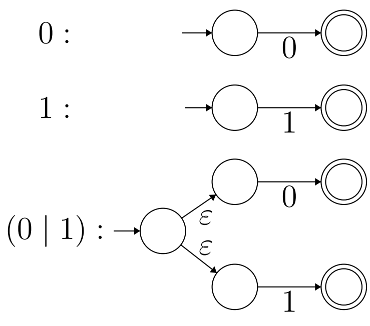


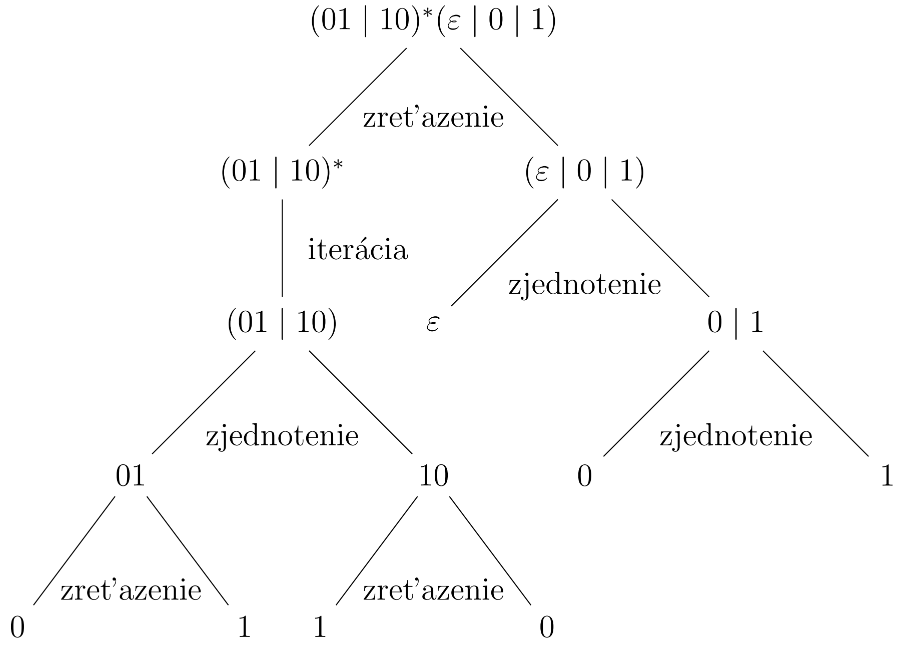
Regulárny výraz $R = (01 \mid 10)^*(\varepsilon \mid 0 \mid 1)$ rozdelíme na aplikácie jednotlivých operácií. Výsledok tohto delenia tvorí strom, ktorý je uvedený na obrázku 3.1, na ktorom sú znázornené príslušné operácie a menšie regulárne výrazy, na ktoré sú aplikované. Listy stromu sú tvorené elementárnymi regulárnymi výrazmi, v tomto prípade ε, 0 a 1.

Na základe uvedeného stromu postupne zostrojíme nedeterministické konečné automaty pre jednotlivé operácie. Začíname od listov stromu a postupujeme smerom ku koreňu.

1. Zjednotenie $(0 \mid 1)$

    - Ide o zjednotenie 2 elementárnych regulárnych výrazov, 0 a 1.
    - Vychádzajúc z NKA pre elementárne regulárne výrazy 0 a 1 a z konštrukcie NKA pre zjednotenie dvoch regulárnych výrazov dostávame NKA pre zjednotenie $0 \mid 1$:

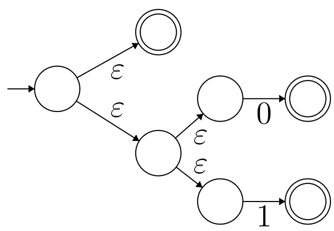




Obr. 3.1: Strom rozdelenia výrazu $(01 \mid 10)^*(\varepsilon \mid 0 \mid 1)$ na jednotlivé operácie

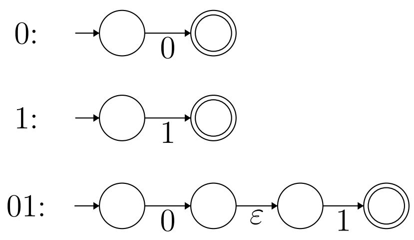
2. Zjednotenie $(\varepsilon \mid (0 \mid 1))$

   - Ide o zjednotenie elementárneho regulárneho výrazu ε a regulárneho výrazu $(0 \mid 1)$, pre ktorý sme zostrojili NKA v predchádzajúcom kroku.
   - Vychádzajúc z NKA pre regulárny výraz ε a pre $(0 \mid 1)$ a z konštrukcie NKA pre zjednotenie dvoch regulárnych výrazov, dostávame NKA pre zjednotenie $(\varepsilon \mid (0 \mid 1))$:

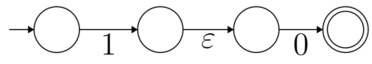


3. Zreťazenie (01)

   - Ide o zreťazenie 2 elementárnych regulárnych výrazov, 0 a 1.
   - Vychádzajúc z NKA pre elementárne regulárne výrazy 0 a 1 a z konštrukcie NKA pre zreťazenie dvoch regulárnych výrazov dostávame NKA pre zreťazenie 01:

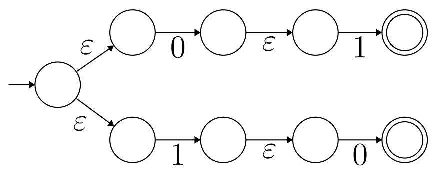


4. Zreťazenie (10) zostrojíme analogicky ako pre (01):



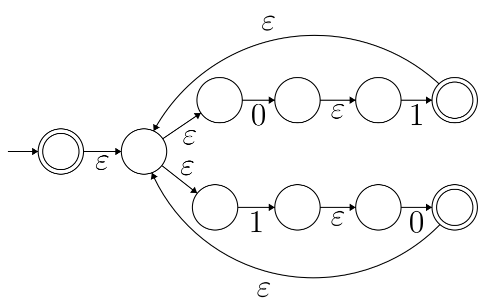
5. Zjednotenie (01 | 10)

   - Ide o zjednotenie 2 regulárnych výrazov, 01 a 10.
   - Keďže v predchádzajúcich krokoch sme zostrojili NKA pre regulárne výrazy 01 a 10, z konštrukcie NKA pre zjednotenie dvoch regulárnych výrazov dostávame NKA pre zjednotenie (01 | 10):



6. Iterácia $(01 \mid 10)^*$

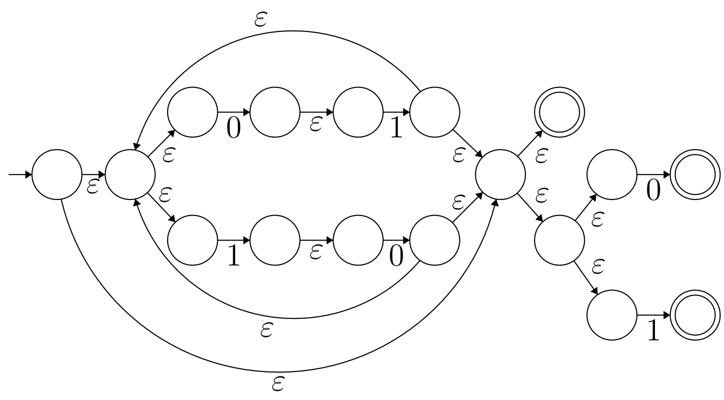
   - Ide o iteráciu regulárneho výrazu, $(01 \mid 10)^*$.
   - Keďže v predchádzajúcom kroku sme zostrojili NKA pre regulárny výraz (01 | 10), z konštrukcie NKA pre iteráciu regulárneho výrazu dostávame NKA pre iteráciu $(01 \mid 10)^*$:



7. Zreťazenie $(01 \mid 10)^*(\varepsilon \mid 0 \mid 1)$

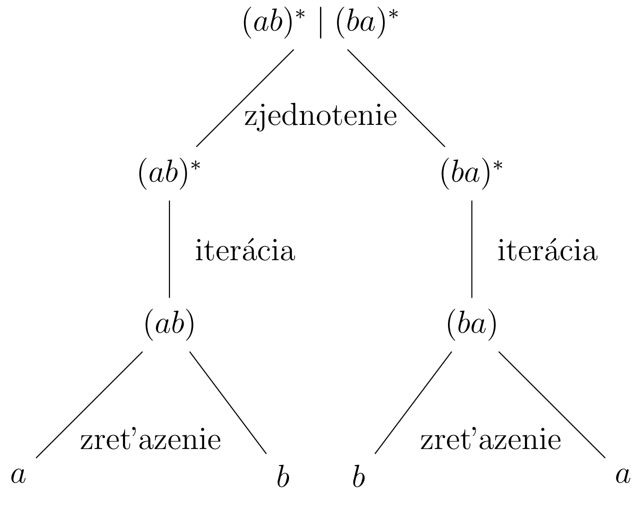
   - Ide o zreťazenie 2 regulárnych výrazov, $(01 \mid 10)^*$ a $(\varepsilon \mid 0 \mid 1)$.

- Vychádzajúc z NKA pre príslušné regulárne výrazy a z konštrukcie NKA pre zreťazenie dvoch regulárnych výrazov dostávame výsledný NKA pre zreťazenie $(01 \mid 10)^*(\varepsilon \mid 0 \mid 1)$, ktorý je zároveň aj hľadaným NKA pre zadaný regulárny výraz:



Úloha č. 3.2.2 Nájdite nedeterministický konečný automat ekvivalentný k regulárnemu výrazu $R = (ab)^* \mid (ba)^*$ nad abecedou $A = \{a, b\}$.

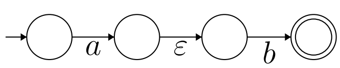
Riešenie: Rozdelenie regulárneho výrazu $R = (ab)^* \mid (ba)^*$ na aplikácie jednotlivých operácií na menšie regulárne výrazy je na obrázku 3.2.



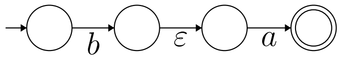
Obr. 3.2: Strom rozdelenia výrazu $(ab)^* \mid (ba)^*$ na jednotlivé operácie

Na základe uvedeného stromu postupne zostrojíme nedeterministické konečné automaty pre jednotlivé operácie. Začíname od listov stromu a postupujeme smerom ku koreňu.

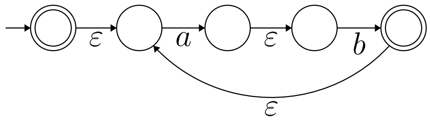
1. Zreťazenie (ab)



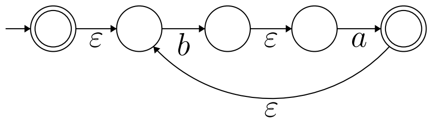
2. Zreťazenie (ba)



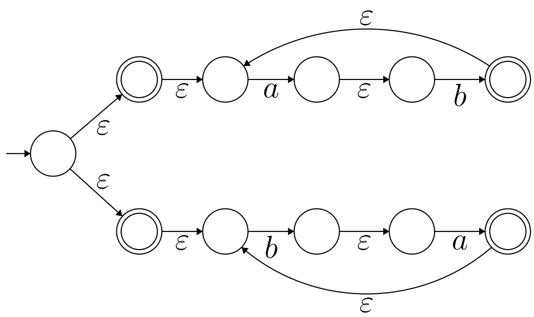
3. Iterácia $(ab)^*$



4. Iterácia $(ba)^*$



5. Zjednotenie $(ab)^* \mid (ba)^*$

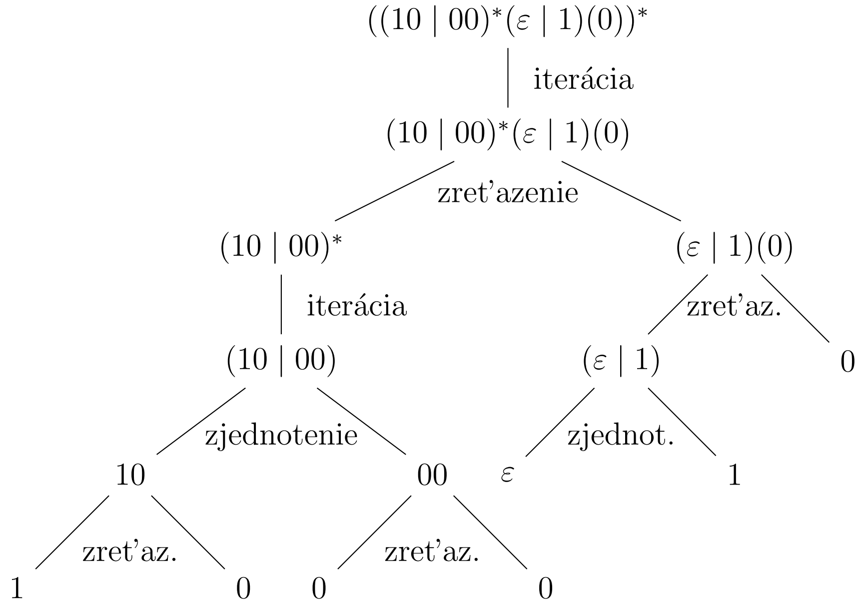


Úloha č. 3.2.3 Nájdite nedeterministický konečný automat ekvivalentný k regulárnemu výrazu $R = ((10 \mid 00)^*(\varepsilon \mid 1)(0))^*$ nad abecedou $A = \{0, 1\}$.

Riešenie: Rozdelenie regulárneho výrazu $R = ((10 \mid 00)^*(\varepsilon \mid 1)(0))^*$ na aplikácie jednotlivých operácií na menšie regulárne výrazy je na obrázku 3.3.

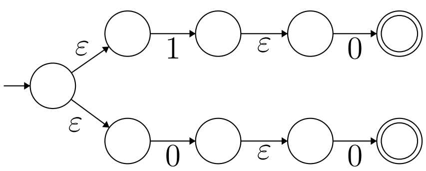
1. Zreťazenie (10)

2. Zreťazenie (00)

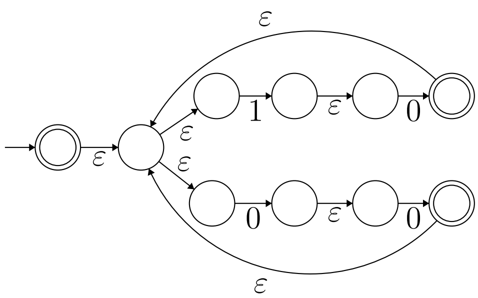


Obr. 3.3: Strom rozdelenia výrazu $((10 \mid 00)^*(\varepsilon \mid 1)(0))^*$ na jednotlivé operácie

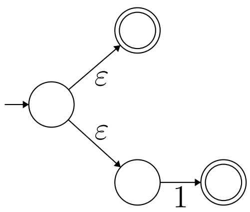
3. Zjednotenie $(10 \mid 00)$



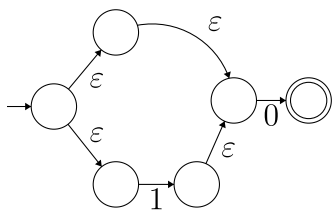
4. Iterácia $(10 \mid 00)^*$



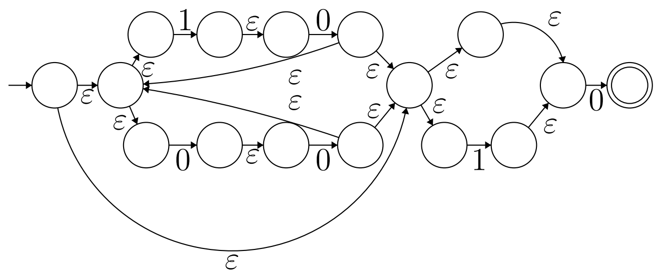
5. Zjednotenie $(\varepsilon \mid 1)$



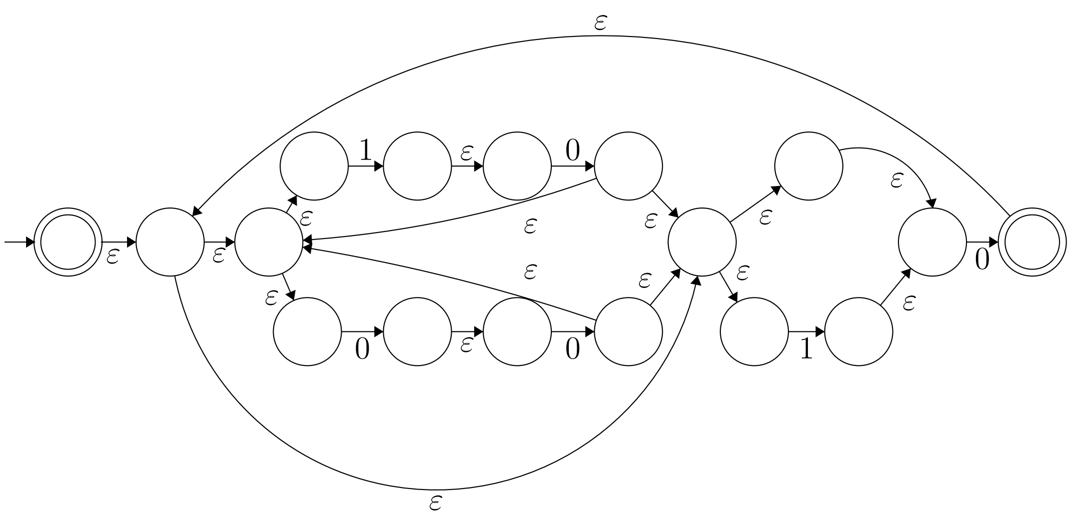
6. Zreťazenie $(\varepsilon \mid 1)(0)$



7. Zreťazenie $(10 \mid 00)^*(\varepsilon \mid 1)(0)$

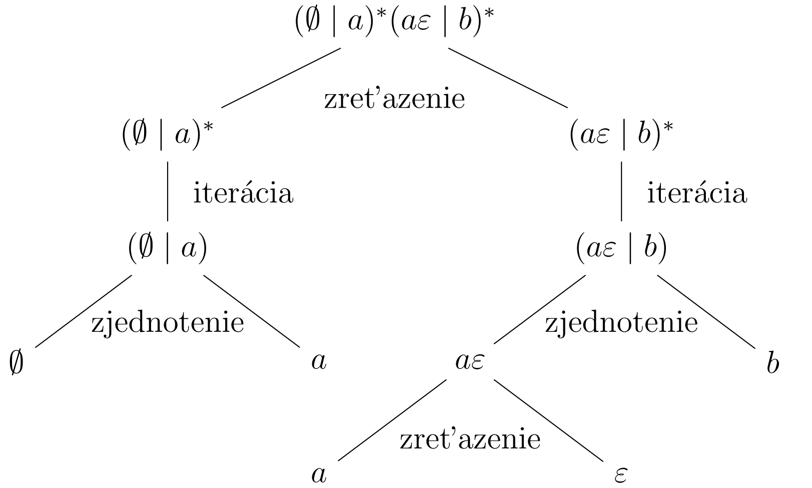


8. Iterácia $((10 \mid 00)^*(\varepsilon \mid 1)(0))^*$, teda výsledný NKA:



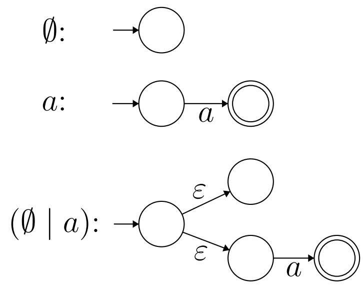
Úloha č. 3.2.4 Nájdite nedeterministický konečný automat ekvivalentný k regulárnemu výrazu $R = (\emptyset \mid a)^*(a\varepsilon \mid b)^*$ nad abecedou $A = \{a, b\}$.

Riešenie:
Rozdelenie regulárneho výrazu $R = (\emptyset \mid a)^*(a\varepsilon \mid b)^*$ na aplikácie jednotlivých operácií na menšie regulárne výrazy je na obrázku 3.4.

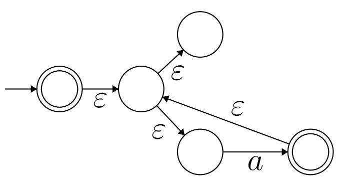


Obr. 3.4: Strom rozdelenia výrazu $(\emptyset \mid a)^*(a\varepsilon \mid b)^*$ na jednotlivé operácie

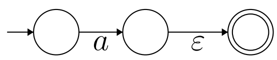
1. Zjednotenie $(\emptyset \mid a)$



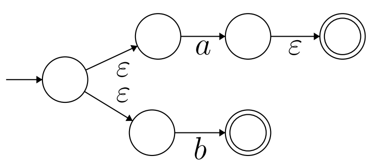
2. Iterácia $(\emptyset \mid a)^*$



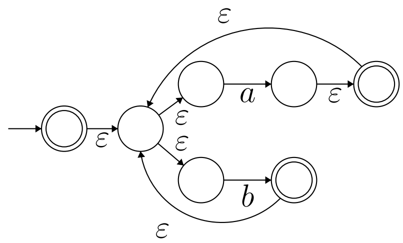
3. Zreťazenie $(a\varepsilon)$



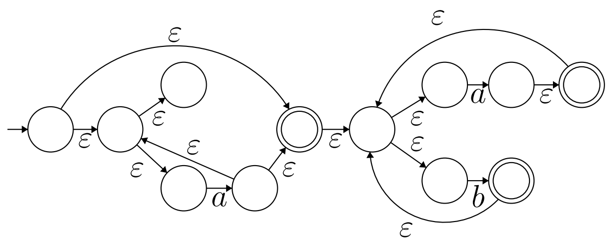
4. Zjednotenie $(a\varepsilon \mid b)$



5. Iterácia $(a\varepsilon \mid b)^*$

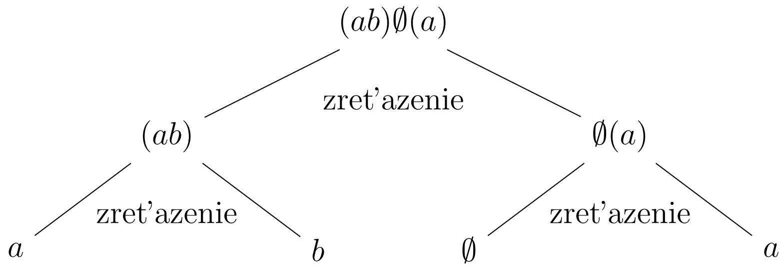


6. Zreťazenie $(\emptyset \mid a)^*(a\varepsilon \mid b)^*$



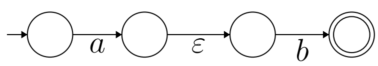
Úloha č. 3.2.5 Nájdite nedeterministický konečný automat ekvivalentný k regulárnemu výrazu $R = (ab)\emptyset(a)$ nad abecedou $A = \{a, b\}$.

Riešenie:
Rozdelenie regulárneho výrazu $R = (ab)\emptyset(a)$ na aplikácie jednotlivých operácií na menšie regulárne výrazy je na obrázku 3.5.



Obr. 3.5: Strom rozdelenia výrazu $(ab)\emptyset(a)$ na jednotlivé operácie

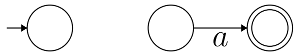
1. Zreťazenie (ab)



2. Zreťazenie $\emptyset(a)$

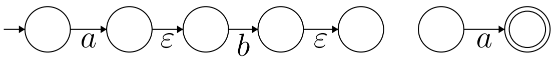
- Podľa schémy zreťazenia 2 príslušných NKA je potrebné zo všetkých **akceptačných stavov** automatu pre $\emptyset$ viesť ε prechody do počiatočného stavu automatu pre a a zároveň zameniť počiatočný stav v automate pre a za neakceptačný stav.

- Avšak, NKA pre $\emptyset$ **nemá žiadne akceptačné stavy!** Preto dostávame NKA, ktorý je rozdelený na 2 disjunktné časti. Je zrejmé, že tento NKA **akceptuje len prázdny jazyk**, pretože sa z počiatočného stavu **nie je možné** dostat' do akceptačného stavu — čo korešponduje so situáciou, že regulárny výraz $\emptyset(a)$ popisuje prázdny jazyk, pretože pre zret'azenie l'ubovol'ného jazyka s prázdnym jazykom platí, že výsledok je vždy prázdny jazyk.



3. Zret'azenie $(ab)\emptyset(a)$

- Ked'že toto zret'azenie vychádza z predošlých 2 automatov, aj výsledok tohto zret'azenia obsahuje 2 časti nespojené žiadnym prechodom, pričom počiatočný stav sa nachádza v prvej časti a akceptačný stav v druhej časti.
- Teda výsledný automat určite neakceptuje žiadne ret'azce, teda akceptuje len prázdny jazyk $\emptyset$.
- Čo znovu korešponduje s faktom, že regulárny výraz $(ab)\emptyset(a)$ popisuje prázdny jazyk, ked'že zret'azenie l'ubovol'ného jazyka s prázdnym jazykom je znovu len prázdny jazyk.



Kapitola 4

Bezkontextové gramatiky

4.1 Derivačné stromy

Úloha č. 4.1.1 Je daná bezkontextová gramatika $G = (\{S, A, B\}, \{a, b\}, P, S)$ s pravidlami P:

- $S \to AB \mid Sba$
- $A \to aB \mid aBb$
- $B \to \varepsilon \mid a \mid b$

Pre uvedenú gramatiku:

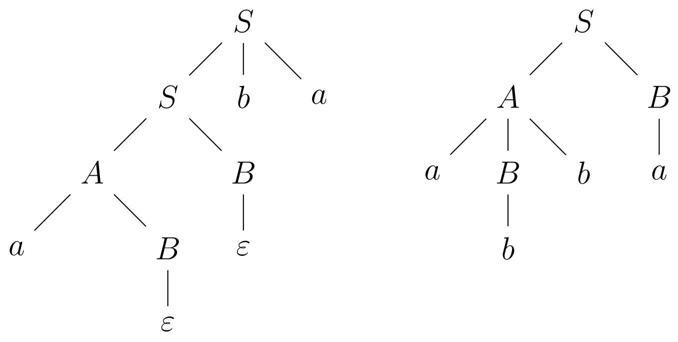
1. Nájdite derivácie ret'azcov $aba, abba$ a nakreslite ich derivačné stromy.
2. Zostrojte l'avé a pravé derivácie ret'azcov $aba, abba$.
3. Je daná gramatika jednoznačná?

Riešenie:

1. Derivácie ret'azcov aba a $abba$ nájdeme v tomto prípade skusmo, dostávame napríklad nasledovné derivácie (**tučným písmom** je v každej vetnej forme zvýraznený ten neterminál, na ktorý sme aplikovali pravidlo gramatiky v danom derivačnom kroku):

$$\mathbf{S} \Rightarrow \mathbf{S}ba \Rightarrow A\mathbf{B}ba \Rightarrow \mathbf{A}ba \Rightarrow a\mathbf{B}ba \Rightarrow aba$$
$$\mathbf{S} \Rightarrow \mathbf{A}B \Rightarrow a\mathbf{B}bB \Rightarrow abb\mathbf{B} \Rightarrow abba$$

   Derivačné stromy, prislúchajúce uvedeným deriváciám, sú uvedené na obrázku 4.1:

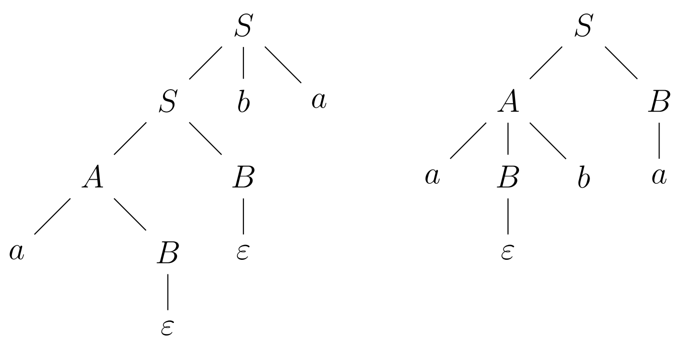


Obr. 4.1: Derivačné stromy ret'azcov $aba, abba$

2. Ľavé/pravé derivácie ret'azcov sú derivácie, ktoré sú špecifické tým, že vždy aplikujeme pravidlo gramatiky na prvý neterminál zľava/sprava. Uvádzame príklad ľavej ($\Rightarrow_l$) a pravej ($\Rightarrow_r$) derivácie uvedených ret'azcov:

$$\mathbf{S} \Rightarrow_l \mathbf{S}ba \Rightarrow_l \mathbf{A}Bba \Rightarrow_l a\mathbf{B}Bba \Rightarrow_l a\mathbf{B}ba \Rightarrow_l aba$$
$$\mathbf{S} \Rightarrow_r \mathbf{S}ba \Rightarrow_r A\mathbf{B}ba \Rightarrow_r \mathbf{A}ba \Rightarrow_r a\mathbf{B}ba \Rightarrow_r aba$$
$$\mathbf{S} \Rightarrow_l \mathbf{A}B \Rightarrow_l a\mathbf{B}bB \Rightarrow_l abb\mathbf{B} \Rightarrow_l abba$$
$$\mathbf{S} \Rightarrow_r A\mathbf{B} \Rightarrow_r \mathbf{A}a \Rightarrow_r a\mathbf{B}ba \Rightarrow_r abba$$

3. Aby sme zistili, či je gramatika **nejednoznačná**, musíme nájsť' aspoň jeden ret'azec generovaný gramatikou, ktorý má aspoň 2 **rôzne derivačné stromy**. Napríklad ret'azec aba má 2 rôzne derivačné stromy, uvedené na obrázku 4.2:



Obr. 4.2: Rôzne derivačné stromy ret'azca aba

Ked'že gramatika generuje taký ret'azec (aba), ktorý má 2 rôzne derivačné stromy, gramatika G zadaná v tejto úlohe je **nejednoznačná**.

Úloha č. 4.1.2 Je daná bezkontextová gramatika $G = (\{S, A, B\}, \{a, b\}, P, S)$ s pravidlami P:

- $S \to AaB \mid BbA$
- $A \to bAa \mid \varepsilon$
- $B \to b \mid S$

Pre uvedenú gramatiku:

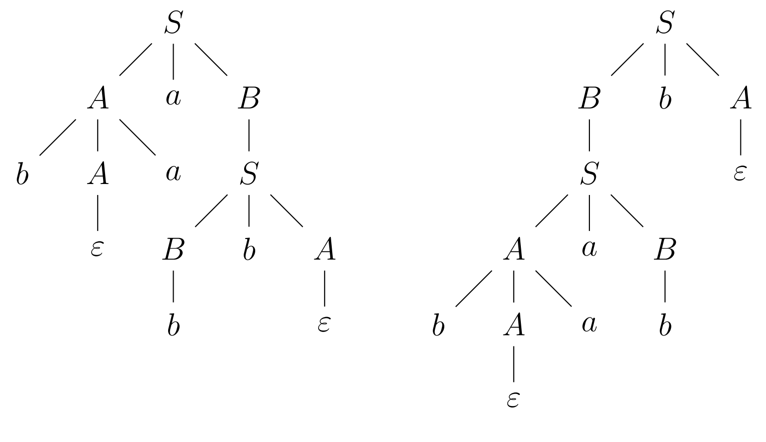
1. Zostrojte ľavú a pravú deriváciu reťazca *baabb*.
2. Dokážte, že uvedená gramatika je nejednoznačná tým, že nájdete 2 rôzne derivačné stromy reťazca *baabb*.

Riešenie:

1. Ľavá a pravá derivácia reťazca *baabb* by mohli vyzerať napríklad nasledovne:

$$\mathbf{S} \Rightarrow_l \mathbf{B}bA \Rightarrow_l \mathbf{S}bA \Rightarrow_l \mathbf{A}aBbA \Rightarrow_l b\mathbf{A}aaBbA \Rightarrow_l baa\mathbf{B}bA \Rightarrow_l baabb\mathbf{A} \Rightarrow_l baabb$$
$$\mathbf{S} \Rightarrow_r Bb\mathbf{A} \Rightarrow_r \mathbf{B}b \Rightarrow_r \mathbf{S}b \Rightarrow_r Aa\mathbf{B}b \Rightarrow_r \mathbf{A}abb \Rightarrow_r b\mathbf{A}aabb \Rightarrow_r baabb$$

2. Reťazec *baabb* má 2 rôzne derivačné stromy, uvedené na obrázku 4.3:



Obr. 4.3: Rôzne derivačné stromy reťazca *baabb*

Úloha č. 4.1.3 Je daná bezkontextová gramatika $G = (N, T, P, S)$, ktorej neterminály $N = \{$<program>, <deklaracie>, <deklaracia>, <typ>, <prikazy>$\}$, terminály $T = \{\texttt{float}, \texttt{int}, \texttt{id}, \texttt{P}, \texttt{;}\}$, <program> je počiatočný neterminál, s pravidlami P:

- <program> $\rightarrow$ <deklaracie><prikazy>
- <deklaracie> $\rightarrow$ <deklaracia> | <deklaracie><deklaracia>
- <deklaracia> $\rightarrow$ <typ> `id ;`
- <typ> $\rightarrow$ `int` | `float`
- <prikazy> $\rightarrow$ `P`

Pre uvedenú gramatiku:

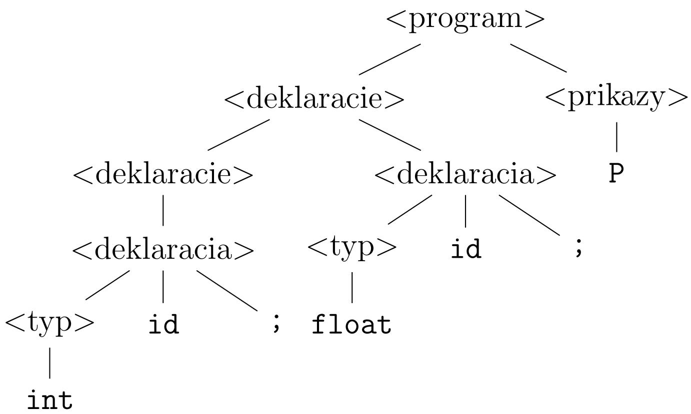
1. Zostrojte ľavú a pravú derivácia reťazca `int id ; float id ; P`
2. Uveďte derivačný strom pre reťazec `int id ; float id ; P`

Riešenie:

1. Ľavá a pravá derivácia reťazca `int id ; float id ; P` by mohli vyzerať napríklad nasledovne:

$$
\begin{aligned}
<\text{program}> &\Rightarrow_l <\text{deklaracie}><\text{prikazy}> \Rightarrow_l \\
&\Rightarrow_l <\text{deklaracie}><\text{deklaracia}><\text{prikazy}> \Rightarrow_l \\
&\Rightarrow_l <\text{deklaracia}><\text{deklaracia}><\text{prikazy}> \Rightarrow_l \\
&\Rightarrow_l <\text{typ}>\ \texttt{id ;}\ <\text{deklaracia}><\text{prikazy}> \Rightarrow_l \\
&\Rightarrow_l \texttt{int id ;}\ <\text{deklaracia}><\text{prikazy}> \Rightarrow_l \\
&\Rightarrow_l \texttt{int id ;}\ <\text{typ}>\ \texttt{id ;}\ <\text{prikazy}> \Rightarrow_l \\
&\Rightarrow_l \texttt{int id ; float id ;}\ <\text{prikazy}> \Rightarrow_l \\
&\Rightarrow_l \texttt{int id ; float id ; P} \\
<\text{program}> &\Rightarrow_r <\text{deklaracie}><\text{prikazy}> \Rightarrow_r <\text{deklaracie}>\texttt{P} \Rightarrow_r \\
&\Rightarrow_r <\text{deklaracie}><\text{deklaracia}>\ \texttt{P} \Rightarrow_r \\
&\Rightarrow_r <\text{deklaracie}><\text{typ}>\ \texttt{id ; P} \Rightarrow_r \\
&\Rightarrow_r <\text{deklaracie}>\ \texttt{float id ; P} \Rightarrow_r \\
&\Rightarrow_r <\text{deklaracia}>\ \texttt{float id ; P} \Rightarrow_r \\
&\Rightarrow_r <\text{typ}>\ \texttt{id ; float id ; P} \Rightarrow_r \texttt{int id ; float id ; P}
\end{aligned}
$$

2. Derivačný strom reťazca `int id ; float id ; P` je uvedený na obrázku 4.4.



Obr. 4.4: Derivačný strom reťazca `int id ; float id ; P`

4.2 Redukcie gramatík

Úloha č. 4.2.1 Je daná bezkontextová gramatika $G = (\{S, A, B\}, \{a, b\}, P, S)$ s pravidlami:

- $S \to aA \mid aS \mid A \mid abB$
- $A \to \varepsilon \mid bA$
- $B \to bB$

Transformujte uvedenú gramatiku na ekvivalentnú redukovanú gramatiku.

Riešenie:
Transformáciu gramatiky na ekvivalentnú redukovanú gramatiku (redukcia gramatiky) vykonáme v 2 krokoch:

1. V prvom kroku nájdeme množinu neterminálov $N_T = \{A \in N \mid A \Rightarrow^* w, w \in T^*\}$, t. j. takých neterminálov, z ktorých je možné v gramatike odvodiť nejaký reťazec nad terminálnymi symbolmi, a odstránime z gramatiky tie neterminály, ktoré **nepatria** do množiny N_T.

2. V druhom kroku, po odstránení týchto neterminálov, hľadáme v gramatike množinu terminálov a neterminálov $V_D = \{X \in N \cup T \mid S \Rightarrow^* \alpha X \beta, \alpha, \beta \in (N \cup T)^*\}$, t. j. takých symbolov gramatiky, ktoré sa môžu vyskytovať v nejakej vetnej forme. Následne z gramatiky odstránime tie symboly, ktoré **nepatria** do množiny V_D.

3. Tým dostávame tzv. **redukovanú gramatiku**, ktorá obsahuje len také neterminály, z ktorých je možné odvodiť nejaký terminálny reťazec, a len také neterminály a terminály, ktoré sú dosiahnuteľné, teda sa môžu vyskytovať vo vetnej forme.

Hľadanie množiny N_T :

1. Na začiatku nastavíme množinu N_T ako prázdnu množinu, $N_T = \emptyset$.
2. Následne do nej pridáme tie neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom nad terminálmi**.
   - Neterminál A v dôsledku pravidla $A \to \varepsilon$.
   - Iné neterminály tam zatiaľ nepridáme, pretože ani S, ani B nemajú také pravidlá, že by sa na ich pravej strane nachádzal reťazec nad terminálmi.
   - Teda po tomto kroku $N_T = \{A\}$, pretože sme zistili, že z neterminálu A je určite možné odvodiť nejaký reťazec nad terminálmi (ε).
3. V ďalšom kroku do množiny N_T pridáme tie neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom zloženým z terminálov a/alebo neterminálov, o ktorých vieme, že patria do** N_T.
   - Neterminál S v dôsledku pravidla $S \to aA$, prípadne $S \to A$, pretože v oboch pravidlách je na pravej strane reťazec zložený z terminálov, prípadne neterminálu A, o ktorom vieme, že patrí do množiny N_T.
   - Iné neterminály tam zatiaľ nepridáme, pretože B nemá také pravidlá, že by sa na ich pravej strane nachádzal reťazec nad terminálmi alebo reťazec zložený z terminálov a neterminálu A.
   - Teda po tomto kroku $N_T = \{A, S\}$.
4. V ďalšom kroku opakujeme predchádzajúci proces, avšak so zmenenou množinou N_T, pretože nám do nej pribudol neterminál S.
   - Avšak zistíme, že žiaden nový neterminál tam nepribudne, pretože B nemá také pravidlá, že by sa na ich pravej strane nachádzal reťazec nad terminálmi alebo reťazec zložený z terminálov a neterminálov S alebo A.
   - Teda po tomto kroku zostáva množina $N_T = \{A, S\}$.
5. Ak nastane situácia, že sa nám množina N_T nezmení, hľadanie množiny N_T končí. V našom prípade je teda $N_T = \{S, A\}$. Keďže neterminál B do tejto množiny nepatrí, **odstránime ho z gramatiky**, spolu so všetkými pravidlami, v ktorých vystupuje.

Keď z gramatiky odstránime neterminál B a pravidlá, v ktorých vystupuje, dostávame množinu pravidiel:

- $S \to aA \mid aS \mid A$
- $A \to \varepsilon \mid bA$

Hľadanie množiny V_D :

1. Na začiatku nastavíme množinu V_D ako množinu obsahujúcu počiatočný neterminál S, $V_D = \{S\}$, keďže ten sa určite vyskytuje v počiatočnej vetnej forme S.
2. Následne do nej pridáme tie symboly gramatiky, ktoré sa nachádzajú v pravých stranách pravidiel pre neterminál S:
   - Terminál a a neterminál A v dôsledku pravidla $S \rightarrow aA$.
   - Neterminál S, ktorý sa nachádza na pravej strane pravidla $S \rightarrow aS$ tam samozrejme nepridáme, pretože sa už v množine V_D nachádza.
   - Iné symboly v pravých stranách pravidiel pre neterminál S nevystupujú.
   - Teda po tomto kroku $V_D = \{S, a, A\}$, pretože sme zistili, že vo vetných formách budú určite vystupovať symboly S, a a A.
3. V ďalšom kroku do množiny V_D pridáme tie symbol gramatiky, ktoré sa nachádzajú v pravých stranách pravidiel pre neterminály, ktoré nám v poslednom kroku pribudli do množiny V_D, t. j. pre neterminál A:
   - Z pravých strán pravidiel neterminálu A pridáme do množiny V_D terminál b v dôsledku pravidla $A \rightarrow bA$.
   - Iné symboly gramatiky sa na pravých stranách neterminálu A nevyskytujú. (Pripomíname, že ε **nie je** symbol gramatiky).
   - Teda po tomto kroku $V_D = \{S, a, A, b\}$.
4. Uvedený proces vždy opakujeme, ak nám do množiny V_D pribudne nový neterminál.
5. Keďže v poslednom kroku nám do množiny pribudol len terminál b, algoritmus končí a výsledná množina symbolov, ktoré sa môžu vyskytovať vo vetných formách je $V_D = \{S, a, A, b\}$.
6. Z gramatiky by sme odstránili tie symboly, ktoré nie sú v množine V_D. V tomto prípade sa však každý symbol gramatiky, ktorú sme dostali po odstránení neterminálov, ktoré nepatrili do N_T, nachádza v množine V_D, teda v tomto kroku neodstránime z gramatiky nič

Dostávame teda výslednú redukovanú gramatiku $G_{red} = (\{S, A\}, \{a, b\}, P, S)$, ktorej pravidlá sú:

- $S \rightarrow aA \mid aS \mid A$
- $A \rightarrow \varepsilon \mid bA$

Úloha č. 4.2.2 Je daná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$ s pravidlami:

- $S \rightarrow aAC \mid bB \mid BB$
- $A \rightarrow aAb$
- $B \rightarrow \varepsilon \mid aBb \mid SB$
- $C \rightarrow CC \mid b \mid c$

Transformujte uvedenú gramatiku na ekvivalentnú redukovanú gramatiku.

Riešenie:
Hľadanie množiny N_T :

1. $N_T = \emptyset$.
2. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom nad terminálmi**.
   - Neterminál B v dôsledku pravidla $B \rightarrow \varepsilon$.
   - Neterminál C v dôsledku pravidla $C \rightarrow b$, resp. $C \rightarrow c$.
   - Po tejto iterácii teda $N_T = \{B, C\}$.
3. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom zloženým z terminálov a/alebo neterminálov, o ktorých vieme, že patria do N_T, t. j.** B **alebo** C.
   - Neterminál S v dôsledku pravidla $S \rightarrow bB$, resp. $S \rightarrow BB$.
   - Po tejto iterácii $N_T = \{B, C, S\}$.
4. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom zloženým z terminálov a/alebo neterminálov** B**,** C **alebo** S.
   - Žiaden nový neterminál nám nepribudne do N_T.
   - Po tejto iterácii zostáva množina $N_T = \{B, C, S\}$.
5. Keďže neterminál $A \notin N_T$, odstránime ho z gramatiky.

Dostávame množinu pravidiel:

- $S \rightarrow bB \mid BB$
- $B \rightarrow \varepsilon \mid aBb \mid SB$
- $C \rightarrow CC \mid b \mid c$

Hľadanie množiny V_D :

1. $V_D = \{S\}$.
2. Pridáme symboly gramatiky z pravých strán pravidiel neterminálu S:
   - Terminál b a neterminál B v dôsledku pravidla $S \to bB$.
   - Po tejto iterácii $V_D = \{S, b, B\}$.
3. V ďalšom kroku do množiny V_D pridáme symboly gramatiky z pravých strán pravidiel pre neterminál B:
   - Terminál a v dôsledku pravidla $B \to aBb$.
   - Po tejto iterácii $V_D = \{S, b, B, a\}$.
4. Keďže nám nepribudol nový neterminál do V_D, algoritmus končí.
5. Z gramatiky odstránime tie symboly, ktoré nie sú v množine V_D, teda odstránime neterminál C a terminál c.

Dostávame teda výslednú redukovanú gramatiku $G_{red} = (\{S, B\}, \{a, b\}, P, S)$, ktorej pravidlá sú:

- $S \to bB \mid BB$
- $B \to \varepsilon \mid aBb \mid SB$

Úloha č. 4.2.3 Je daná bezkontextová gramatika $G = (\{S, A, B, C, D, E\}, \{a, b, c, d\}, P, S)$ s pravidlami:

- $S \to abD$
- $A \to BaB \mid bB$
- $B \to AB \mid AC$
- $C \to S \mid b \mid c$
- $D \to C$
- $E \to S \mid d \mid \varepsilon$

Transformujte uvedenú gramatiku na ekvivalentnú redukovanú gramatiku.

Riešenie:
Hľadanie množiny N_T :

1. $N_T = \emptyset$.
2. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom nad terminálmi**.

- Neterminál C v dôsledku pravidla $S \rightarrow b$, resp. $S \rightarrow c$.
- Neterminál E v dôsledku pravidla $E \rightarrow d$, resp. $E \rightarrow \varepsilon$.
- Po tejto iterácii teda $N_T = \{C, E\}$.

3. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom zloženým z terminálov a/alebo neterminálov** C **alebo** E.
   - Neterminál D v dôsledku pravidla $D \rightarrow C$.
   - Po tejto iterácii $N_T = \{C, E, D\}$.
4. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom zloženým z terminálov a/alebo neterminálov** C**,** E **alebo** D.
   - Neterminál S v dôsledku pravidla $S \rightarrow abD$.
   - Po tejto iterácii $N_T = \{C, E, D, S\}$.
5. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom zloženým z terminálov a/alebo neterminálov** C**,** E **,** D **alebo** S.
   - Žiaden nový neterminál nám nepribudne do N_T.
   - Výsledná množina $N_T = \{C, E, D, S\}$.
6. Keďže neterminály $A, B \notin N_T$, odstránime ich aj s pravidlami, kde vystupujú.

Dostávame množinu pravidiel:

- $S \rightarrow abD$
- $C \rightarrow S \mid b \mid c$
- $D \rightarrow C$
- $E \rightarrow S \mid d \mid \varepsilon$

Hľadanie množiny V_D :

1. $V_D = \{S\}$.
2. Pridáme symboly gramatiky z pravých strán pravidiel neterminálu S:
   - Terminály a, b a neterminál D v dôsledku pravidla $S \rightarrow abD$.
   - Po tejto iterácii $V_D = \{S, a, b, D\}$.
3. V ďalšom kroku do množiny V_D pridáme symboly gramatiky z pravých strán pravidiel pre neterminál D:

- Neterminál C v dôsledku pravidla $D \to C$.
- Po tejto iterácii $V_D = \{S, a, b, D, C\}$.

4. V ďalšom kroku do množiny V_D pridáme symboly gramatiky z pravých strán pravidiel pre neterminál C:
   - Terminál c v dôsledku pravidla $C \to c$.
   - Po tejto iterácii $V_D = \{S, a, b, D, C, c\}$.
5. Keďže nám nepribudol nový neterminál do V_D, algoritmus končí.
6. Z gramatiky odstránime tie symboly, ktoré nie sú v množine V_D, teda odstránime neterminál E a terminál d.

Dostávame teda výslednú redukovanú gramatiku $G_{red} = (\{S, C, D\}, \{a, b, c\}, P, S)$, ktorej pravidlá sú:

- $S \to abD$
- $C \to S \mid b \mid c$
- $D \to C$

Úloha č. 4.2.4 Je daná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$ s pravidlami:

- $S \to aAb$
- $A \to \varepsilon \mid aAA$
- $B \to b \mid CC$
- $C \to c \mid aBb$

Transformujte uvedenú gramatiku na ekvivalentnú redukovanú gramatiku.

Riešenie:
Hľadanie množiny N_T :

1. $N_T = \emptyset$.
2. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom nad terminálmi**.
   - Neterminál A v dôsledku pravidla $A \to \varepsilon$.
   - Neterminál B v dôsledku pravidla $B \to b$.
   - Neterminál C v dôsledku pravidla $C \to c$.
   - Po tejto iterácii $N_T = \{A, B, C\}$.

3. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom zloženým z terminálov a/alebo neterminálov** A, B, C:
   - Neterminál S v dôsledku pravidla $S \to aAb$.
   - Po tejto iterácii $N_T = \{A, B, C, S\}$.
4. Keďže sme dostali množinu N_T, v ktorej sú všetky neterminály gramatiky, algoritmus končí a v tomto kroku z gramatiky nič neodstránime.

Množina pravidiel sa teda zatiaľ nemení:

- $S \to aAb$
- $A \to \varepsilon \mid aAA$
- $B \to b \mid CC$
- $C \to c \mid aBb$

Hľadanie množiny V_D :

1. $V_D = \{S\}$.
2. Pridáme symboly gramatiky z pravých strán pravidiel neterminálu S:
   - Terminály a, b a neterminál A v dôsledku pravidla $S \to aAb$.
   - Po tejto iterácii $V_D = \{S, a, b, A\}$.
3. V ďalšom kroku do množiny V_D pridáme symboly gramatiky z pravých strán pravidiel pre neterminál A:
   - Keďže na pravých stranách pravidiel pre neterminál A sú len symboly a, A, ktoré v V_D máme, nepridáme nič.
   - Po tejto iterácii sa teda V_D nezmení, $V_D = \{S, a, b, A\}$.
4. Keďže nám nepribudol nový neterminál do V_D, algoritmus končí.
5. Z gramatiky odstránime tie symboly, ktoré nie sú v množine V_D, teda odstránime neterminály B, C a terminál c.

Dostávame teda výslednú redukovanú gramatiku $G_{red} = (\{S, A\}, \{a, b\}, P, S)$, ktorej pravidlá sú:

- $S \to aAb$
- $A \to \varepsilon \mid aAA$

Úloha č. 4.2.5 Je daná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b\}, P, S)$ s pravidlami:

- $S \to ABC$
- $A \to BBa$
- $B \to AaC$
- $C \to aAb \mid abb$

Transformujte uvedenú gramatiku na ekvivalentnú redukovanú gramatiku.

Riešenie:
Hľadanie množiny N_T :

1. $N_T = \emptyset$.
2. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom nad terminálmi**.
   - Neterminál C v dôsledku pravidla $C \to abb$.
   - Po tejto iterácii $N_T = \{C\}$.
3. Pridáme neterminály, pre ktoré existuje v gramatike pravidlo, ktorého pravá strana je tvorená **reťazcom zloženým z terminálov a/alebo neterminálu** C:
   - Žiaden nový neterminál nám nepribudne do N_T.
   - Výsledná množina $N_T = \{C\}$.
4. Z gramatiky teda odstránime všetky neterminály, ktoré nie sú v množine N_T, teda odstránime neterminály S, A, B spolu s príslušnými pravidlami.
5. Tým dostávame gramatiku **bez** počiatočného neterminálu, keďže počiatočný neterminál S sme odstránili, pretože $S \notin N_T$. Gramatika teda negeneruje žiadne reťazce. V takom prípade nemá zmysel pokračovať s tvorbou množiny V_D. Keďže zadaná gramatika negeneruje žiadne reťazce, **nemá zmysel ju redukovať**.
6. Platí teda, že ak **počiatočný neterminál** S **nie je** elementom množiny N_T, gramatika **negeneruje žiadne reťazce**, a teda nemá zmysel uvažovať jej transformáciu na redukovanú gramatiku.

4.3 Množina N_ε

Úloha č. 4.3.1 Je daná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

- $S \to ABC \mid aSb$
- $A \to aAb \mid c \mid C$
- $B \to BabB \mid AA$
- $C \to \varepsilon \mid baCab$

Nájdite množinu N_ε pre uvedenú gramatiku.

Riešenie:
Hľadanie množiny $N_\varepsilon = \{A \in N \mid A \Rightarrow^* \varepsilon\}$, teda množiny všetkých tých neterminálov, z ktorých je možné odvodiť ε, postupuje nasledovným spôsobom:

1. V prvom kroku do množiny N_ε pridáme všetky také neterminály, ktorých pravidlá majú na pravej strane priamo ε:
   - V našom prípade je to neterminál C, v dôsledku pravidla $C \to \varepsilon$. Teda $N_\varepsilon = \{C\}$.
2. Následne budeme opakovať nasledovný proces: do množiny N_ε pridáme všetky také neterminály, ktorých pravidlá majú na pravej strane reťazec pozostávajúci len z neterminálov, o ktorých vieme, že patria do množiny N_ε :
   - Aktuálne vieme, že $N_\varepsilon = \{C\}$. Do množiny N_ε pridáme teda tie neterminály, ktoré majú aspoň 1 také pravidlo, že na jeho pravej strane sa nachádza reťazec, ktorý obsahuje len symboly C:
     - Takým pravidlom je napríklad $A \to C$ pre neterminál A.
     - Preto do množiny N_ε pridáme aj neterminál A, $N_\varepsilon = \{C, A\}$.
   - Aktuálne vieme, že $N_\varepsilon = \{C, A\}$. Do množiny N_ε pridáme teda tie neterminály, ktoré majú aspoň 1 také pravidlo, že na jeho pravej strane sa nachádza reťazec, ktorý obsahuje symboly C alebo A:
     - Takým pravidlom je napríklad $B \to AA$ pre neterminál B, pretože jeho pravá strana je tvorená len opakovaním neterminálu A.
     - Preto do množiny N_ε pridáme aj neterminál B, $N_\varepsilon = \{C, A, B\}$.
   - Aktuálne vieme, že $N_\varepsilon = \{C, A, B\}$. Do množiny N_ε pridáme teda tie neterminály, ktoré majú aspoň 1 také pravidlo, že na jeho pravej strane sa nachádza reťazec, ktorý obsahuje symboly C, A alebo B:
     - Takým pravidlom je napríklad $S \to ABC$ pre neterminál S, pretože jeho pravá strana je reťazec tvorený neterminálmi A, B, C.

  - Preto do množiny N_ε pridáme aj neterminál S, $N_\varepsilon = \{C, A, B, S\}$.
- V ďalšom kroku by sme už do množiny N_ε nový neterminál nepridali, takže výsledná množina $N_\varepsilon = \{C, A, B, S\}$.

Úloha č. 4.3.2 Je daná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

- $S \to AbBB$
- $A \to CC \mid cSA$
- $B \to aCa \mid bb \mid \varepsilon$
- $C \to \varepsilon \mid BC$

Nájdite množinu N_ε pre uvedenú gramatiku.

Riešenie:

1. V prvom kroku do množiny N_ε pridáme všetky také neterminály, ktorých pravidlá majú na pravej strane priamo ε:
   - V našom prípade sú to neterminály B a C, v dôsledku pravidiel $B \to \varepsilon$ a $C \to \varepsilon$. Teda $N_\varepsilon = \{B, C\}$.
2. Následne budeme opakovať nasledovný proces: do množiny N_ε pridáme všetky také neterminály, ktorých pravidlá majú na pravej strane reťazec pozostávajúci len z neterminálov, o ktorých vieme, že patria do množiny N_ε :
   - Aktuálne vieme, že $N_\varepsilon = \{B, C\}$. Do množiny N_ε pridáme teda tie neterminály, ktoré majú aspoň 1 také pravidlo, že na jeho pravej strane sa nachádza reťazec, ktorý obsahuje len symboly B a C:
     - Takým pravidlom je napríklad $A \to CC$ pre neterminál A.
     - Preto do množiny N_ε pridáme aj neterminál A, $N_\varepsilon = \{B, C, A\}$.
   - Aktuálne vieme, že $N_\varepsilon = \{B, C, A\}$. Do množiny N_ε pridáme teda tie neterminály, ktoré majú aspoň 1 také pravidlo, že na jeho pravej strane sa nachádza reťazec, ktorý obsahuje symboly B, C alebo A:
     - V tomto kroku už žiaden neterminál nepridáme. Jediný neterminál, ktorý by potenciálne prichádzal do úvahy je neterminál S, avšak jeho jediné pravidlo je tvaru $S \to AbBB$, teda na pravej strane sa **nenachádza** reťazec zložený len zo symbolov A, B, C, pretože navyše obsahuje aj terminál b.
     - Výsledná množina $N_\varepsilon = \{A, B, C\}$.

Úloha č. 4.3.3 Je daná bezkontextová gramatika $G = (\{\text{<blok>}, \text{<prikazy>}, \text{<prikaz>}\}, \{\texttt{p}, \texttt{;}, \texttt{begin}, \texttt{end}\}, P, \text{<blok>})$, ktorej pravidlá P sú:

- $\text{<blok>} \rightarrow$ `begin` <prikazy> `end`
- $\text{<prikazy>} \rightarrow \text{<prikaz>}$ `;` $\text{<prikazy>} \mid \varepsilon$
- $\text{<prikaz>} \rightarrow \text{<blok>} \mid$ `p` $\mid \varepsilon$

Nájdite množinu N_ε pre uvedenú gramatiku.

Riešenie:

1. V prvom kroku do množiny N_ε pridáme všetky také neterminály, ktorých pravidlá majú na pravej strane priamo ε:
   - V našom prípade sú to neterminály <prikazy> a <prikaz>, v dôsledku pravidiel $\text{<prikazy>} \rightarrow \varepsilon$ a $\text{<prikaz>} \rightarrow \varepsilon$.
   - Teda po tomto kroku $N_\varepsilon = \{\text{<prikazy>}, \text{<prikaz>}\}$.
2. Následne by sme do množiny N_ε pridali také neterminály, ktorých pravidlá majú na pravej strane reťazec pozostávajúci len z neterminálov, o ktorých vieme, že patria do množiny N_ε :
   - Aktuálne vieme, že $N_\varepsilon = \{\text{<prikazy>}, \text{<prikaz>}\}$. Do množiny N_ε pridáme teda tie neterminály, ktoré majú aspoň 1 také pravidlo, že na jeho pravej strane sa nachádza reťazec, ktorý obsahuje len symboly <prikazy> alebo <prikaz>.
     – V tomto kroku žiaden neterminál nepridáme. Jediný neterminál, ktorý by potenciálne prichádzal do úvahy je neterminál <blok>, avšak jeho jediné pravidlo je tvaru $\text{<blok>} \rightarrow$ `begin` <prikazy> `end`, teda na pravej strane sa **nenachádza** reťazec zložený len zo symbolov <prikazy> alebo <prikaz>, pretože navyše obsahuje aj terminály `begin` a `end`.
   - Výsledná množina $N_\varepsilon = \{\text{<prikazy>}, \text{<prikaz>}\}$.

4.4 Množina $FIRST$

Úloha č. 4.4.1 Je daná redukovaná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

- $S \to ABC \mid aSb$
- $A \to aAb \mid c \mid C$
- $B \to BabB \mid AA$
- $C \to \varepsilon \mid baCab$

Nájdite množinu $FIRST$ pre jednotlivé neterminály gramatiky.

Riešenie:
Množinu $FIRST$ štandardne určujeme pre redukované gramatiky. Keďže zadaná gramatika už redukovaná je, môžeme pristúpiť k jej hľadaniu.

Množina $FIRST(\alpha)$ nejakého reťazca symbolov gramatiky $\alpha \in (N \cup T)^*$ je definovaná ako:

$$FIRST(\alpha) = \{a \in T \mid \alpha \Rightarrow^* a\beta, \beta \in (N \cup T)^*\} \cup \{\varepsilon \mid \alpha \Rightarrow^* \varepsilon\}$$

Teda pre reťazec symbolov gramatiky α je jeho množina $FIRST$ tvorená:

1. všetkými terminálmi a, ktoré môžu stáť na **prvom mieste** reťazca symbolov gramatiky, ktorý vieme z reťazca α odvodiť,
2. symbolom ε, ak je v gramatike možné z reťazca α odvodiť **prázdny reťazec**.

Hľadanie množiny $FIRST$ pre jednotlivé neterminály gramatiky vykonáme v 3 krokoch:

1. Nájdeme množinu $N_\varepsilon = \{A \in N \mid A \Rightarrow^* \varepsilon\}$. Ak pre nejaký neterminál A platí $A \in N_\varepsilon$, potom určite $\varepsilon \in FIRST(A)$.
2. Ak gramatika pre nejaký neterminál A obsahuje pravidlo $A \to a\beta, a \in T$, $\beta \in (N \cup T)^*$, potom určite $a \in FIRST(A)$, teda ak existuje pravidlo, ktoré má na ľavej strane neterminál A a ktorého prvý symbol na pravej strane je terminál a, potom určite terminál a patrí do množiny $FIRST(A)$.
3. Ak gramatika pre nejaký neterminál A obsahuje pravidlo $A \to \beta, \beta \in (N \cup T)^*$, kde β začína neterminálom, potom platí $FIRST(\beta) \subseteq FIRST(A)$.

V prípade danej gramatiky:

1. Množina N_ε tejto gramatiky je $N_\varepsilon = \{S, A, B, C\}$, pozri príklad č. 4.3.1.

   Po tomto kroku teda $FIRST(S) = \{\varepsilon\}, FIRST(A) = \{\varepsilon\}$, $FIRST(B) = \{\varepsilon\}$, $FIRST(C) = \{\varepsilon\}$.

2. Na základe pravidiel, ktorých prvý symbol na pravej strane je terminál:
   - z pravidla $S \to aSb$ vieme, že $a \in FIRST(S)$,
   - z pravidiel $A \to aAb \mid c$ vieme, že $a, c \in FIRST(A)$,
   - z pravidla $C \to baCab$ vieme, že $b \in FIRST(C)$.

   Po tomto kroku teda $FIRST(S) = \{\varepsilon, a\}, FIRST(A) = \{\varepsilon, a, c\}$, $FIRST(B) = \{\varepsilon\}$, $FIRST(C) = \{\varepsilon, b\}$.

3. V ďalšom kroku budeme postupne prechádzať pravidlá, ktoré začínajú neterminálom a potenciálne pridávať nové terminály do množín $FIRST$ na základe ich aktuálnych hodnôt:
   - Pravidlo $S \to ABC$

     Podľa vyššie uvedeného pravidla platí $FIRST(ABC) \subseteq FIRST(S)$. Na základe aktuálnych hodnôt množín $FIRST$ vyšetríme, akú hodnotu môže mať $FIRST(ABC)$, teda aké symboly môžu stáť na prvom mieste reťazca, ktorý vieme odvodiť z ABC, prípadne ε, ak vieme odvodiť $ABC \Rightarrow^* \varepsilon$:
     - V reťazci ABC je na prvom mieste neterminál A. Jeho množina $FIRST(A)$ je aktuálne $FIRST(A) = \{a, c, \varepsilon\}$.
     - Keďže $a, c \in FIRST(A)$, znamená to, že z neterminálu A vieme odvodiť reťazec začínajúcu a, resp. c, teda $A \Rightarrow^* a\alpha$, resp. $A \Rightarrow^* c\gamma$. Potom určite existuje derivácia $ABC \Rightarrow^* a\alpha BC$, resp. $ABC \Rightarrow^* c\gamma BC$, teda aj z ABC vieme odvodiť reťazec začínajúci a, resp. c, a teda $a, c \in FIRST(ABC)$.
     - Ďalej vieme, že $\varepsilon \in FIRST(A)$. V reťazci teda ABC vieme teoreticky prepísať A na ε a platí $ABC \Rightarrow^* BC$, čím dostávame situáciu, že aj $FIRST(B)$ ovplyvňuje $FIRST(ABC)$.
     - Aktuálne vieme, že $FIRST(B) = \{\varepsilon\}$. To znamená, že vieme prepísať neterminál B na ε, a teda existuje derivácia $ABC \Rightarrow^* BC \Rightarrow^* C$, čím dostávame situáciu, že aj $FIRST(C)$ ovplyvňuje $FIRST(ABC)$.
     - Keďže $b \in FIRST(C)$, znamená to, že z neterminálu C vieme odvodiť reťazec začínajúci b. A keďže sme zistili, že z neterminálov A a B vieme v gramatike vyrobiť ε, tak existuje derivácia $ABC \Rightarrow^* BC \Rightarrow^* C \Rightarrow^* b\beta$, teda $b \in FIRST(ABC)$.
     - Ďalej vieme, že $\varepsilon \in FIRST(C)$. Teda v reťazci ABC vieme teoreticky prepísať $ABC \Rightarrow^* BC \Rightarrow^* C \Rightarrow^* \varepsilon$, čím dostávame situáciu, že z celého reťazca ABC vieme odvodiť ε, a teda $\varepsilon \in FIRST(ABC)$.

     Pri aktuálnych hodnotách $FIRST$ sme teda zistili, že $FIRST(ABC) = \{a, b, c, \varepsilon\}$. Keďže $FIRST(ABC) \subseteq FIRST(S)$, do množiny $FIRST(S)$ doplníme všetky tie symboly, ktoré sú vo $FIRST(ABC)$, t. j. po tomto kroku vieme, že $FIRST(S) = \{a, b, c, \varepsilon\}$.

- Pravidlo $A \to C$

  Z tohto pravidla dostávame $FIRST(C) \subseteq FIRST(A)$. V tomto prípade sa pravá strana vyšetrovaného pravidla rovná len neterminálu C, takže priamo môžeme všetky symboly z množiny $FIRST(C)$ pridať do množiny $FIRST(A)$. Do množiny $FIRST(A)$ teda pridáme terminál b, ktorý patrí do $FIRST(C)$ (do $FIRST(C)$ patrí aj ε, ale ten sa už v množine $FIRST(A)$ nachádza), t. j. po tomto kroku $FIRST(A) = \{a, b, c, \varepsilon\}$.

- Pravidlo $B \to BabB$

  Platí, že $FIRST(BabB) \subseteq FIRST(B)$. Na základe aktuálnych hodnôt množín $FIRST$ vyšetríme, akú hodnotu môže mať $FIRST(BabB)$:

  - V reťazci $BabB$ je na prvom mieste neterminál B. Jeho množina $FIRST(B)$ je aktuálne $FIRST(B) = \{\varepsilon\}$, teda v gramatike existuje odvodenie $B \Rightarrow^* \varepsilon$.
  - Z reťazca $BabB$ teda vieme odvodiť $BabB \Rightarrow^* abB$, čiže reťazec začínajúci terminálom a, čiže určite $a \in FIRST(BabB)$.

  Pri aktuálnych hodnotách $FIRST$ sme teda zistili, že $FIRST(BabB) = \{a\}$. Keďže $FIRST(BabB) \subseteq FIRST(B)$, do množiny $FIRST(B)$ doplníme symbol a, teda aktuálne $FIRST(B) = \{a, \varepsilon\}$.

- Pravidlo $B \to AA$

  Platí, že $FIRST(AA) \subseteq FIRST(B)$. Na základe aktuálnych hodnôt množín $FIRST$ vyšetríme, akú hodnotu môže mať $FIRST(AA)$:

  - V reťazci AA je na prvom mieste neterminál A. Jeho množina $FIRST(A)$ je aktuálne $FIRST(A) = \{a, b, c, \varepsilon\}$.
  - Keďže $a, b, c \in FIRST(A)$, znamená to, že z reťazca AA vieme odvodiť reťazce začínajúce a, b, c. Teda určite $a, b, c \in FIRST(AA)$.
  - Ďalej vieme, že $\varepsilon \in FIRST(A)$. Teda v reťazci AA vieme teoreticky prepísať $AA \Rightarrow^* A \Rightarrow^* \varepsilon$, čím dostávame situáciu, že z celého reťazca AA vieme odvodiť ε, a teda $\varepsilon \in FIRST(AA)$.

  Pri aktuálnych hodnotách $FIRST$ sme teda zistili, že $FIRST(AA) = \{a, b, c, \varepsilon\}$. Keďže $FIRST(AA) \subseteq FIRST(B)$, do množiny $FIRST(B)$ doplníme symboly b, c, teda aktuálne $FIRST(B) = \{a, b, c, \varepsilon\}$.

- Tým sme prešli všetky pravidlá gramatiky, ktoré začínajú neterminálom. Keďže počas práce s týmito pravidlami sme pracovali s **vtedy aktuálnymi množinami** $FIRST$, ktoré sa postupne menili (pribúdali do nich nové symboly), mali by sme celý proces zopakovať, pretože množiny $FIRST$ sa nám priebežne rozšírili.

- V tejto gramatike však nemá zmysel pravidlá vyšetrovať znovu, pretože neterminály S, A, B už určite nemôžu obsahovať žiaden ďalší symbol v množine $FIRST$ a neterminál C nemá na pravej strane pravidlo začínajúce neterminálom.

Výsledné množiny $FIRST$ pre neterminály gramatiky teda sú:

- $FIRST(S) = \{\varepsilon, a, b, c\}$,
- $FIRST(A) = \{\varepsilon, a, b, c\}$,
- $FIRST(B) = \{\varepsilon, a, b, c\}$,
- $FIRST(C) = \{\varepsilon, b\}$.

Úloha č. 4.4.2 Je daná redukovaná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

- $S \to AbBB$
- $A \to CC \mid cSA$
- $B \to aCa \mid bb \mid \varepsilon$
- $C \to \varepsilon \mid BC$

Nájdite množinu $FIRST$ pre jednotlivé neterminály gramatiky.

Riešenie:
Pre túto gramatiku:

1. Množina N_ε tejto gramatiky je $N_\varepsilon = \{A, B, C\}$, pozri príklad č. 4.3.2.

   Po tomto kroku teda $FIRST(S) = \emptyset, FIRST(A) = \{\varepsilon\}$, $FIRST(B) = \{\varepsilon\}$, $FIRST(C) = \{\varepsilon\}$.

2. Na základe pravidiel, ktorých prvý symbol na pravej strane je terminál:

   - z pravidla $A \to cSA$ vieme, že $c \in FIRST(A)$,
   - z pravidiel $B \to aCa \mid bb$ vieme, že $a, b \in FIRST(B)$.

   Po tomto kroku teda

   - $FIRST(S) = \emptyset$,
   - $FIRST(A) = \{\varepsilon, c\}$,
   - $FIRST(B) = \{\varepsilon, a, b\}$,
   - $FIRST(C) = \{\varepsilon\}$.

3. V ďalšom kroku budeme postupne prechádzať pravidlá, ktoré začínajú neterminálom a potenciálne pridávať nové terminály do množín $FIRST$ na základe ich aktuálnych hodnôt:

   - Pravidlo $S \to AbBB$

     Tu dostávame, že $FIRST(AbBB) \subseteq FIRST(S)$. Na základe aktuálnych hodnôt množín $FIRST$ vyšetríme, akú hodnotu môže mať $FIRST(AbBB)$:

- V reťazci $AbBB$ je na prvom mieste neterminál A. Jeho množina $FIRST(A)$ je aktuálne $FIRST(A) = \{c, \varepsilon\}$.
- Keďže $c \in FIRST(A)$, znamená to, že z neterminálu A vieme odvodiť $A \Rightarrow^* c\gamma$. Keďže A je prvý symbol reťazca $AbBB$, existuje derivácia $AbBB \Rightarrow^* c\gamma bBB$, teda určite $c \in FIRST(AbBB)$.
- Ďalej vieme, že $\varepsilon \in FIRST(A)$. Teda v reťazci $AbBB$ vieme teoreticky prepísať A na ε a teda $AbBB \Rightarrow^* bBB$, čím dostávame $b \in FIRST(AbBB)$.
- Keďže iné symboly aktuálne v množine $FIRST(A)$ nemáme, v tejto iterácii sme zistili, že $c, b \in FIRST(AbBB)$.

Pri aktuálnych hodnotách $FIRST$ máme $FIRST(AbBB) = \{b, c\}$. Keďže $FIRST(AbBB) \subseteq FIRST(S)$, do množiny $FIRST(S)$ pridáme b aj c, t. j. $FIRST(S) = \{b, c\}$.

- Pravidlo $A \to CC$

  Tu dostávame, že $FIRST(CC) \subseteq FIRST(A)$. Na základe aktuálnych hodnôt množín $FIRST$:

  - V reťazci CC je na prvom mieste neterminál C. Jeho množina $FIRST(C)$ je aktuálne $FIRST(C) = \{\varepsilon\}$.
  - Keďže zatiaľ vieme len toľko, že $FIRST(C)$ obsahuje ε, tak vieme, že neterminál C je možné prepísať na prázdny reťazec. Teda z reťazca CC vieme odvodiť $CC \Rightarrow^* C \Rightarrow^* \varepsilon$, čiže o množine $FIRST(CC)$ vieme zatiaľ povedať len toľko, že obsahuje ε.

  Pri aktuálnych hodnotách $FIRST$ máme $FIRST(CC) = \{\varepsilon\}$. Keďže $FIRST(CC) \subseteq FIRST(A)$, do množiny $FIRST(A)$ by sme pridali ε, avšak keďže ho tam už máme, v tomto kroku do $FIRST(A)$ nepribudne nič nové.

- Pravidlo $C \to BC$

  Tu dostávame, že $FIRST(BC) \subseteq FIRST(C)$. Na základe aktuálnych hodnôt množín $FIRST$ vyšetríme, akú hodnotu môže mať $FIRST(BC)$:

  - V reťazci BC je na prvom mieste neterminál B. Jeho množina $FIRST(B)$ je aktuálne $FIRST(B) = \{a, b, \varepsilon\}$.
  - Keďže $a, b \in FIRST(B)$, znamená to, že z neterminálu B vieme odvodiť $A \Rightarrow^* a\alpha$, resp. $B \Rightarrow^* b\beta$. Keďže B je prvý symbol reťazca BC, existuje derivácia $BC \Rightarrow^* a\alpha C$, resp. $BC \Rightarrow^* b\beta C$ teda určite $a, b \in FIRST(BC)$.
  - Ďalej vieme, že $\varepsilon \in FIRST(B)$. Teda v reťazci BC vieme teoreticky prepísať B na ε a teda $BC \Rightarrow^* C$, čím dostávame, že aj $FIRST(C)$ ovplyvňuje $FIRST(BC)$. Keďže $FIRST(C) = \{\varepsilon\}$, tak vidíme, že je možná derivácia $BC \Rightarrow^* C \Rightarrow^* \varepsilon$, a teda $\varepsilon \in FIRST(BC)$.

Pri aktuálnych hodnotách $FIRST$ máme $FIRST(BC) = \{a, b, \varepsilon\}$. Keďže $FIRST(BC) \subseteq FIRST(C)$, do množiny $FIRST(C)$ pridáme a aj b, t. j. $FIRST(C) = \{a, b, \varepsilon\}$.

Všimnite si, že nám nastala situácia, že pri vyšetrovaní pravidiel sme pracovali s takými množinami $FIRST$, ktoré sa neskôr zmenili, t. j. došlo k ich rozšíreniu. Napríklad pri vyšetrovaní pravidla $A \to CC$ sme pracovali s množinou $FIRST(C) = \{\varepsilon\}$, avšak neskôr došlo k jej rozšíreniu na $FIRST(C) = \{\varepsilon, a, b\}$. Preto musíme znovu prejsť cez všetky pravidlá s neterminálom na začiatku a proces zopakovať, pretože v novom prechode cez pravidlá budeme pracovať s aktuálnejšími hodnotami množín $FIRST$, čo môže spôsobiť, že získame znovu nové symboly do niektorých množín $FIRST$.

Aktuálne množiny $FIRST$ pre neterminály gramatiky teda sú:

- $FIRST(S) = \{b, c\}$,
- $FIRST(A) = \{\varepsilon, c\}$,
- $FIRST(B) = \{\varepsilon, a, b\}$,
- $FIRST(C) = \{\varepsilon, a, b\}$.

4. V ďalšom kroku budeme znovu prechádzať pravidlá, ktoré začínajú neterminálom:

   - Pravidlo $S \to AbBB$
     - Keďže množina $FIRST(A)$ sa nezmenila od posledného vyšetrovania tohto pravidla, nič nové tu nedostávame, t. j. $FIRST(AbBB) = \{b, c\}$.

     Pri aktuálnych hodnotách $FIRST$ máme $FIRST(AbBB) = \{b, c\}$. Keďže $FIRST(AbBB) \subseteq FIRST(S)$, do množiny $FIRST(S)$ nepribudne v tomto kroku nič, $FIRST(S) = \{b, c\}$.
   - Pravidlo $A \to CC$
     - Množina $FIRST(C)$ sa zmenila oproti poslednému vyšetrovaniu tohto pravidla, konkrétne do nej pribudli symboly a, b, teda vieme, že $C \Rightarrow^* a\alpha$, resp. $C \Rightarrow^* b\beta$, teda aj $CC \Rightarrow^* a\alpha C$, resp. $CC \Rightarrow^* b\beta C$, čiže $a, b \in FIRST(CC)$ sú nové informácie.
     - Taktiež platí (ale to sme vedeli už v minulom kole), že keďže $\varepsilon \in FIRST(C)$, tak $CC \Rightarrow^* C \Rightarrow^* \varepsilon$, teda $\varepsilon \in FIRST(CC)$.

     Pri aktuálnych hodnotách $FIRST$ máme $FIRST(CC) = \{\varepsilon, a, b\}$. Keďže $FIRST(CC) \subseteq FIRST(A)$, do množiny $FIRST(A)$ teda pridáme symboly a a b a po tomto kroku $FIRST(A) = \{\varepsilon, a, b, c\}$.
   - Pravidlo $C \to BC$
     - V reťazci BC je na prvom mieste neterminál B. Jeho množina $FIRST(B)$ je aktuálne $FIRST(B) = \{a, b, \varepsilon\}$.

  - Keďže $a, b \in FIRST(B)$, znamená to, že z neterminálu B vieme odvodiť $B \Rightarrow^* a\alpha$, resp. $B \Rightarrow^* b\beta$. Keďže B je prvý symbol reťazca BC, existuje derivácia $BC \Rightarrow^* a\alpha C$, resp. $BC \Rightarrow^* b\beta C$ teda určite $a, b \in FIRST(BC)$.
  - Ďalej vieme, že $\varepsilon \in FIRST(B)$. Teda v reťazci BC vieme teoreticky prepísať B na ε a teda $BC \Rightarrow^* C$, čím dostávame, že aj $FIRST(C)$ ovplyvňuje $FIRST(BC)$. Keďže $FIRST(C) = \{\varepsilon, a, b\}$, tak vidíme, že je možná derivácia $BC \Rightarrow^* C \Rightarrow^* \varepsilon$, a teda $\varepsilon \in FIRST(BC)$. Taktiež, keďže $a, b \in FIRST(C)$ je možné urobiť derivácie $BC \Rightarrow^* C \Rightarrow^* a\alpha$, resp. $BC \Rightarrow^* C \Rightarrow^* b\beta$, čiže $a, b \in FIRST(BC)$, čo však už vieme.

  Pri aktuálnych hodnotách $FIRST$ máme $FIRST(BC) = \{a, b, \varepsilon\}$. Keďže takúto informáciu sme mali už v predchádzajúcom vyšetrovaní tohto pravidla, nič nové sme nezistili.

Znovu nastala situácia, že sa nám zmenila niektorá z množín $FIRST$, konkrétne $FIRST(A)$.

Aktuálne množiny $FIRST$ pre neterminály gramatiky teda sú:

- $FIRST(S) = \{b, c\}$,
- $FIRST(A) = \{\varepsilon, a, b, c\}$,
- $FIRST(B) = \{\varepsilon, a, b\}$,
- $FIRST(C) = \{\varepsilon, a, b\}$.

5. Keďže v poslednom prechode pravidlami nastala zmena v niektorej množine $FIRST$, znovu budeme prechádzať pravidlá, ktoré začínajú neterminálom:

- Pravidlo $S \to AbBB$
  - Keďže množina $FIRST(A)$ sa v poslednom kroku zmenila, po novom vieme, že $FIRST(AbBB) = \{a, b, c\}$.

  Pri aktuálnych hodnotách $FIRST$ máme $FIRST(AbBB) = \{a, b, c\}$. Keďže $FIRST(AbBB) \subseteq FIRST(S)$, do množiny $FIRST(S)$ pribudne v tomto kroku terminál a, $FIRST(S) = \{a, b, c\}$.
- Pravidlo $A \to CC$

  Tu nepribudne žiadna nová informácia, $FIRST(A) = \{\varepsilon, a, b, c\}$.
- Pravidlo $C \to BC$

  Tu nepribudne žiadna nová informácia, $FIRST(C) = \{\varepsilon, a, b\}$

Znovu nastala situácia, že sa nám zmenila niektorá z množín $FIRST$, konkrétne $FIRST(S)$.

Aktuálne množiny $FIRST$ pre neterminály gramatiky teda sú:

- $FIRST(S) = \{a, b, c\}$,
- $FIRST(A) = \{\varepsilon, a, b, c\}$,
- $FIRST(B) = \{\varepsilon, a, b\}$,
- $FIRST(C) = \{\varepsilon, a, b\}$.

6. Keďže v poslednom prechode pravidlami nastala zmena v niektorej množine $FIRST$, znovu by sme mali prechádzať jednotlivé pravidlá a vyšetrovať ich. Avšak v tomto prechode už žiadne nové symboly nepribudnú, teda výsledné množiny $FIRST$ pre jednotlivé neterminály gramatiky sú:

   - $FIRST(S) = \{a, b, c\}$,
   - $FIRST(A) = \{\varepsilon, a, b, c\}$,
   - $FIRST(B) = \{\varepsilon, a, b\}$,
   - $FIRST(C) = \{\varepsilon, a, b\}$.

Úloha č. 4.4.3 Je daná bezkontextová gramatika $G = (\{\text{<blok>}, \text{<prikazy>}, \text{<prikaz>}\}, \{\texttt{p}, \texttt{;}, \texttt{begin}, \texttt{end}\}, P, \text{<blok>})$, ktorej pravidlá P sú:

- <blok> $\rightarrow$ `begin` <prikazy> `end`
- <prikazy> $\rightarrow$ <prikaz> `;` <prikazy> | ε
- <prikaz> $\rightarrow$ <blok> | `p` | ε

Nájdite množiny $FIRST$ pre neterminály gramatiky.

Riešenie:

1. V tejto gramatike $N_\varepsilon = \{\text{<prikazy>}, \text{<prikaz>}\}$, teda po tomto kroku:

   - $FIRST(\text{<blok>}) = \emptyset$,
   - $FIRST(\text{<prikazy>}) = \{\varepsilon\}$,
   - $FIRST(\text{<prikaz>}) = \{\varepsilon\}$,

2. Z pravidiel, ktorých prvý symbol na pravej strane je terminál:

   - z pravidla <blok> $\rightarrow$ `begin` <prikazy> `end` vieme, že `begin` $\in FIRST(\text{<blok>})$,
   - z pravidla <prikaz> $\rightarrow$ `p` vieme, že `p` $\in FIRST(\text{<prikaz>})$.

   Po tomto kroku:

   - $FIRST(\text{<blok>}) = \{\texttt{begin}\}$,
   - $FIRST(\text{<prikazy>}) = \{\varepsilon\}$,

- $FIRST$(<prikaz>) = $\{\varepsilon, \texttt{p}\}$.

3. Z pravidiel, ktorých prvý symbol na pravej strane je neterminál:

   - <prikazy> → <prikaz> ; <prikazy>
     - Pri súčasnom stave množín $FIRST$ máme $FIRST$(<prikaz> ; <prikazy>) = {`p`, `;`}.
     - Teda do množiny $FIRST$(<prikazy>) budú patriť aj terminály `p` a `;`
   - <prikaz> → <blok>
     - Pri súčasnom stave množín $FIRST$ máme $FIRST$(<blok>) = {`begin`}.
     - Teda do množiny $FIRST$(<prikaz>) bude patriť aj terminál `begin`.

   Po tomto kroku:

   - $FIRST$(<blok>) = {`begin`},
   - $FIRST$(<prikazy>) = {ε, `p`, `;`},
   - $FIRST$(<prikaz>) = {ε, `p`, `begin`}.

4. Keďže v poslednej iterácii došlo k zmene množín $FIRST$, znovu opakujeme prechod pravidlami:

   - <prikazy> → <prikaz> ; <prikazy>
     - Pri súčasnom stave množín $FIRST$ máme $FIRST$(<prikaz> ; <prikazy>) = {`p`, `;`, `begin`}.
     - Teda do množiny $FIRST$(<prikazy>) bude patriť aj terminál `begin`, ktorý sme tam doteraz nemali.
   - <prikaz> → <blok>
     - Pri súčasnom stave množín $FIRST$ máme $FIRST$(<blok>) = {`begin`}.
     - Teda tu sme nezistili nič nové, keďže to, že `begin` patrí do $FIRST$(<prikaz>) už vieme.

   Po tomto kroku:

   - $FIRST$(<blok>) = {`begin`},
   - $FIRST$(<prikazy>) = {ε, `p`, `;`, `begin`},
   - $FIRST$(<prikaz>) = {ε, `p`, `begin`}.

5. Keďže v poslednej iterácii došlo k zmene množín $FIRST$, znovu opakujeme prechod pravidlami, avšak v tomto prípade by sme už žiadnu množinu $FIRST$ nerozšírili o nové symboly a hľadanie množín $FIRST$ by skončilo. Výsledné množiny $FIRST$ v tejto gramatike:

- $FIRST(\text{<blok>}) = \{\texttt{begin}\}$,
- $FIRST(\text{<prikazy>}) = \{\varepsilon, \texttt{p}, \texttt{;}, \texttt{begin}\}$,
- $FIRST(\text{<prikaz>}) = \{\varepsilon, \texttt{p}, \texttt{begin}\}$.

Úloha č. 4.4.4 Je daná bezkontextová gramatika $G = (\{S, A, B, C, D\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

- $S \to \varepsilon \mid BBA$
- $A \to Bc$
- $B \to SD \mid ABb$
- $C \to aCD \mid a$
- $D \to Cb \mid \varepsilon$

Nájdite množiny $FIRST$ pre neterminály gramatiky.

Riešenie:

1. V tejto gramatike platí, že $N_\varepsilon = \{D, S, B\}$, teda po tomto kroku:
   - $FIRST(S) = \{\varepsilon\}$,
   - $FIRST(A) = \emptyset$,
   - $FIRST(B) = \{\varepsilon\}$,
   - $FIRST(C) = \emptyset$,
   - $FIRST(D) = \{\varepsilon\}$,
2. Z pravidiel, ktorých prvý symbol na pravej strane je terminál:
   - z pravidiel $C \to aCD$, resp. $C \to a$ vieme, že $a \in FIRST(C)$,

   Po tomto kroku:
   - $FIRST(S) = \{\varepsilon\}$,
   - $FIRST(A) = \emptyset$,
   - $FIRST(B) = \{\varepsilon\}$,
   - $FIRST(C) = \{a\}$,
   - $FIRST(D) = \{\varepsilon\}$,
3. Z pravidiel, ktorých prvý symbol na pravej strane je neterminál:
   - $S \to BBA$

- Pri súčasnom stave množín $FIRST$ máme $FIRST(BBA) = \emptyset$, pretože $FIRST(A)$ je aktuálne len prázdna množina.
- Keďže $FIRST(BBA) \subseteq FIRST(S)$, do množiny $FIRST(S)$ nám v tomto kroku nepribudne nič.

- $A \to Bc$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(Bc) = \{c\}$.
  - Do množiny $FIRST(A)$ nám v tomto kroku pribudne c, $FIRST(A) = \{c\}$.
- $B \to SD$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(SD) = \{\varepsilon\}$.
  - Do množiny $FIRST(B)$ nám na základe tohto pravidla nepribudne nič.
- $B \to ABb$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(ABb) = \{c\}$, pretože $FIRST(A) = \{c\}$.
  - Do množiny $FIRST(B)$ nám na základe tohto pravidla pribudne c, $FIRST(B) = \{\varepsilon, c\}$.
- $D \to Cb$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(Cb) = \{a\}$, pretože $FIRST(C) = \{a\}$.
  - Do množiny $FIRST(D)$ nám na základe tohto pravidla pribudne a, $FIRST(D) = \{\varepsilon, a\}$.

Po tomto kroku:

- $FIRST(S) = \{\varepsilon\}$,
- $FIRST(A) = \{c\}$,
- $FIRST(B) = \{\varepsilon, c\}$,
- $FIRST(C) = \{a\}$,
- $FIRST(D) = \{\varepsilon, a\}$,

4. Keďže v poslednej iterácii došlo k zmene množín $FIRST$, znovu opakujeme prechod pravidlami, ktorých pravá strana začína neterminálom.

   - $S \to BBA$
     - Pri súčasnom stave množín $FIRST$ máme $FIRST(BBA) = \{c\}$, dokonca z viacerých dôvodov:
       (a) Keďže $FIRST(B)$ obsahuje c, tak aj $FIRST(BBA)$ obsahuje c.
       (b) Keďže $FIRST(B)$ obsahuje ε, tak $BBA \Rightarrow^* BA \Rightarrow^* A$ a keďže $FIRST(A)$ obsahuje c, tak aj z tohto dôvodu $FIRST(BBA)$ obsahuje c.

  - Do množiny $FIRST(S)$ teda pribudne c, $FIRST(S) = \{\varepsilon, c\}$.
- $A \to Bc$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(Bc) = \{c\}$.
  - Do množiny $FIRST(A)$ nám v tomto kroku nepribudne nič.
- $B \to SD$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(SD) = \{\varepsilon, a, c\}$.
  - Do množiny $FIRST(B)$ nám na základe tohto pravidla pribudne terminál a, $FIRST(B) = \{\varepsilon, a, c\}$.
- $B \to ABb$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(ABb) = \{c\}$, pretože $FIRST(A) = \{c\}$.
  - Do množiny $FIRST(B)$ nám na základe tohto pravidla nepribudne nič.
- $D \to Cb$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(Cb) = \{a\}$.
  - Do množiny $FIRST(D)$ nám na základe tohto pravidla nepribudne nič.

Po tomto kroku:

- $FIRST(S) = \{\varepsilon, c\}$,
- $FIRST(A) = \{c\}$,
- $FIRST(B) = \{\varepsilon, a, c\}$,
- $FIRST(C) = \{a\}$,
- $FIRST(D) = \{\varepsilon, a\}$,

5. Keďže v poslednej iterácii došlo k zmene množín $FIRST$, znovu opakujeme prechod pravidlami, ktorých pravá strana začína neterminálom.
   - $S \to BBA$
     - Pri súčasnom stave množín $FIRST$ máme $FIRST(BBA) = \{a, c\}$.
     - Do množiny $FIRST(S)$ teda pribudne a, $FIRST(S) = \{\varepsilon, a, c\}$.
   - $A \to Bc$
     - Pri súčasnom stave množín $FIRST$ máme $FIRST(Bc) = \{a, c\}$.
     - Do množiny $FIRST(A)$ nám v tomto kroku pribudne terminál a, $FIRST(A) = \{a, c\}$.
   - $B \to SD$
     - Pri súčasnom stave množín $FIRST$ máme $FIRST(SD) = \{\varepsilon, a, c\}$.
     - Do množiny $FIRST(B)$ nám na základe tohto pravidla nepribudne nič.

- $B \to ABb$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(ABb) = \{a, c\}$.
  - Do množiny $FIRST(B)$ nám na základe tohto pravidla nepribudne nič.
- $D \to Cb$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(Cb) = \{a\}$.
  - Do množiny $FIRST(D)$ nám na základe tohto pravidla nepribudne nič.

Po tomto kroku:

- $FIRST(S) = \{\varepsilon, a, c\}$,
- $FIRST(A) = \{a, c\}$,
- $FIRST(B) = \{\varepsilon, a, c\}$,
- $FIRST(C) = \{a\}$,
- $FIRST(D) = \{\varepsilon, a\}$,

6. Keďže v poslednej iterácii došlo k zmene množín $FIRST$, znovu by sme zopakovali prechod cez pravidlá, avšak v tejto iterácii by sme už nové symboly do množín $FIRST$ nepridali, teda výsledné množiny $FIRST$ v tejto gramatike sú:

   - $FIRST(S) = \{\varepsilon, a, c\}$,
   - $FIRST(A) = \{a, c\}$,
   - $FIRST(B) = \{\varepsilon, a, c\}$,
   - $FIRST(C) = \{a\}$,
   - $FIRST(D) = \{\varepsilon, a\}$.

Úloha č. 4.4.5 Je daná bezkontextová gramatika $G = (\{$<Regex>, <Term>, <Factor>, <Base>, <Char>$\}, \{a, b, !, +, *, (,)\}, P,$ <Regex>$)$, ktorej pravidlá P sú:

- <Regex> $\to$ <Term> | <Term> + <Regex>
- <Term> $\to$ <Factor> | <Factor><Term>
- <Factor> $\to$ <Base> | <Base>$*$
- <Base> $\to$ <Char> | (<Regex>)
- <Char> $\to a \mid b \mid\ !$

Nájdite množiny $FIRST$ pre neterminály gramatiky.

Riešenie:

1. V tejto gramatike $N_\varepsilon = \emptyset$, teda po tomto kroku:
   - $FIRST$(<Regex>) $= \emptyset$,
   - $FIRST$(<Term>) $= \emptyset$,
   - $FIRST$(<Factor>) $= \emptyset$,
   - $FIRST$(<Base>) $= \emptyset$,
   - $FIRST$(<Char>) $= \emptyset$.

2. Z pravidiel, ktorých prvý symbol na pravej strane je terminál:
   - z pravidla <Base> $\rightarrow$ (<Regex>) vieme, že terminál (, teda ľavá zátvorka, bude patriť do $FIRST$(<Base>),
   - z pravidiel <Char> $\rightarrow a \mid b \mid\ !$ vieme, že terminály $a, b, !$ budú patriť do $FIRST$(<Char>).

   Po tomto kroku:
   - $FIRST$(<Regex>) $= \emptyset$,
   - $FIRST$(<Term>) $= \emptyset$,
   - $FIRST$(<Factor>) $= \emptyset$,
   - $FIRST$(<Base>) $= \{(\}$,
   - $FIRST$(<Char>) $= \{a, b, !\}$.

3. Z pravidiel, ktorých prvý symbol na pravej strane je neterminál:
   - <Regex> $\rightarrow$ <Term>
   - <Regex> $\rightarrow$ <Term> + <Regex>
     - Pri súčasnom stave množín $FIRST$ máme $FIRST$(<Term>) $= \emptyset$, teda do množiny $FIRST$(<Regex>) nám v tomto kroku z oboch pravidiel nepribudne nič.
   - <Term> $\rightarrow$ <Factor>
   - <Term> $\rightarrow$ <Factor><Term>
     - Pri súčasnom stave množín $FIRST$ máme $FIRST$(<Factor>) $= \emptyset$, teda do množiny $FIRST$(<Term>) nám v tomto kroku z oboch pravidiel nepribudne nič.
   - <Factor> $\rightarrow$ <Base>
   - <Factor> $\rightarrow$ <Base>$*$

- Pri súčasnom stave množín $FIRST$ máme $FIRST(\text{<Base>}) = \{(\}$, resp. $FIRST(\text{<Base>}*) = \{(\}$, teda do množiny $FIRST(\text{<Factor>})$ nám v tomto kroku pribudne ľavá zátvorka, $FIRST(\text{<Factor>}) = \{(\}$.

- $\text{<Base>} \to \text{<Char>}$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(\text{<Char>}) = \{a, b, !\}$, teda do množiny $FIRST(\text{<Base>})$ nám v tomto kroku pribudnú terminály $a, b, !$, $FIRST(\text{<Base>}) = \{(, a, b, !\}$.

Po tomto kroku:

- $FIRST(\text{<Regex>}) = \emptyset$,
- $FIRST(\text{<Term>}) = \emptyset$,
- $FIRST(\text{<Factor>}) = \{(\}$,
- $FIRST(\text{<Base>}) = \{(, a, b, !\}$,
- $FIRST(\text{<Char>}) = \{a, b, !\}$.

4. Keďže v poslednej iterácii došlo k zmene množín $FIRST$, znovu opakujeme prechod pravidlami, ktorých pravá strana začína neterminálom.

5. Z pravidiel, ktorých prvý symbol na pravej strane je neterminál:

- $\text{<Regex>} \to \text{<Term>}$
- $\text{<Regex>} \to \text{<Term>} + \text{<Regex>}$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(\text{<Term>}) = \emptyset$, teda do množiny $FIRST(\text{<Regex>})$ nám v tomto kroku z oboch pravidiel nepribudne nič.
- $\text{<Term>} \to \text{<Factor>}$
- $\text{<Term>} \to \text{<Factor>}\text{<Term>}$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(\text{<Factor>}) = \{(\}$, teda do množiny $FIRST(\text{<Term>})$ nám v tomto kroku pribudne terminál ľavá zátvorka, $FIRST(\text{<Term>}) = \{(\}$.
- $\text{<Factor>} \to \text{<Base>}$
- $\text{<Factor>} \to \text{<Base>}*$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(\text{<Base>}) = \{(, a, b, !\}$, resp. $FIRST(\text{<Base>}*) = \{(, a, b, !\}$, teda do množiny $FIRST(\text{<Factor>})$ nám v tomto kroku pribudnú terminály $a, b, !$, $FIRST(\text{<Factor>}) = \{(, a, b, !\}$.
- $\text{<Base>} \to \text{<Char>}$
  - Pri súčasnom stave množín $FIRST$ máme $FIRST(\text{<Char>}) = \{a, b, !\}$, teda do množiny $FIRST(\text{<Base>})$ nám nepribudne žiaden nový terminál.

Po tomto kroku:

- $FIRST(<\text{Regex}>) = \emptyset$,
- $FIRST(<\text{Term}>) = \{(\}$,
- $FIRST(<\text{Factor}>) = \{(, a, b, !\}$,
- $FIRST(<\text{Base}>) = \{(, a, b, !\}$,
- $FIRST(<\text{Char}>) = \{a, b, !\}$.

6. Keďže v poslednej iterácii došlo k zmene množín $FIRST$, znovu opakujeme prechod pravidlami, ktorých pravá strana začína neterminálom. Bez rozpisovania uvádzame, že po tejto iterácii by bol stav množín $FIRST$:
   - $FIRST(<\text{Regex}>) = \{(\}$,
   - $FIRST(<\text{Term}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Factor}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Base}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Char}>) = \{a, b, !\}$.
7. Keďže v poslednej iterácii došlo znovu k zmene množín $FIRST$, opakujeme prechod pravidlami, ktorých pravá strana začína neterminálom. Bez rozpisovania uvádzame, že po tejto iterácii by bol stav množín $FIRST$:
   - $FIRST(<\text{Regex}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Term}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Factor}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Base}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Char}>) = \{a, b, !\}$.
8. Keďže aj v poslednej iterácii došlo znovu k zmene množín $FIRST$, opakujeme prechod pravidlami, ktorých pravá strana začína neterminálom. V tomto prechode však už k zmene množín nedôjde a výsledný obsah množín $FIRST$ by v tejto gramatike bol:
   - $FIRST(<\text{Regex}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Term}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Factor}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Base}>) = \{(, a, b, !\}$,
   - $FIRST(<\text{Char}>) = \{a, b, !\}$.

4.5 Množina $FOLLOW$

Úloha č. 4.5.1 Je daná redukovaná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

- $S \to ABC \mid aSb$
- $A \to aAb \mid c \mid C$
- $B \to BabB \mid AA$
- $C \to \varepsilon \mid baCab$

Nájdite množinu $FOLLOW$ pre jednotlivé neterminály gramatiky.

Riešenie:
Množinu $FOLLOW$ štandardne určujeme pre redukované gramatiky. Keďže zadaná gramatika už redukovaná je, môžeme pristúpiť k jej hľadaniu.

Množina $FOLLOW(A)$ nejakého neterminálu gramatiky $A \in N$ je definovaná ako:

$$FOLLOW(A) = \{a \in T \mid S \Rightarrow^* \alpha A a \beta\} \cup \{\varepsilon \mid S \Rightarrow^* \gamma A\},$$

kde $\alpha, \beta, \gamma \in (N \cup T)^*$. Teda pre neterminál A je jeho množina $FOLLOW$ tvorená:

1. všetkými terminálmi a, ktoré môžu stáť **hneď za** neterminálom A (t. j. terminály **nasledujú** neterminál A) v nejakej vetnej forme,
2. symbolom ε, ak neterminál A môže stáť **na konci** nejakej vetnej formy.

Hľadanie množiny $FOLLOW$ pre jednotlivé neterminály gramatiky vykonáme v 2 krokoch:

1. V prvom kroku do množiny $FOLLOW(S)$ pre počiatočný neterminál S pridáme ε, pretože S je vždy prvá vetná forma, ktorou začína každá derivácia, teda S určite stojí na konci nejakej vetnej formy.
2. Každé pravidlo, ktoré na pravej strane obsahuje aspoň 1 neterminál, teda tvaru $A \to \alpha B \beta$, kde $A, B \in N; \alpha, \beta \in (N \cup T)^*$, ovplyvňuje množinu $FOLLOW$ neterminálu B, v závislosti na množine $FIRST$ reťazca β, ktorý tvorí zvyšok pravej strany pravidla za neterminálom B:
   - $(FIRST(\beta) \setminus \varepsilon) \subseteq FOLLOW(B)$,
   - Ak $\varepsilon \in FIRST(\beta)$, potom $FOLLOW(A) \subseteq FOLLOW(B)$.

   Ak pravidlo gramatiky na pravej strane obsahuje viacero neterminálov, tak ho treba spracovať pre každý neterminál na pravej strane zvlášť.

Z uvedeného vyplýva, že v prípade vyšetrovania množín $FOLLOW$ potrebujeme poznať aj množiny $FIRST$. Množiny $FIRST$ jednotlivých neterminálov tejto gramatiky sú:

- $FIRST(S) = \{\varepsilon, a, b, c\}$,
- $FIRST(A) = \{\varepsilon, a, b, c\}$,
- $FIRST(B) = \{\varepsilon, a, b, c\}$,
- $FIRST(C) = \{\varepsilon, b\}$.

V prípade danej gramatiky:

1. Do množiny $FOLLOW(S)$ pridáme ε.

   Po tomto kroku:

   - $FOLLOW(S) = \{\varepsilon\}$,
   - $FOLLOW(A) = \emptyset$,
   - $FOLLOW(B) = \emptyset$,
   - $FOLLOW(C) = \emptyset$,

2. V ďalšom kroku budeme postupne prechádzať pravidlá, ktoré na pravej strane obsahujú aspoň 1 neterminál. Každé pravidlo vyšetríme toľkokrát, koľko neterminálov obsahuje na pravej strane — aktuálne uvažovaný neterminál na pravej strane bude v zápise podčiarknutý.

   - Pravidlo $S \to \underline{A}BC$, t. j. vyšetrujeme $FOLLOW(A)$ a $\beta = BC$:
     - Na základe známych množín $FIRST$ vypočítame $FIRST(BC) = \{a, b, c, \varepsilon\}$:
     - Podľa vyššie uvedeného vzťahu musí platiť, že $(FIRST(BC) \setminus \varepsilon) \subseteq FOLLOW(A)$, teda ak z množiny $FIRST(BC) = \{a, b, c, \varepsilon\}$ odstránime prázdny reťazec, výsledné symboly určite patria do $FOLLOW(A)$; $a, b, c \in FOLLOW(A)$.
     - Navyše, ak $\varepsilon \in FIRST(BC)$, čo je náš prípad, tak potom každý symbol $FOLLOW$ neterminálu na ľavej strane pravidla, $FOLLOW(S)$, bude zároveň patriť aj do množiny $FOLLOW$ vyšetrovaného neterminálu, $FOLLOW(A)$. Keďže aktuálne $FOLLOW(S) = \{\varepsilon\}$, tak aj $\varepsilon \in FOLLOW(A)$.
     - Na základe pravidla $S \to ABC$ teda aktuálne máme $FOLLOW(A) = \{\varepsilon, a, b, c\}$.
   - Pravidlo $S \to A\underline{B}C$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = C$:
     - Na základe známych množín $FIRST$ máme $FIRST(C) = \{b, \varepsilon\}$.

  - Podľa vyššie uvedeného vzťahu musí platiť, že $(FIRST(C) \setminus \varepsilon) \subseteq FOLLOW(B)$, teda ak z množiny $FIRST(C) = \{b, \varepsilon\}$ odstránime prázdny reťazec, výsledné symboly určite patria do $FOLLOW(B)$, $b \in FOLLOW(B)$.
  - Navyše, ak $\varepsilon \in FIRST(C)$, čo je náš prípad, tak potom každý symbol $FOLLOW$ neterminálu na ľavej strane pravidla, $FOLLOW(S)$, bude zároveň patriť aj do množiny $FOLLOW$ vyšetrovaného neterminálu, $FOLLOW(B)$. Keďže aktuálne $FOLLOW(S) = \{\varepsilon\}$, tak aj $\varepsilon \in FOLLOW(B)$.
  - Na základe pravidla $S \rightarrow ABC$ teda aktuálne máme $FOLLOW(B) = \{\varepsilon, b\}$.

- Pravidlo $S \rightarrow AB\underline{C}$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = \varepsilon$ (β označuje zvyšok pravej strany pravidla za podčiarknutým neterminálom, v tomto prípade je podčiarknutý neterminál posledným symbolom pravej strany, t. j. $\beta = \varepsilon$):
  - V tomto prípade $FIRST(\beta) = FIRST(\varepsilon) = \{\varepsilon\}$.
  - Teda v tomto prípade $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
  - Keďže $\varepsilon \in FIRST(\beta) = FIRST(\varepsilon)$, tak potom $FOLLOW(S)$ bude zároveň patriť aj do množiny $FOLLOW$ vyšetrovaného neterminálu, $FOLLOW(C)$. Keďže aktuálne $FOLLOW(S) = \{\varepsilon\}$, tak aj $\varepsilon \in FOLLOW(C)$.
  - Na základe pravidla $S \rightarrow ABC$ teda aktuálne máme $FOLLOW(C) = \{\varepsilon\}$.
- Pravidlo $S \rightarrow a\underline{S}b$, t. j. vyšetrujeme $FOLLOW(S)$ a $\beta = b$:
  - V tomto prípade $FIRST(b) = \{b\}$.
  - Teda v tomto prípade $(FIRST(b) \setminus \varepsilon) = \{b\}$ a $b \in FOLLOW(S)$.
  - Na základe pravidla $S \rightarrow aSb$ sme teda do množiny $FOLLOW(S)$ pridali terminál b, teda aktuálne máme $FOLLOW(S) = \{\varepsilon, b\}$.
- Pravidlo $A \rightarrow a\underline{A}b$, t. j. vyšetrujeme $FOLLOW(A)$ a $\beta = b$:
  - V tomto prípade $FIRST(b) = \{b\}$.
  - Teda v tomto prípade $(FIRST(b) \setminus \varepsilon) = \{b\}$ a $b \in FOLLOW(A)$.
  - Na základe pravidla $A \rightarrow aAb$ by sme do množiny $FOLLOW(A)$ pridali terminál b, avšak o ňom už vieme, že tam patrí, takže v tomto kroku nezískame novú informáciu.
- Pravidlo $A \rightarrow \underline{C}$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = \varepsilon$:
  - V tomto prípade $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Teda v tomto prípade $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
  - Keďže $\varepsilon \in FIRST(\beta) = FIRST(\varepsilon)$, tak potom $FOLLOW(A)$ bude zároveň patriť aj do množiny $FOLLOW$ vyšetrovaného neterminálu,

$FOLLOW(C)$. Keďže aktuálne $FOLLOW(A) = \{\varepsilon, a, b, c\}$, tak do množiny $FOLLOW(C)$, ktorá doteraz obsahovala len ε, pridáme terminály a, b, c.
  - Na základe pravidla $A \to C$ teda aktuálne máme $FOLLOW(C) = \{\varepsilon, a, b, c\}$.

- Pravidlo $B \to \underline{B}abB$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = abB$:
  - V tomto prípade $FIRST(abB) = \{a\}$.
  - Teda v tomto prípade $(FIRST(abB) \setminus \varepsilon) = \{a\}$ a teda $a \in FOLLOW(B)$.
  - Na základe pravidla $B \to BabB$ teda aktuálne máme $FOLLOW(B) = \{\varepsilon, b, a\}$.
- Pravidlo $B \to Bab\underline{B}$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = \varepsilon$:
  - V tomto prípade $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Teda v tomto prípade $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
  - Keďže $\varepsilon \in FIRST(\beta) = FIRST(\varepsilon)$, potom by $FOLLOW$ neterminálu na ľavej strane, $FOLLOW(B)$ bol podmnožinou $FOLLOW$ podčiarknutého neterminálu, $FOLLOW(B)$. Avšak v tomto prípade ide o tú istú množinu, t. j. $FOLLOW(B) \subseteq FOLLOW(B)$, teda sa nič nové nedozvieme.
- Pravidlo $B \to \underline{A}A$, t. j. vyšetrujeme $FOLLOW(A)$ a $\beta = A$:
  - V tomto prípade $FIRST(A) = \{\varepsilon, a, b, c\}$.
  - Teda v tomto prípade $(FIRST(\varepsilon) \setminus \varepsilon) = \{a, b, c\}$ a do množiny $FOLLOW(A)$ by sme pridali a, b, c, ktoré sa tam však už nachádzajú, takže nepridáme nič.
  - Keďže $\varepsilon \in FIRST(A)$, potom $FOLLOW(B) \subseteq FOLLOW(A)$. Teda by sme do množiny $FOLLOW(A)$ pridali všetky symboly, ktoré sa aktuálne nachádzajú vo $FOLLOW(B) = \{\varepsilon, a, b\}$. Také symboly tam však už máme.
- Pravidlo $B \to A\underline{A}$, t. j. vyšetrujeme $FOLLOW(A)$ a $\beta = \varepsilon$:
  - V tomto prípade $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Teda v tomto prípade $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$, takže nezistíme nič.
  - Keďže $\varepsilon \in FIRST(\varepsilon)$, potom $FOLLOW(B) \subseteq FOLLOW(A)$. Teda by sme do množiny $FOLLOW(A)$ pridali všetky symboly, ktoré sa aktuálne nachádzajú vo $FOLLOW(B) = \{\varepsilon, a, b\}$. Také symboly tam však už máme.
- Pravidlo $C \to ba\underline{C}ab$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = ab$:
  - V tomto prípade $FIRST(ab) = \{a\}$.
  - Teda v tomto prípade $(FIRST(ab) \setminus \varepsilon) = \{a\}$, a do $FOLLOW(C)$ by sme pridali terminál a, avšak ten tam už máme.

Teda po tomto prechode pravidlami máme aktuálny stav množín $FOLLOW$:

- $FOLLOW(S) = \{\varepsilon, b\}$,
- $FOLLOW(A) = \{\varepsilon, a, b, c\}$,
- $FOLLOW(B) = \{\varepsilon, a, b\}$,
- $FOLLOW(C) = \{\varepsilon, a, b, c\}$,

3. Keďže pri počítaní množín $FOLLOW$ môže nastať situácia, že do množiny $FOLLOW$ aktuálne vyšetrovaného neterminálu vkladáme aktuálne známe symboly množiny $FOLLOW$ iného neterminálu, ak počas prechodu pravidlami dôjde k zmene množín $FOLLOW$, musíme proces prechodu pravidlami zopakovať, pretože sa môžeme v nasledovnej iterácii dozvedieť nové informácie.

4. V tomto prípade by sme sa však žiadnu novú informáciu pri opakovanom prechode pravidlami nedozvedeli a výsledné množiny $FOLLOW$ majú hodnotu:

   - $FOLLOW(S) = \{\varepsilon, b\}$,
   - $FOLLOW(A) = \{\varepsilon, a, b, c\}$,
   - $FOLLOW(B) = \{\varepsilon, a, b\}$,
   - $FOLLOW(C) = \{\varepsilon, a, b, c\}$.

Úloha č. 4.5.2 Je daná redukovaná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

- $S \to AbBB$
- $A \to CC \mid cSA$
- $B \to aCa \mid bb \mid \varepsilon$
- $C \to \varepsilon \mid BC$

Nájdite množinu $FOLLOW$ pre jednotlivé neterminály gramatiky.

Riešenie:
Množiny $FIRST$ jednotlivých neterminálov gramatiky sú:

- $FIRST(S) = \{a, b, c\}$,
- $FIRST(A) = \{\varepsilon, a, b, c\}$,
- $FIRST(B) = \{\varepsilon, a, b\}$,
- $FIRST(C) = \{\varepsilon, a, b\}$.

V prípade danej gramatiky:

1. Do množiny $FOLLOW(S)$ pridáme ε.

Po tomto kroku:

- $FOLLOW(S) = \{\varepsilon\}$,
- $FOLLOW(A) = \emptyset$,
- $FOLLOW(B) = \emptyset$,
- $FOLLOW(C) = \emptyset$,

2. V ďalšom kroku budeme postupne prechádzať pravidlá, ktoré na pravej strane obsahujú aspoň 1 neterminál. Každé pravidlo vyšetríme toľkokrát, koľko neterminálov obsahuje na pravej strane — aktuálne uvažovaný neterminál na pravej strane bude v zápise podčiarknutý.

   - Pravidlo $S \to \underline{A}bBB$, t. j. vyšetrujeme $FOLLOW(A)$ a $\beta = bBB$:
     - $FIRST(bBB) = \{b\}$, teda $b \in FOLLOW(A)$ a priebežne dostávame $FOLLOW(A) = \{b\}$.
   - Pravidlo $S \to Ab\underline{B}B$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = B$:
     - $FIRST(B) = \{\varepsilon, a, b\}$.
     - $(FIRST(B) \setminus \varepsilon) \subseteq FOLLOW(B)$, teda ak z množiny $FIRST(B) = \{\varepsilon, a, b\}$ odstránime prázdny reťazec, výsledné symboly určite patria do $FOLLOW(B)$; $a, b \in FOLLOW(B)$.
     - Navyše, keďže $\varepsilon \in FIRST(B)$, tak potom každý symbol $FOLLOW$ neterminálu na ľavej strane pravidla, $FOLLOW(S)$, bude zároveň patriť aj do množiny $FOLLOW$ vyšetrovaného neterminálu, $FOLLOW(B)$. Keďže aktuálne $FOLLOW(S) = \{\varepsilon\}$, tak aj $\varepsilon \in FOLLOW(B)$.
     - Aktuálne teda dostávame $FOLLOW(B) = \{a, b, \varepsilon\}$.
   - Pravidlo $S \to AbB\underline{B}$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = \varepsilon$:
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$, teda z tohto sa nedozvieme nič.
     - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak potom každý symbol $FOLLOW$ neterminálu na ľavej strane pravidla, $FOLLOW(S)$, bude zároveň patriť aj do množiny $FOLLOW$ vyšetrovaného neterminálu, $FOLLOW(B)$. Keďže aktuálne $FOLLOW(S) = \{\varepsilon\}$, tak aj $\varepsilon \in FOLLOW(B)$, avšak to sme už zistili v predchádzajúcom kroku.
     - Naďalej teda $FOLLOW(B) = \{a, b, \varepsilon\}$.
   - Pravidlo $A \to \underline{C}C$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = C$:
     - $FIRST(C) = \{\varepsilon, a, b\}$.
     - $(FIRST(C) \setminus \varepsilon) = \{a, b\}$, teda do množiny $FOLLOW(C)$ patria a, b.
     - Keďže $\varepsilon \in FIRST(C)$, tak potom každý symbol $FOLLOW$ neterminálu na ľavej strane pravidla, $FOLLOW(A)$, bude zároveň patriť aj do množiny $FOLLOW$ vyšetrovaného neterminálu, $FOLLOW(C)$. Keďže aktuálne $FOLLOW(A) = \{b\}$, tak z tohto vyplýva, že $b \in FOLLOW(C)$, čo sme však už zistili v predchádzajúcom bode.

  - Teda aktuálne máme $FOLLOW(C) = \{a, b\}$.
- Pravidlo $A \to C\underline{C}$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = \varepsilon$:
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
  - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak potom $FOLLOW(A) \subseteq FOLLOW(C)$. Keďže aktuálne $FOLLOW(A) = \{b\}$, tak z tohto vyplýva, že $b \in FOLLOW(C)$, čo tam však už máme.
  - Teraz sme teda nezistili nič nové.
- Pravidlo $A \to c\underline{S}A$, t. j. vyšetrujeme $FOLLOW(S)$ a $\beta = A$:
  - $FIRST(A) = \{\varepsilon, a, b, c\}$.
  - $(FIRST(A) \setminus \varepsilon) = \{a, b, c\}$, teda a, b, c pridáme do množiny $FOLLOW(S)$.
  - Keďže $\varepsilon \in FIRST(A)$, tak potom $FOLLOW(A) \subseteq FOLLOW(S)$. Keďže aktuálne $FOLLOW(A) = \{b\}$, tak z tohto vyplýva, že $b \in FOLLOW(S)$, čo tam však už máme.
  - Po tomto vyšetrení teda máme $FOLLOW(S) = \{\varepsilon, a, b, c\}$.
- Pravidlo $A \to cS\underline{A}$, t. j. vyšetrujeme $FOLLOW(A)$ a $\beta = \varepsilon$:
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
  - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak potom $FOLLOW(A) \subseteq FOLLOW(A)$, čo nám neposkytne novú informáciu.
- Pravidlo $B \to a\underline{C}a$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = a$:
  - $FIRST(a) = \{a\}$.
  - $(FIRST(a) \setminus \varepsilon) = \{a\}$, teda do množiny $FOLLOW(C)$ by sme pridali terminál a, ktorý tam však už máme.
- Pravidlo $C \to \underline{B}C$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = C$:
  - $FIRST(C) = \{\varepsilon, a, b\}$.
  - $(FIRST(C) \setminus \varepsilon) = \{a, b\}$, teda do množiny $FOLLOW(B)$ by sme pridali terminály a, b, ktoré tam však už máme.
  - Keďže $\varepsilon \in FIRST(C)$, tak potom $FOLLOW(C) \subseteq FOLLOW(B)$, čo nám neposkytne novú informáciu, pretože už vieme, že a, b patria do $FOLLOW(B)$.
- Pravidlo $C \to B\underline{C}$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = \varepsilon$:
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
  - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak potom $FOLLOW(C) \subseteq FOLLOW(C)$, čo nám neposkytne novú informáciu.

Teda po tomto prechode pravidlami máme aktuálny stav množín $FOLLOW$:

- $FOLLOW(S) = \{\varepsilon, a, b, c\}$,
- $FOLLOW(A) = \{b\}$,
- $FOLLOW(B) = \{\varepsilon, a, b\}$,
- $FOLLOW(C) = \{a, b\}$,

3. Keďže počas prechodu pravidiel sa množiny $FOLLOW$ zmenili, prechod pravidlami zopakujeme:

   - Pravidlo $S \to \underline{A}bBB$, t. j. vyšetrujeme $FOLLOW(A)$ a $\beta = bBB$:
     - $FIRST(bBB) = \{b\}$, teda $b \in FOLLOW(A)$, čo už vieme.
   - Pravidlo $S \to Ab\underline{B}B$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = B$:
     - $FIRST(B) = \{\varepsilon, a, b\}$.
     - $(FIRST(B) \setminus \varepsilon) \subseteq FOLLOW(B)$, teda $\{a, b\} \subseteq FOLLOW(B)$, čo už vieme.
     - Navyše, keďže $\varepsilon \in FIRST(B)$, tak $FOLLOW(S) \subseteq FOLLOW(B)$. Keďže v množine $FOLLOW(S)$ nastala zmena od momentu, keď sme toto pravidlo spracovávali naposledy a aktuálne $FOLLOW(S) = \{\varepsilon, a, b, c\}$, dostávame, že $\{\varepsilon, a, b, c\} \subseteq FOLLOW(B)$. Teda do $FOLLOW(B)$ doplníme symboly z množiny $\{\varepsilon, a, b, c\}$, ktoré tam ešte nemáme, čiže terminál c.
     - Aktuálne teda dostávame $FOLLOW(B) = \{a, b, c, \varepsilon\}$.
   - Pravidlo $S \to AbB\underline{B}$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = \varepsilon$:
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$, teda z tohto sa nedozvieme nič.
     - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak potom $FOLLOW(S) \subseteq FOLLOW(B)$. Keďže aktuálne $FOLLOW(S) = \{\varepsilon, a, b, c\}$ a aj $FOLLOW(B) = \{\varepsilon, a, b, c\}$, nedostávame nič nové.
   - Pravidlo $A \to \underline{C}C$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = C$:
     - $FIRST(C) = \{\varepsilon, a, b\}$.
     - $(FIRST(C) \setminus \varepsilon) = \{a, b\}$, teda do množiny $FOLLOW(C)$ patria a, b, čo už vieme.
     - Keďže $\varepsilon \in FIRST(C)$, tak potom $FOLLOW(A) \subseteq FOLLOW(C)$, čo nám nič nové v tomto kroku nepridá.
   - Pravidlo $A \to C\underline{C}$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = \varepsilon$:
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
     - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak $FOLLOW(A) \subseteq FOLLOW(C)$, čo nám nič nové nepridá.

- Pravidlo $A \to c\underline{S}A$, t. j. vyšetrujeme $FOLLOW(S)$ a $\beta = A$:
  - $FIRST(A) = \{\varepsilon, a, b, c\}$.
  - $(FIRST(A) \setminus \varepsilon) = \{a, b, c\}$, teda $a, b, c \in FOLLOW(S)$, čo už vieme.
  - Keďže $\varepsilon \in FIRST(A)$, tak potom $FOLLOW(A) \subseteq FOLLOW(S)$, čo nám nič nové nepridá.
- Pravidlo $A \to cS\underline{A}$, t. j. vyšetrujeme $FOLLOW(A)$ a $\beta = \varepsilon$:
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
  - Keďže $\varepsilon \in FIRST(\varepsilon)$, dostávame $FOLLOW(A) \subseteq FOLLOW(A)$, čo nám nič nové nepridá.
- Pravidlo $B \to a\underline{C}a$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = a$:
  - $FIRST(a) = \{a\}$.
  - $(FIRST(a) \setminus \varepsilon) = \{a\}$, teda do množiny $FOLLOW(C)$ by sme pridali terminál a, ktorý tam však už máme.
- Pravidlo $C \to \underline{B}C$, t. j. vyšetrujeme $FOLLOW(B)$ a $\beta = C$:
  - $FIRST(C) = \{\varepsilon, a, b\}$.
  - $(FIRST(C) \setminus \varepsilon) = \{a, b\}$, teda do množiny $FOLLOW(B)$ by sme pridali terminály a, b, ktoré tam však už máme.
  - Keďže $\varepsilon \in FIRST(C)$, tak potom $FOLLOW(C) \subseteq FOLLOW(B)$, čo nám nič nové nepridá, pretože a, b v množine $FOLLOW(B)$ už máme.
- Pravidlo $C \to B\underline{C}$, t. j. vyšetrujeme $FOLLOW(C)$ a $\beta = \varepsilon$:
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - $(FIRST(\varepsilon) \setminus \varepsilon) = \emptyset$.
  - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak potom $FOLLOW(C) \subseteq FOLLOW(C)$, čo nám neposkytne novú informáciu.

Teda po tomto druhom prechode pravidlami máme aktuálny stav množín $FOLLOW$:

- $FOLLOW(S) = \{\varepsilon, a, b, c\}$,
- $FOLLOW(A) = \{b\}$,
- $FOLLOW(B) = \{\varepsilon, a, b, c\}$,
- $FOLLOW(C) = \{a, b\}$,

4. Keďže sa nám počas druhého prechodu pravidlami zmenila množina *FOLLOW* neterminálu B, mali by sme znovu prechod pravidlami zopakovať. V tomto prípade však už nič nové nedostaneme, čím dostávame výsledné množiny *FOLLOW*:
   - $FOLLOW(S) = \{\varepsilon, a, b, c\}$,
   - $FOLLOW(A) = \{b\}$,
   - $FOLLOW(B) = \{\varepsilon, a, b, c\}$,
   - $FOLLOW(C) = \{a, b\}$.

Úloha č. 4.5.3 Je daná bezkontextová gramatika $G = (\{\text{<blok>}, \text{<prikazy>}, \text{<prikaz>}\}, \{\texttt{p}, \texttt{;}, \texttt{begin}, \texttt{end}\}, P, \text{<blok>})$, ktorej pravidlá P sú:

- $\text{<blok>} \rightarrow \texttt{begin}\ \text{<prikazy>}\ \texttt{end}$
- $\text{<prikazy>} \rightarrow \text{<prikaz>}\ \texttt{;}\ \text{<prikazy>} \mid \varepsilon$
- $\text{<prikaz>} \rightarrow \text{<blok>} \mid \texttt{p} \mid \varepsilon$

Nájdite množiny *FOLLOW* pre neterminály gramatiky.

Riešenie:
Množiny *FIRST* jednotlivých neterminálov gramatiky sú:

- $FIRST(\text{<blok>}) = \{\texttt{begin}\}$,
- $FIRST(\text{<prikazy>}) = \{\varepsilon, \texttt{p}, \texttt{;}, \texttt{begin}\}$,
- $FIRST(\text{<prikaz>}) = \{\varepsilon, \texttt{p}, \texttt{begin}\}$.

V danej gramatike:

1. Do množiny $FOLLOW(\text{<blok>})$ pridáme ε, keďže <blok> je počiatočný neterminál.

   Po tomto kroku:
   - $FOLLOW(\text{<blok>}) = \{\varepsilon\}$,
   - $FOLLOW(\text{<prikazy>}) = \emptyset$,
   - $FOLLOW(\text{<prikaz>}) = \emptyset$,
2. Prvý prechod pravidlami:
   - Pravidlo $\text{<blok>} \rightarrow \texttt{begin}\underline{\text{<prikazy>}}\texttt{end}$, t. j. vyšetrujeme $FOLLOW(\text{<prikazy>})$ a $\beta = \texttt{end}$:
     - $FIRST(\texttt{end}) = \{\texttt{end}\}$, teda $\texttt{end} \in FOLLOW(\text{<prikazy>})$ a priebežne dostávame $FOLLOW(\text{<prikazy>}) = \{\texttt{end}\}$.

- Pravidlo <prikazy> $\rightarrow$ <prikaz> ; <prikazy>, t. j. vyšetrujeme $FOLLOW$(<prikaz>) a $\beta =$; <prikazy>
  - $FIRST$(;<prikazy>) = {;}, teda symbol ; (bodkočiarka) patrí do množiny $FOLLOW$(<prikaz>) a priebežne dostávame $FOLLOW$(<prikaz>) = {;}.
- Pravidlo <prikazy> $\rightarrow$ <prikaz> ; <prikazy>, t. j. vyšetrujeme $FOLLOW$(<prikazy>) a $\beta = \varepsilon$
  - $FIRST(\varepsilon) = \{\varepsilon\}$, teda $(FIRST(\beta) \setminus \varepsilon) \subseteq FOLLOW$(<prikazy>) z čoho nedostávame žiadnu informáciu.
  - Keďže $\varepsilon \in FIRST(\beta)$, platí $FOLLOW$(<prikazy>) $\subseteq FOLLOW$(<prikazy>), z čoho sa určite nedozvieme nič nové.
- Pravidlo <prikaz> $\rightarrow$ <blok>, vyšetrujeme $FOLLOW$(<blok>) a $\beta = \varepsilon$:
  - $FIRST(\varepsilon) = \{\varepsilon\}$, teda $(FIRST(\beta) \setminus \varepsilon) \subseteq FOLLOW$(<blok>), z čoho nedostávame žiadnu informáciu.
  - Keďže $\varepsilon \in FIRST(\beta)$, potom $FOLLOW$(<prikaz>) = {;} $\subseteq FOLLOW$(<blok>), teda vieme, že symbol ; (bodkočiarka) patrí do množiny $FOLLOW$(<blok>), čím dostávame $FOLLOW$(<blok>) = {ε, ;}.

Teda po tomto prechode pravidlami máme aktuálny stav množín $FOLLOW$:

- $FOLLOW$(<blok>) = {ε, ;},
- $FOLLOW$(<prikazy>) = {`end`},
- $FOLLOW$(<prikaz>) = {;},

3. Keďže sa nám počas prechodu pravidlami zmenili množiny $FOLLOW$ neterminálov, mali by sme znovu prechod pravidlami zopakovať'. V tomto prípade však už nič nové nedostaneme, čím dostávame výsledné množiny $FOLLOW$:

   - $FOLLOW$(<blok>) = {ε, ;},
   - $FOLLOW$(<prikazy>) = {`end`},
   - $FOLLOW$(<prikaz>) = {;}.

Úloha č. 4.5.4 Je daná bezkontextová gramatika $G = (\{S, A, B, C, D\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

- $S \to \varepsilon \mid BBA$
- $A \to Bc$
- $B \to SD \mid ABb$
- $C \to aCD \mid a$
- $D \to Cb \mid \varepsilon$

Nájdite množiny $FOLLOW$ pre neterminály gramatiky.

Riešenie: Množiny $FIRST$ jednotlivých neterminálov gramatiky sú:

- $FIRST(S) = \{\varepsilon, a, c\}$,
- $FIRST(A) = \{a, c\}$,
- $FIRST(B) = \{\varepsilon, a, c\}$.
- $FIRST(C) = \{a\}$,
- $FIRST(D) = \{\varepsilon, a\}$.

V danej gramatike:

1. Do množiny $FOLLOW(S)$ pridáme ε, keďže S je počiatočný neterminál.

   Po tomto kroku:

   - $FOLLOW(S) = \{\varepsilon\}$,
   - $FOLLOW(A) = \emptyset$,
   - $FOLLOW(B) = \emptyset$,
   - $FOLLOW(C) = \emptyset$,
   - $FOLLOW(D) = \emptyset$.

2. Prvý prechod pravidlami:

   - Pravidlo $S \to \underline{B}BA$, vyšetrujeme $FOLLOW(B)$, $\beta = BA$
     - $FIRST(BA) = \{a, c\}$, teda $a, c \in FOLLOW(B)$ a priebežne dostávame $FOLLOW(B) = \{a, c\}$.
   - Pravidlo $S \to B\underline{B}A$, vyšetrujeme $FOLLOW(B)$, $\beta = A$
     - $FIRST(A) = \{a, c\}$, teda $a, c \in FOLLOW(B)$, čo už vieme.
   - Pravidlo $S \to BB\underline{A}$, vyšetrujeme $FOLLOW(A)$, $\beta = \varepsilon$

- $FIRST(\varepsilon) = \{\varepsilon\}$.
- Keďže $\varepsilon \in FIRST(\beta)$, tak $FOLLOW(S) \subseteq FOLLOW(A)$, teda $\varepsilon \in FOLLOW(A)$ a priebežne dostávame $FOLLOW(A) = \{\varepsilon\}$.

- Pravidlo $A \to \underline{B}c$, vyšetrujeme $FOLLOW(B)$, $\beta = c$
  - $FIRST(c) = \{c\}$, teda $c \in FOLLOW(B)$, čo však už vieme.
- Pravidlo $B \to \underline{S}D$, vyšetrujeme $FOLLOW(S)$, $\beta = D$
  - $FIRST(D) = \{a, \varepsilon\}$, teda $a \in FOLLOW(S)$.
  - Keďže $\varepsilon \in FIRST(\beta)$, tak $FOLLOW(B) \subseteq FOLLOW(S)$. Aktuálne $FOLLOW(B) = \{a, c\}$, teda do $FOLLOW(S)$ pribudne aj c, $FOLLOW(S) = \{\varepsilon, a, c\}$.
- Pravidlo $B \to S\underline{D}$, vyšetrujeme $FOLLOW(D)$, $\beta = \varepsilon$
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Keďže $\varepsilon \in FIRST(\beta)$, tak $FOLLOW(B) \subseteq FOLLOW(D)$, teda $a, c \in FOLLOW(D)$ a priebežne dostávame $FOLLOW(D) = \{a, c\}$.
- Pravidlo $B \to \underline{A}Bb$, vyšetrujeme $FOLLOW(A)$, $\beta = Bb$
  - $FIRST(Bb) = \{a, b, c\}$, teda $a, b, c \in FOLLOW(A)$. Keďže doteraz sme mali len $FOLLOW(A) = \{\varepsilon\}$, po tejto informácii máme $FOLLOW(A) = \{\varepsilon, a, b, c\}$.
- Pravidlo $B \to A\underline{B}b$, vyšetrujeme $FOLLOW(B)$, $\beta = b$
  - $FIRST(b) = \{b\}$, teda $b \in FOLLOW(B)$. Toto je nová informácia, takže dostávame $FOLLOW(B) = \{a, b, c\}$.
- Pravidlo $C \to a\underline{C}D$, vyšetrujeme $FOLLOW(C)$, $\beta = D$
  - $FIRST(D) = \{a, \varepsilon\}$, teda $a \in FOLLOW(C)$. Toto je nová informácia, takže dostávame $FOLLOW(C) = \{a\}$.
  - Keďže $\varepsilon \in FIRST(\beta)$, tak $FOLLOW(C) \subseteq FOLLOW(C)$, z čoho nezískame žiadnu informáciu.
- Pravidlo $C \to aC\underline{D}$, vyšetrujeme $FOLLOW(D)$, $\beta = \varepsilon$
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Keďže $\varepsilon \in FIRST(\beta)$, tak $FOLLOW(C) \subseteq FOLLOW(D)$, teda $a \in FOLLOW(D)$, čo už vieme.
- Pravidlo $D \to \underline{C}b$, vyšetrujeme $FOLLOW(C)$, $\beta = b$
  - $FIRST(b) = \{b\}$, takže $b \in FOLLOW(C)$, teda $FOLLOW(C) = \{a, b\}$.

Teda po tomto prechode pravidlami máme aktuálny stav množín $FOLLOW$:

- $FOLLOW(S) = \{\varepsilon, a, c\}$,
- $FOLLOW(A) = \{\varepsilon, a, b, c\}$,

- $FOLLOW(B) = \{a, b, c\}$,
- $FOLLOW(C) = \{a, b\}$,
- $FOLLOW(D) = \{a, c\}$.

3. Keďže sa nám počas prechodu pravidlami zmenili množiny $FOLLOW$ neterminálov, prechod pravidlami zopakujeme. Pre jednoduchosť uvádzame len tie vyšetrovania pravidiel, ktoré nám poskytnú nové informácie:

   - Pravidlo $B \to \underline{S}D$, vyšetrujeme $FOLLOW(S)$, $\beta = D$
     - $FIRST(D) = \{a, \varepsilon\}$, teda $a \in FOLLOW(S)$, čo už vieme.
     - Keďže $\varepsilon \in FIRST(\beta)$, tak $FOLLOW(B) \subseteq FOLLOW(S)$. Aktuálne $FOLLOW(B) = \{a, b, c\}$, teda $a, b, c \in FOLLOW(S)$, kde $b \in FOLLOW(S)$ je nová informácia, aktuálne $FOLLOW(S) = \{\varepsilon, a, b, c\}$.
   - Pravidlo $B \to S\underline{D}$, vyšetrujeme $FOLLOW(D)$, $\beta = \varepsilon$
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - Keďže $\varepsilon \in FIRST(\beta)$, tak $FOLLOW(B) \subseteq FOLLOW(D)$, teda $a, b, c \in FOLLOW(D)$, z čoho $b \in FOLLOW(D)$ je nová informácia, teda $FOLLOW(D) = \{a, b, c\}$.

   Teda po tomto prechode pravidlami máme aktuálny stav množín $FOLLOW$:

   - $FOLLOW(S) = \{\varepsilon, a, b, c\}$,
   - $FOLLOW(A) = \{\varepsilon, a, b, c\}$,
   - $FOLLOW(B) = \{a, b, c\}$,
   - $FOLLOW(C) = \{a, b\}$,
   - $FOLLOW(D) = \{a, b, c\}$.

4. Keďže sa nám počas prechodu pravidlami zmenili množiny $FOLLOW$ neterminálov, mali by sme znovu prechod pravidlami zopakovať. V tomto prípade však už nič nové nedostaneme, čím dostávame výsledné množiny $FOLLOW$:

   - $FOLLOW(S) = \{\varepsilon, a, b, c\}$,
   - $FOLLOW(A) = \{\varepsilon, a, b, c\}$,
   - $FOLLOW(B) = \{a, b, c\}$,
   - $FOLLOW(C) = \{a, b\}$,
   - $FOLLOW(D) = \{a, b, c\}$.

Úloha č. 4.5.5 Je daná bezkontextová gramatika $G = (\{<\text{Regex}>, <\text{Term}>, <\text{Factor}>, <\text{Base}>, <\text{Char}>\}, \{a, b, !, +, *, (,)\}, P, <\text{Regex}>)$, ktorej pravidlá P sú:

- $<\text{Regex}> \rightarrow <\text{Term}> \mid <\text{Term}> + <\text{Regex}>$
- $<\text{Term}> \rightarrow <\text{Factor}> \mid <\text{Factor}><\text{Term}>$
- $<\text{Factor}> \rightarrow <\text{Base}> \mid <\text{Base}>*$
- $<\text{Base}> \rightarrow <\text{Char}> \mid (<\text{Regex}>)$
- $<\text{Char}> \rightarrow a \mid b \mid\ !$

Nájdite množiny $FOLLOW$ pre neterminály gramatiky.

Riešenie:
Množiny $FIRST$ jednotlivých neterminálov gramatiky sú:

- $FIRST(<\text{Regex}>) = \{(, a, b, !\}$,
- $FIRST(<\text{Term}>) = \{(, a, b, !\}$,
- $FIRST(<\text{Factor}>) = \{(, a, b, !\}$,
- $FIRST(<\text{Base}>) = \{(, a, b, !\}$,
- $FIRST(<\text{Char}>) = \{a, b, !\}$.

V danej gramatike:

1. Do množiny $FOLLOW(<\text{Regex}>)$ pridáme ε, keďže <Regex> je počiatočný neterminál.

   Po tomto kroku:

   - $FOLLOW(<\text{Regex}>) = \{\varepsilon\}$,
   - $FOLLOW(<\text{Term}>) = \emptyset$,
   - $FOLLOW(<\text{Factor}>) = \emptyset$,
   - $FOLLOW(<\text{Base}>) = \emptyset$,
   - $FOLLOW(<\text{Char}>) = \emptyset$.

2. Prvý prechod pravidlami:

   - Pravidlo $<\text{Regex}> \rightarrow \underline{<\text{Term}>}$, vyšetrujeme $FOLLOW(<\text{Term}>)$, $\beta = \varepsilon$
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(<\text{Regex}>) \subseteq FOLLOW(<\text{Term}>)$, teda aktuálne $FOLLOW(<\text{Term}>) = \{\varepsilon\}$.

- Pravidlo $\text{<Regex>} \to \underline{\text{<Term>}} + \text{<Regex>}$, vyšetrujeme $FOLLOW(\text{<Term>})$, $\beta = +\text{<Regex>}$
  - $FIRST(+\text{<Regex>}) = \{+\}$, teda $+ \in FOLLOW(\text{<Term>})$ a aktuálne $FOLLOW(\text{<Term>}) = \{\varepsilon, +\}$.
- Pravidlo $\text{<Regex>} \to \text{<Term>} + \underline{\text{<Regex>}}$, vyšetrujeme $FOLLOW(\text{<Regex>})$, $\beta = \varepsilon$
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(\text{<Regex>}) \subseteq FOLLOW(\text{<Regex>})$, z čoho nezistíme nič.
- Pravidlo $\text{<Term>} \to \underline{\text{<Factor>}}$, vyšetrujeme $FOLLOW(\text{<Factor>})$, $\beta = \varepsilon$
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(\text{<Term>}) \subseteq FOLLOW(\text{<Factor>})$, teda aktuálne $FOLLOW(\text{<Factor>}) = \{\varepsilon, +\}$.
- Pravidlo $\text{<Term>} \to \underline{\text{<Factor>}}\text{<Term>}$, vyšetrujeme $FOLLOW(\text{<Factor>})$, $\beta = \text{<Term>}$
  - $FIRST(\text{<Term>}) = \{(, a, b, !\}$, teda $(, a, b, ! \in FOLLOW(\text{<Factor>})$ a aktuálne $FOLLOW(\text{<Factor>}) = \{\varepsilon, +, (, a, b, !\}$.
- Pravidlo $\text{<Term>} \to \text{<Factor>}\underline{\text{<Term>}}$, vyšetrujeme $FOLLOW(\text{<Term>})$, $\beta = \varepsilon$
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(\text{<Term>}) \subseteq FOLLOW(\text{<Term>})$, z čoho nezistíme nič.
- Pravidlo $\text{<Factor>} \to \underline{\text{<Base>}}$, vyšetrujeme $FOLLOW(\text{<Base>})$, $\beta = \varepsilon$
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(\text{<Factor>}) \subseteq FOLLOW(\text{<Base>})$, z čoho dostávame aktuálne $FOLLOW(\text{<Base>}) = \{\varepsilon, +, (, a, b, !\}$.
- Pravidlo $\text{<Factor>} \to \underline{\text{<Base>}}*$, vyšetrujeme $FOLLOW(\text{<Base>})$, $\beta = *$
  - $FIRST(*) = \{*\}$, teda $* \in FOLLOW(\text{<Base>})$ a aktuálne $FOLLOW(\text{<Base>}) = \{\varepsilon, +, (, a, b, !, *\}$.
- Pravidlo $\text{<Base>} \to \underline{\text{<Char>}}$, vyšetrujeme $FOLLOW(\text{<Char>})$, $\beta = \varepsilon$
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(\text{<Base>}) \subseteq FOLLOW(\text{<Char>})$, z čoho dostávame aktuálne $FOLLOW(\text{<Char>}) = \{\varepsilon, +, (, a, b, !, *\}$.
- Pravidlo $\text{<Base>} \to (\underline{\text{<Regex>}})$, vyšetrujeme $FOLLOW(\text{<Regex>})$, $\beta =)$ (teda β obsahuje pravú zátvorku)
  - $FIRST()) = \{)\}$, teda $) \in FOLLOW(\text{<Regex>})$ a aktuálne $FOLLOW(\text{<Regex>}) = \{\varepsilon,)\}$.

Po tomto prechode pravidlami máme aktuálny stav množín $FOLLOW$:

- $FOLLOW(<\text{Regex}>) = \{\varepsilon,)\}$,
- $FOLLOW(<\text{Term}>) = \{\varepsilon, +\}$,
- $FOLLOW(<\text{Factor}>) = \{\varepsilon, +, (, a, b, !\}$,
- $FOLLOW(<\text{Base}>) = \{\varepsilon, +, (, a, b, !, *\}$,
- $FOLLOW(<\text{Char}>) = \{\varepsilon, +, (, a, b, !, *\}$.

3. Keďže sa nám počas prechodu pravidlami zmenili množiny $FOLLOW$ neterminálov, prechod pravidlami zopakujeme. Pre jednoduchosť uvádzame len tie vyšetrovania pravidiel, ktoré nám poskytnú nové informácie:

   - Pravidlo $<\text{Regex}> \to \underline{<\text{Term}>}$, vyšetrujeme $FOLLOW(<\text{Term}>)$, $\beta = \varepsilon$
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(<\text{Regex}>) \subseteq FOLLOW(<\text{Term}>)$, čím dostávame $FOLLOW(<\text{Term}>) = \{\varepsilon, +,)\}$.
   - Pravidlo $<\text{Term}> \to \underline{<\text{Factor}>}$, vyšetrujeme $FOLLOW(<\text{Factor}>)$, $\beta = \varepsilon$
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(<\text{Term}>) \subseteq FOLLOW(<\text{Factor}>)$, čím dostávame $FOLLOW(<\text{Factor}>) = \{\varepsilon, +, (, a, b, !,)\}$.
   - Pravidlo $<\text{Factor}> \to \underline{<\text{Base}>}$, vyšetrujeme $FOLLOW(<\text{Base}>)$, $\beta = \varepsilon$
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(<\text{Factor}>) \subseteq FOLLOW(<\text{Base}>)$, čím dostávame $FOLLOW(<\text{Base}>) = \{\varepsilon, +, (, a, b, !, *,)\}$.
   - Pravidlo $<\text{Base}> \to \underline{<\text{Char}>}$, vyšetrujeme $FOLLOW(<\text{Char}>)$, $\beta = \varepsilon$
     - $FIRST(\varepsilon) = \{\varepsilon\}$.
     - Keďže $\varepsilon \in FIRST(\beta)$, $FOLLOW(<\text{Base}>) \subseteq FOLLOW(<\text{Char}>)$, čím dostávame $FOLLOW(<\text{Char}>) = \{\varepsilon, +, (, a, b, !, *,)\}$.

   Teda po tomto prechode pravidlami máme aktuálny stav množín $FOLLOW$:

   - $FOLLOW(<\text{Regex}>) = \{\varepsilon,)\}$,
   - $FOLLOW(<\text{Term}>) = \{\varepsilon, +,)\}$,
   - $FOLLOW(<\text{Factor}>) = \{\varepsilon, +, (, a, b, !,)\}$,
   - $FOLLOW(<\text{Base}>) = \{\varepsilon, +, (, a, b, !, *,)\}$,
   - $FOLLOW(<\text{Char}>) = \{\varepsilon, +, (, a, b, !, *,)\}$.

4. Keďže sa nám počas prechodu pravidlami zmenili množiny $FOLLOW$ neterminálov, mali by sme znovu prechod pravidlami zopakovať. V tomto prípade však už nič nové nedostaneme, čím dostávame výsledné množiny $FOLLOW$:
   - $FOLLOW(\text{<Regex>}) = \{\varepsilon,)\}$,
   - $FOLLOW(\text{<Term>}) = \{\varepsilon, +,)\}$,
   - $FOLLOW(\text{<Factor>}) = \{\varepsilon, +, (, a, b, !,)\}$,
   - $FOLLOW(\text{<Base>}) = \{\varepsilon, +, (, a, b, !, *,)\}$,
   - $FOLLOW(\text{<Char>}) = \{\varepsilon, +, (, a, b, !, *,)\}$.

Kapitola 5

Zásobníkové automaty

5.1 Výpočet zásobníkového automatu

Úloha č. 5.1.1 Je daný zásobníkový automat (ZA) $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, ktorého množina stavov $Q = \{q_0, q_1, q_2, q_3\}$, vstupná abeceda $\Sigma = \{0, 1\}$, zásobníková abeceda $\Gamma = \{Z_0, 0\}$, počiatočný stav automatu je q_0, počiatočný zásobníkový symbol je Z_0, akceptačné stavy sú $F = \{q_0, q_3\}$ a prechodová funkcia δ je daná predpisom:

- $\delta(q_0, 0, Z_0) = \{(q_1, 0Z_0)\}$,
- $\delta(q_1, 0, 0) = \{(q_1, 00)\}$,
- $\delta(q_1, 1, 0) = \{(q_2, \varepsilon)\}$,
- $\delta(q_2, 1, 0) = \{(q_2, \varepsilon)\}$,
- $\delta(q_2, \varepsilon, Z_0) = \{(q_3, \varepsilon)\}$.

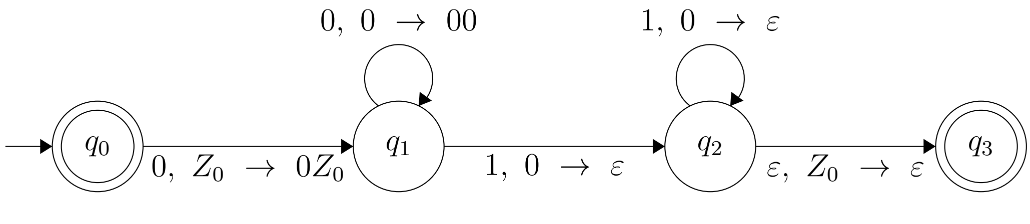
1. Nakreslite grafickú reprezentáciu daného ZA pomocou prechodového diagramu.
2. Zistite, či daný ZA akceptuje ret'azce: $\varepsilon, 0, 1, 01, 10, 0011$.
3. Určte, aký jazyk $L(M)$ akceptuje daný ZA.

Riešenie:

1. Prechodový diagram reprezentujúci ZA je znázornený na obrázku 5.1.

   Prechodový diagram je podobný prechodovému diagramu konečného automatu, s tým rozdielom, že hrany sú ohodnotené výrazom v tvare $a, X \to s$, kde:

   - a je vstupný symbol, ktorý sa pri prechode číta zo vstupu (prípadne $a = \varepsilon$, ak sa vstup pri prechode ignoruje),



Obr. 5.1: Prechodový diagram ZA z úlohy 5.1.1

- X je zásobníkový symbol na vrchu zásobníka, ktorý sa pri prechode prečíta (a odstráni) z vrchu zásobníka (prípadne $X = \varepsilon$, ak sa vrchný symbol zásobníka pri prechode ignoruje),
- s je ret'azec zásobníkových symbolov, ktoré sa následne pri prechode vložia na vrch zásobníka (prípadne $s = \varepsilon$, ak sa pri prechode do zásobníka nič nevkladá).

2. Výpočty daného ZA pre jednotlivé ret'azce zapisujeme, podobne ako pri konečných automatoch, pomocou konfigurácií. Každá konfigurácia predstavuje usporiadanú trojicu (q, x, y) v ktorej:

   - q je aktuálny stav, v ktorom sa ZA nachádza,
   - x je aktuálne neprečítaná čast' vstupu, pričom v danom momente vidí ZA prvý neprečítaný symbol vstupu, t. j. prvý symbol x,
   - y je aktuálny obsah zásobníka, pričom symboly y sú v zásobníku usporiadané od vrchu po spodok, t. j. prvý symbol y sa nachádza na vrchu zásobníka a posledný symbol y sa nachádza na dne zásobníka.

   Budeme uvažovat' tzv. **akceptáciu pomocou akceptačného stavu**, t. j. vstupný ret'azec bude zásobníkový automat akceptovat' vtedy, ked' prečíta všetky jeho symboly a skončí v niektorom z akceptačných stavov.

   - ret'azec ε: (q_0, ε, Z_0) — tento výpočet skončí v stave q_0. Ked'že stav q_0 je akceptačným stavom a zároveň bol vstupný ret'azec celý prečítaný, slovo ε ZA **akceptuje**, teda patrí do jazyka akceptovaného ZA, $\varepsilon \in L(M)$.
   - ret'azec 0: $(q_0, 0, Z_0) \vdash (q_1, \varepsilon, 0Z_0)$ — tento výpočet skončí v stave q_1. Vstupný ret'azec bol celý prečítaný, avšak výpočet skončil v neakceptačnom stave. Ked'že neexistuje iný výpočet pre ret'azec 0, ktorý by spracoval 0 a skončil v akceptačnom stave, slovo 0 automat **neakceptuje**, $0 \notin L(M)$.
   - ret'azec 1: $(q_0, 1, Z_0)$ — tento výpočet sa zasekne v stave q_0, pretože v aktuálnej konfigurácii nie je možný žiaden d'alší krok výpočtu. Slovo 1 automat **neakceptuje**, $1 \notin L(M)$.
   - ret'azec 01: $(q_0, 01, Z_0) \vdash (q_1, 1, 0Z_0) \vdash (q_2, \varepsilon, Z_0) \vdash (q_3, \varepsilon, \varepsilon)$ — vidíme, že automat prečítal celý vstup 01 a zároveň skončil v akceptačnom stave, teda slovo 01 **akceptuje**, $01 \in L(M)$.

- ret'azec 10: $(q_0, 10, Z_0)$ — tento výpočet sa zasekne v stave q_0, pretože v aktuálnej konfigurácii nie je možný žiaden d'alší krok výpočtu. Slovo 10 automat **neakceptuje**, $10 \notin L(M)$.
- ret'azec 0011: $(q_0, 0011, Z_0) \vdash (q_1, 011, 0Z_0) \vdash (q_1, 11, 00Z_0) \vdash (q_2, 1, 0Z_0) \vdash (q_2, \varepsilon, Z_0) \vdash (q_3, \varepsilon, \varepsilon)$ — vidíme, že automat prečítal celý vstup 0011 a zároveň skončil v akceptačnom stave, teda slovo 0011 **akceptuje**, $0011 \in L(M)$.

3. Jazyk $L(M) = \{0^n1^n \mid n \in \mathbb{Z}_0^+\}$, teda ret'azce obsahujúce rovnaký počet núl a jednotiek, pričom tieto symboly sú v ret'azci usporiadané, najprv nuly, potom jednotky.

   - Aby sme určili, aký jazyk automat akceptuje, môžeme skúsit' odhadnút', ako vyzerajú ret'azce, pre ktoré výpočet začínajúci v počiatočnom stave q_0 dospeje do niektorého z akceptačných stavov — q_0 alebo q_3.
   - Jediný ret'azec, ktorý akceptujeme v stave q_0, je ret'azec ε, teda $\varepsilon \in L(M)$.
   - Všetky ostatné ret'azce, ktoré automat akceptuje, akceptuje v stave q_3. Aby sa výpočet dostal do stavu q_3, musí sa ocitnút' v stave q_2, vstup musí byt' celý prečítaný a na vrchu zásobníka musí byt' počiatočný zásobníkový symbol Z_0.
   - Podl'a pravidiel je zrejmé, že automat na začiatku výpočtu zo vstupu číta nuly a postupne ich ukladá do zásobníka:
     - Prvú nulu vstupu vloží nad počiatočný zásobníkový symbol Z_0 a prepne sa do stavu q_1,
     - v stave q_1 číta všetky d'alšie nuly, po ich prečítaní ich vloží nad ostatné nuly v zásobníku.
     - Ak by vstup začínal jednotkou, automat sa zasekne v stave q_0.
   - Následne, ak sa na vstupe nachádzajú jednotky:
     - Prvú jednotku vstupu prečíta v stave q_1 a prepne sa do stavu q_2, pričom odstráni jednu nulu zo zásobníka.
     - V stave q_2 postupne číta jednotky na vstupe, pričom pre každú prečítanú jednotku zároveň odstráni jednu nulu zo zásobníka.
     - Ak by sa v tomto momente na vstupe nachádzala nula, automat sa zasekne v stave q_2.
   - Ak sa na vstupe nachádzal rovnaký počet núl ako jednotiek a boli utriedené v poradí najprv nuly, potom jednotky, tak automat po prečítaní celého vstupu skončí v stave q_2, pričom každá nula, ktorá bola prečítaná a vložená do zásobníka, bola následne zo zásobníka odstránená čítaním jednotiek a na vrchu zásobníka musí v takom prípade zostat' symbol Z_0.
   - Teda, ak sa automat nachádza v stave q_2 a na vrchu zásobníka je symbol Z_0, automat prejde do akceptačného stavu q_3. Ak bol vstup celý prečítaný, dochádza k jeho akceptácii a musel byt' v tvare 0^n1^n, kde $n \in \mathbb{Z}^+$.
   - Teda spolu s prázdnym ret'azcom ide o jazyk $L(M) = \{0^n1^n \mid n \in \mathbb{Z}_0^+\}$.

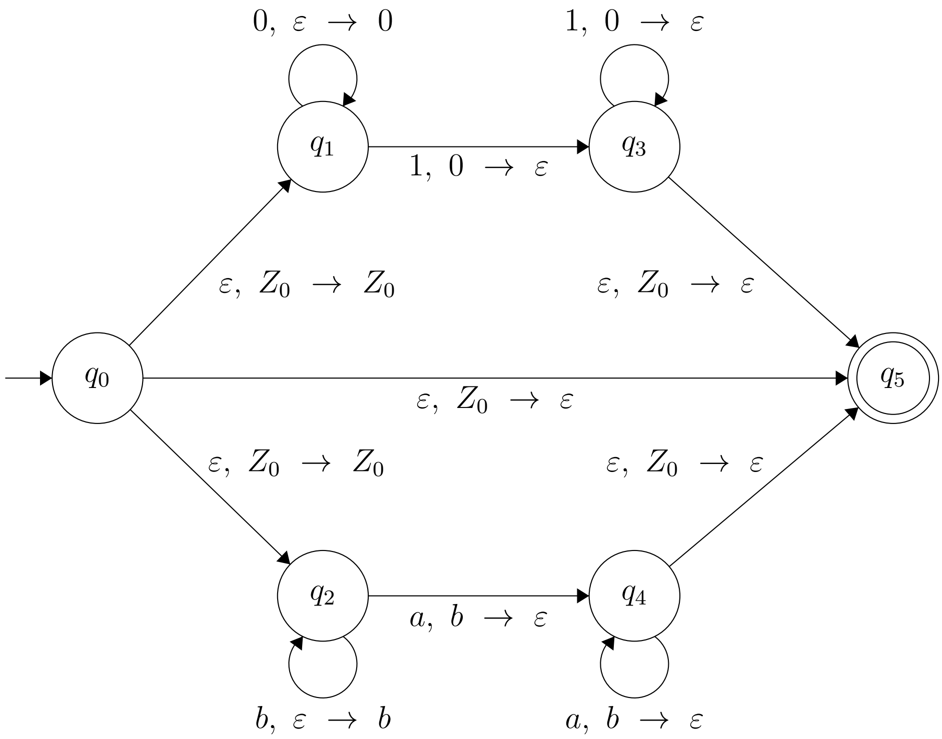
Úloha č. 5.1.2 Je daný zásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde stavy konečného automatu $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$, vstupná abeceda $\Sigma = \{0, 1, a, b\}$, zásobníková abeceda $\Gamma = \{Z_0, 0, b\}$, počiatočný stav automatu je q_0, počiatočný zásobníkový symbol je Z_0, akceptačné stavy sú $F = \{q_5\}$ a prechodová funkcia δ je daná predpisom:

- $\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0), (q_2, Z_0), (q_5, \varepsilon)\}$,
- $\delta(q_1, 0, \varepsilon) = \{(q_1, 0)\}$,
- $\delta(q_1, 1, 0) = \{(q_3, \varepsilon)\}$,
- $\delta(q_2, b, \varepsilon) = \{(q_2, b)\}$,
- $\delta(q_2, a, b) = \{(q_4, \varepsilon)\}$,
- $\delta(q_3, 1, 0) = \{(q_3, \varepsilon)\}$,
- $\delta(q_4, a, b) = \{(q_4, \varepsilon)\}$,
- $\delta(q_3, \varepsilon, Z_0) = \{(q_5, \varepsilon)\}$,
- $\delta(q_4, \varepsilon, Z_0) = \{(q_5, \varepsilon)\}$,

1. Nakreslite grafickú reprezentáciu daného ZA pomocou prechodového diagramu.
2. Zistite, či daný ZA akceptuje ret'azce: $01, ba, 011$.
3. Určte, aký jazyk $L(M)$ akceptuje daný ZA.

Riešenie:

1. Prechodový diagram reprezentujúci ZA je znázornený na obrázku 5.2.
2. Budeme uvažovať tzv. **akceptáciu pomocou akceptačného stavu**, t. j. vstupný ret'azec bude zásobníkový automat akceptovať vtedy, keď prečíta všetky jeho symboly a skončí v niektorom z akceptačných stavov.
   - Vidíme, že v tomto ZA existujú nedeterministické prechody — napríklad zo stavu q_0 vieme ísť do jedného zo stavov q_1, q_2, q_5 bez čítania vstupného symbolu, ak je na vrchu zásobníka Z_0, takže pri zistení, či ZA akceptuje ret'azce, musíme skúmať všetky možnosti a hľadať akceptačný výpočet:
   - ret'azec 01:
     - $(q_0, 01, Z_0) \vdash (q_5, 01, \varepsilon)$ — tento výpočet sa zasekne v stave q_5 s neprečítanou časťou vstupu,
     - $(q_0, 01, Z_0) \vdash (q_2, 01, Z_0)$ — tento výpočet sa zasekne v stave q_2 s neprečítanou časťou vstupu,



Obr. 5.2: Prechodový diagram ZA z úlohy 5.1.2

- $(q_0, 01, Z_0) \vdash (q_1, 01, Z_0) \vdash (q_1, 1, 0Z_0) \vdash (q_3, \varepsilon, Z_0) \vdash (q_5, \varepsilon, \varepsilon)$ — tento výpočet prečíta celý vstup a skončí v akceptačnom stave, teda slovo 01 ZA **akceptuje**.

- reťazec ba:
  - $(q_0, ba, Z_0) \vdash (q_5, ba, \varepsilon)$ — tento výpočet sa zasekne v stave q_5 s neprečítanou časťou vstupu,
  - $(q_0, ba, Z_0) \vdash (q_1, ba, Z_0)$ — tento výpočet sa zasekne v stave q_1 s neprečítanou časťou vstupu,
  - $(q_0, ba, Z_0) \vdash (q_2, ba, Z_0) \vdash (q_2, a, bZ_0) \vdash (q_4, \varepsilon, Z_0) \vdash (q_5, \varepsilon, \varepsilon)$ — tento výpočet prečíta celý vstup a skončí v akceptačnom stave, teda slovo ba ZA **akceptuje**.

- reťazec 011:
  - $(q_0, 011, Z_0) \vdash (q_5, 011, \varepsilon)$ — tento výpočet sa zasekne v stave q_5 s neprečítanou časťou vstupu,
  - $(q_0, 011, Z_0) \vdash (q_2, 011, Z_0)$ — tento výpočet sa zasekne v stave q_2 s neprečítanou časťou vstupu,
  - $(q_0, 011, Z_0) \vdash (q_1, 011, Z_0) \vdash (q_1, 11, 0Z_0) \vdash (q_3, 1, Z_0) \vdash (q_5, 1, \varepsilon)$ — tento výpočet sa zasekne v akceptačnom stave, avšak vstup nebol celý prečítaný, teda ani tento výpočet nie je akceptačný.

- Iné výpočty pre ret'azec 011 nie sú v tomto ZA možné, teda neexistuje akceptačný výpočet pre ret'azec 011 a ZA ret'azec 011 **neakceptuje**.

3. Jazyk $L(M) = \{0^n1^n \mid n \in \mathbb{Z}^+\} \cup \{b^m a^m \mid m \in \mathbb{Z}^+\} \cup \{\varepsilon\}$, teda obsahuje alebo prázdny ret'azec, alebo ret'azce v tvare 0^n1^n alebo $b^m a^m$, kde n, m sú kladné celé čísla, pretože:

   - uvedený ZA de facto pozostáva z 2 menších ZA:
     - ZA, ktorý „začína" v stave q_1, ktorý akceptuje ret'azce tvaru 0^n1^n,
     - ZA, ktorý „začína" v stave q_2, ktorý akceptuje ret'azce tvaru $b^m a^m$.
     - Zo stavu q_0 sa nedeterministicky vyberie prechod alebo do q_1, alebo do q_2. Ak je na vstupe slovo z množiny $\{0^n1^n \mid n \in \mathbb{Z}^+\}$, tak určite existuje akceptačný výpočet v prípade, ak sa ZA rozhodne v stave q_0 pre prechod do stavu q_1 (a analogicky pre množinu $\{b^m a^m \mid m \in \mathbb{Z}^+\}$ a prechod do stavu q_2).
   - Okrem toho vie uvedený ZA akceptovat' aj prázdny ret'azec vd'aka prechodu priamo z q_0 do q_5.

5.2 Konštrukcia zásobníkového automatu

Úloha č. 5.2.1 Je daný jazyk $L = \{ww^R \mid w \in \{0,1\}^*\}$ nad abecedou $A = \{0,1\}$. Nájdite zásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, ktorý akceptuje jazyk L.

Riešenie: Pri zostrojení zásobníkového automatu M, ktorý akceptuje nejaký požadovaný jazyk L, musíme analogicky s konštrukciou gramatiky alebo konečného automatu dbat' na to, aby boli splnené 2 podmienky:

1. Každý ret'azec, ktorý bude zásobníkový automat M akceptovat', musí byt' zároveň ret'azcom patriacim do jazyka L, teda musí platit' $L(M) \subseteq L$.

2. Každý ret'azec z jazyka L musí mat' v zásobníkovom automate M akceptačný výpočet, teda musí platit' $L \subseteq L(M)$.

Ak sú tieto 2 podmienky splnené, potom platí $L(M) = L$, teda jazyk $L(M)$ akceptovaný zásobníkovým automatom M je totožný s jazykom L.

Pri konštrukcii zásobníkového automatu si taktiež musíme stanovit', aký typ akceptácie budeme využívat':

- Akceptácia akceptačným stavom — ZA akceptuje vstupný ret'azec, ak ho celý prečítal a výpočet skončil v akceptačnom stave.

- Akceptácia prázdnym zásobníkom — ZA akceptuje vstupný ret'azec, ak ho celý prečítal a výpočet skončil s prázdnym zásobníkom, t. j. v zásobníku nezostali žiadne symboly.

- Oba prístupy sú ekvivalentné, takže my si vyberieme prvý spôsob — akceptáciu pomocou akceptačných stavov.

Podobne, ako tomu bolo pri konštrukcii gramatiky, resp. KA, aj pri konštrukcii ZA je dobré začať tým, že si vymenujeme aspoň najkratšie reťazce patriace do jazyka L, aby sme videli, aké reťazce vlastne potrebujeme akceptovať. Do zadaného jazyka $L = \{ww^R \mid w \in \{0,1\}^*\}$ patria napríklad reťazce:

- ε, kde $w = \varepsilon, w^R = \varepsilon$
- 00, kde $w = 0, w^R = 0$
- 11, kde $w = 1, w^R = 1$
- 0000, kde $w = 00, w^R = 00$
- 0110, kde $w = 01, w^R = 10$
- 1001, kde $w = 10, w^R = 01$
- 1111, kde $w = 11, w^R = 11$

Ako je zo zadaného jazyka zrejmé, každý reťazec, ktorý do jazyka patrí, sa dá rozdeliť na 2 podreťazce polovičnej dĺžky:

- Ľubovoľný reťazec w zložený z núl a jednotiek,
- jeho zrkadlový obraz w^R.

ZA spracuje vstupné slovo v 2 fázach:

- V prvej fáze bude ZA čítať prvú polovicu vstupného slova — reťazec w, pričom každý prečítaný symbol vloží na vrch zásobníka.
- Po prečítaní reťazca w sa teda na dne zásobníka nachádza symbol Z_0, nad ním prvý symbol reťazca w, a tak ďalej, až na vrchu zásobníka sa nachádza posledný symbol reťazca w.
- V druhej fáze bude automat kontrolovať, či sa na vstupe nachádza zrkadlový obraz reťazca w, teda w^R. Ak áno, tak potom prvý symbol w^R musí byť rovnaký, ako bol posledný symbol w, teda ten symbol, ktorý je aktuálne **na vrchu zásobníka**, keďže tam bol pridaný ako posledný.
- Postupne sa teda budú čítať vstupné symboly w^R a porovnávať, či sa zhodujú so symbolmi na vrchu zásobníka. Ak áno, po prečítaní celého reťazca w^R na vrchu zásobníka zostane symbol Z_0.

Keďže zadaný jazyk L obsahuje len reťazce zložené zo symbolov $\{0, 1\}$, aj automat zostrojíme tak, že jeho vstupnou abecedou budú symboly, $\Sigma = \{0, 1\}$. Keďže tieto symboly budeme rovnako vkladať do zásobníka, kde sa na začiatku nachádza aj počiatočný zásobníkový symbol Z_0, zásobníková abeceda bude $\Gamma = \{Z_0, 0, 1\}$.

Na začiatku je automat v počiatočnom stave q_0. Stav q_0 bude predstavovať situáciu, že sa číta predpona w vstupného reťazca, t. j. ZA bude čítať vstupné symboly a vkladať ich na vrch zásobníka:

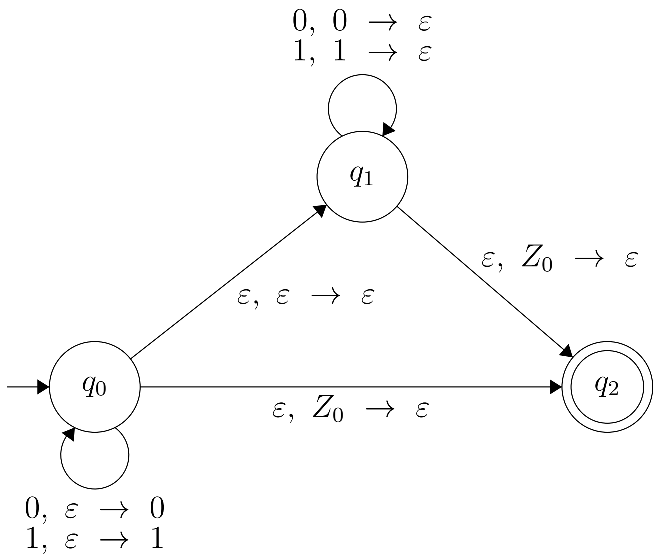
- Pri čítaní vstupného symbolu a jeho vložení do zásobníka môžeme aktuálny symbol na vrchu zásobníka ignorovať, čo zapíšeme pomocou prechodovej funkcie: $\delta(q_0, 0, \varepsilon) = \{(q_0, 0)\}, \delta(q_0, 1, \varepsilon) = \{(q_0, 1)\}$.
- Do ZA pridáme prechod, ktorý umožní ZA prejsť do nového stavu q_1 v momente, keď ZA prečíta prvú polovicu vstupu, t. j. prefix w, pričom tento prechod bude možné vykonať bez čítania vstupného symbolu a bez ohľadu na to, čo je na vrchu zásobníka, pričom do zásobníka sa nič nevloží, t. j. prechod: $\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \varepsilon)\}$.
- Upozorňujeme, že v rámci výpočtu nejakého vstupného slova ZA **nevie**, kde sa nachádza polovica vstupu, pretože ZA v každom momente vidí len aktuálny vstupný symbol a nevie určiť, aký je počet zvyšných symbolov na vstupe. Teda tým, že tento prechod z q_0 do q_1 urobíme **nedeterministicky** pokryjeme všetky situácie, **vrátane tej**, ktorá je žiadúca, teda že k tomuto prechodu dôjde práve v momente, keď sa prečíta polovica vstupu — prefix w.

Stav q_1 bude predstavovať situáciu, že sa číta druhá polovica vstupu, t. j. slovo w^R, ktoré je zrkadlovým obrazom prvej polovice vstupu. Ak bolo na vstupe naozaj slovo ww^R a k prechodu zo stavu q_0 do stavu q_1 prišlo v momente, keď sa prečítala prvá polovica vstupu, reťazec w, v zásobníku sa vždy navrchu nachádza práve ešte neprečítaná časť reťazca w^R, teda aktuálne čítaný symbol a symbol navrchu zásobníka sa musia zhodovať:

- Pri čítaní vstupného symbolu sa teda pozrieme na vrchný symbol zásobníka a ak sa zhodujú, symbol na vstupe sa prečíta a symbol sa zo zásobníka odstráni: $\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}, \delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$.
- Ak sa na vrchu zásobníka ocitne symbol Z_0, predpokladáme, že sme práve na vstupe dočítali reťazec w^R a prejdeme do akceptačného stavu q_2, $\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$.

Stav q_2 bude jediný akceptačný stav ZA. Ak sa v ňom automat ocitne, predpokladáme, že na vstupe bol reťazec v tvare ww^R. V stave q_2 už potom nebudú žiadne prechody, avšak pridáme ešte prechod zo stavu q_0 do stavu q_2 bez čítania vstupu, avšak za predpokladu, že na vrchu zásobníka je Z_0 — tento prechod sa bude používať na akceptovanie prázdneho reťazca:

- $\delta(q_0, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$.



Obr. 5.3: Prechodový diagram ZA z úlohy č. 5.2.1

Prechodový diagram výsledného zásobníkového automatu uvádzame na obrázku 5.3. Skontrolujme, aspoň neformálne, či sú splnené nasledovné podmienky:

1. Platí $L(M) \subseteq L$?

   - Zisťujeme, či každý reťazec, ktorý náš ZA akceptuje, spĺňa zároveň podmienku jazyka L.
   - Aby nami zostrojený ZA akceptoval vstupný reťazec, musí výpočet skončiť v stave q_2, pričom na vstupe nesmie zostať žiaden neprečítaný symbol:
     (a) Ak je na vstupe ε, je možné skončiť v q_2, napríklad $(q_0, \varepsilon, Z_0) \vdash (q_2, \varepsilon, \varepsilon)$.
     (b) Ak je na vstupe neprázdny reťazec, tak pri jeho čítaní musel byť ZA aj v stave q_1, pretože v stave q_0 sa čítané symboly ukladajú do zásobníka a teda nie je možné prejsť priamo zo stavu q_0 do stavu q_2, keďže tento prechod požaduje, aby na vrchu zásobníka bol symbol Z_0.
     (c) Ak sa pri čítaní dostal ZA do stavu q_1, potom z tohto stavu sa môže dostať do stavu q_2 len vtedy, ak bude na vrchu zásobníka Z_0. Na to je potrebné v stave q_1 odstrániť zo zásobníka všetky symboly 0/1, ktoré tam boli pridané v stave q_0. Je zrejmé, že k takejto situácii môže dôjsť len v prípade, že v stave q_0 bola prečítaná (a vložená do zásobníka) práve polovica vstupu a v stave q_1 bola prečítaná (a odstránená zo zásobníka) druhá polovica vstupu, ktorá musela byť zrkadlovým obrazom prvej polovice (inak by sa pri čítaní symbolov a porovnávaní so symbolmi zo zásobníka výpočet niekde zasekol).
   - Teda celý vstupný reťazec musel byť alebo ε, alebo v tvare ww^R, kde w pozostáva z núl a jednotiek, čiže vstupné slovo zároveň patrí do jazyka L, teda platí $L(M) \subseteq L$.

2. Platí $L \subseteq L(M)$?

   - Zisťujeme, či každý reťazec, ktorý patrí do jazyka L, má zároveň v ZA akceptačný výpočet.
   - Uvažujme ľubovoľný reťazec w patriaci do jazyka L, t. j. reťazec tvaru ww^R, $w \in \{0,1\}^*$.
   - V ZA určite existuje výpočtová vetva v rámci ktorej sa:
     - Prečíta prefix w v stave q_0, teda všetky jeho symboly sa postupne uložia do zásobníka. Po prečítaní prefixu w je teda obsah zásobníka v tvare $w^R Z_0$.
     - ZA následne prejde do stavu q_1, kde úspešne postupne porovná zvyšok vstupu w^R so symbolmi v zásobníku, až prečíta celý vstup a v zásobníku zostane len Z_0.
     - Následne ZA už len prejde do stavu q_2, pričom vstup celý spracoval.
   - V ZA teda určite existuje nasledovný výpočet:

$$(q_0, ww^R, Z_0) \vdash^* (q_0, w^R, w^R Z_0) \vdash (q_1, w^R, w^R Z_0) \vdash^* (q_1, \varepsilon, Z_0) \vdash (q_2, \varepsilon, \varepsilon)$$

   - Ak je teda na vstupe slovo z jazyka L, určite v ZA existuje jeho akceptačný výpočet, teda $L \subseteq L(M)$.

Keďže $L(M) \subseteq L$ a zároveň $L \subseteq L(M)$, tak určite platí $L(M) = L$, čo znamená, že jazyk $L(M)$ akceptovaný automatom M je totožný s jazykom L, a teda je náš zásobníkový automat správny.

Len pre úplnosť, zostrojili sme zásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, ktorého stavy $Q = \{q_0, q_1, q_2\}$, vstupná abeceda $\Sigma = \{0,1\}$, zásobníková abeceda $\Gamma = \{0, 1, Z_0\}$, q_0 je počiatočný stav, Z_0 je počiatočný zásobníkový symbol, množina akceptačných stavov $F = \{q_2\}$ a prechodová funkcia δ je znázornená na obrázku 5.3.

Úloha č. 5.2.2 Je daný jazyk $L = \{w \in \{a,b\}^* \mid \sharp_a(w) = \sharp_b(w)\}$ nad abecedou $A = \{a,b\}$. Nájdite zásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, ktorý akceptuje jazyk L.

Riešenie:
Do zadaného jazyka $L = \{w \in \{a,b\}^* \mid \sharp_a(w) = \sharp_b(w)\}$ patria všetky reťazce nad abecedou $A = \{a,b\}$, v ktorých je rovnaký počet symbolov a a b:

- ε, (0-krát a a b)
- ab, ba, (1-krát a a b)
- $aabb, abab, abba, baab, baba, bbaa$, (2-krát a a b),
- atď.

V našej konštrukcii využijeme spôsob akceptácie pomocou akceptačných stavov. Náš zásobníkový automat musí akceptovať len tie reťazce, ktoré majú rovnaký počet symbolov a a b. Ak teda napríklad prvý prečítaný symbol bude a, niekde neskôr na vstupe sa musí zjaviť korešpondujúci symbol b, a naopak. Zásobník automatu teda využijeme na zapamätanie toho, koľko „opačných“ symbolov (k a považujeme v tomto prípade za opačný symbol b a naopak) ešte musíme prečítať zo vstupu, aby bol počet a a b vyrovnaný. Každý chýbajúci symbol b (resp. a) na vstupe bude reprezentovaný symbolom b (resp. a) v zásobníku.

Pri spracovaní reťazca budeme uvažovať 3 situácie vzhľadom na **už prečítanú časť vstupu**:

1. V prečítanej časti vstupu bol rovnaký počet a a b — táto situácia je o.i. počiatočnou situáciou, keď ešte nebol prečítaný žiaden vstupný symbol. V tejto situácii sa na vrchu zásobníka bude nachádzať symbol Z_0 a bude reprezentovaná stavom q_0.
2. V prečítanej časti vstupu bol väčší počet symbolov a než b — pre akceptovanie vstupu je teda potrebné prečítať ešte minimálne jeden symbol b. V tejto situácii sa na vrchu zásobníka bude nachádzať minimálne jeden symbol b a bude reprezentovaná stavom $q_{a>b}$.
3. V prečítanej časti vstupu bol väčší počet symbolov b než a — pre akceptovanie vstupu je teda potrebné prečítať ešte minimálne jeden symbol a. V tejto situácii sa na vrchu zásobníka bude nachádzať minimálne jeden symbol a a bude reprezentovaná stavom $q_{b>a}$.

Keďže zadaný jazyk L obsahuje len reťazce zložené zo symbolov $\{a, b\}$, vstupnou abecedou budú symboly, $\Sigma = \{a, b\}$. Keďže symboly a, b budeme rovnako vkladať do zásobníka, kde sa na začiatku nachádza aj počiatočný zásobníkový symbol Z_0, zásobníková abeceda bude $\Gamma = \{Z_0, a, b\}$.

V stave q_0 je na vrchu zásobníka symbol Z_0 označujúci, že sme doteraz prečítali rovnaký počet a a b.

- Ak je aktuálnym vstupným symbolom a, ZA prejde do stavu $q_{a>b}$ a do zásobníka vloží symbol b, $\delta(q_0, a, Z_0) = \{(q_{a>b}, bZ_0)\}$.
- Ak je aktuálnym vstupným symbolom b, ZA prejde do stavu $q_{b>a}$ a do zásobníka vloží symbol a, $\delta(q_0, b, Z_0) = \{(q_{b>a}, aZ_0)\}$.

Ak je v stave $q_{a>b}$ na vrchu zásobníka symbol b, znamená to, že očakávame, že na vstupe bude ešte minimálne jeden symbol b, aby sa vyrovnali počty prečítaných symbolov a a b:

- Ak je aktuálnym vstupným symbolom a, ZA zostane v stave $q_{a>b}$ a do zásobníka vloží **ďalší** symbol b, pretože práve prečítané a nám signalizuje ďalšie chýbajúce b na vstupe, $\delta(q_{a>b}, a, b) = \{(q_{a>b}, bb)\}$.

- Ak je aktuálnym vstupným symbolom b, ZA zostane v stave $q_{a>b}$ a zo zásobníka odstráni symbol b, pretože sme práve na vstupe prečítali chýbajúci symbol b, $\delta(q_{a>b}, b, b) = \{(q_{a>b}, \varepsilon)\}$.
- Ak v stave $q_{a>b}$ dôjde k tomu, že prečítaním symbolu b zo vstupu sa **vyrovnajú** počty doteraz prečítaných symbolov a a b, na vrch zásobníka sa dostane symbol Z_0. V takom prípade sa ZA vráti do stavu q_0 bez čítania vstupu a pri zachovaní symbolu Z_0 na vrchu zásobníka, $\delta(q_{a>b}, \varepsilon, Z_0) = \{(q_0, Z_0)\}$.

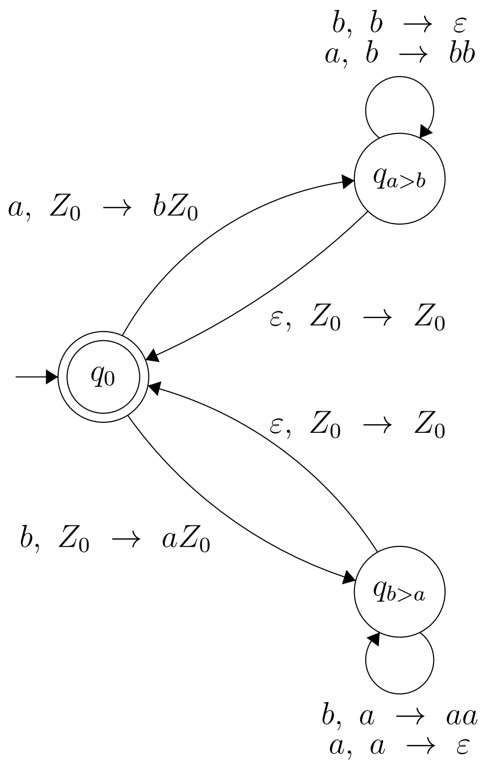
Stav $q_{b>a}$ sa správa analogicky k stavu $q_{a>b}$, avšak v stave $q_{b>a}$ očakávame minimálne jeden výskyt symbolu a na vstupe:

- Ak je aktuálnym vstupným symbolom b, $\delta(q_{b>a}, b, a) = \{(q_{b>a}, aa)\}$.
- Ak je aktuálnym vstupným symbolom a, $\delta(q_{b>a}, a, a) = \{(q_{b>a}, \varepsilon)\}$.
- Ak v stave $q_{b>a}$ dôjde k tomu, že prečítaním symbolu a zo vstupu sa **vyrovnajú** počty doteraz prečítaných symbolov a a b, na vrch zásobníka sa dostane symbol Z_0. V takom prípade $\delta(q_{b>a}, \varepsilon, Z_0) = \{(q_0, Z_0)\}$.

Stav q_0 predstavujúci situáciu, že doteraz prečítaný počet symbolov a a b bol rovnaký, bude jediný akceptačný stav ZA. Ak sa v ňom automat ocitne po prečítaní celého vstupu, na vstupe musel byť' ret'azec obsahujúci rovnaký počet a a b. Prechodový diagram výsledného zásobníkového automatu uvádzame na obrázku 5.4.

Skontrolujme, aspoň neformálne, či sú splnené nasledovné podmienky:

1. Platí $L(M) \subseteq L$?
   - Zist'ujeme, či každý ret'azec, ktorý náš automat akceptuje, spĺňa zároveň podmienku jazyka L.
   - Triviálne vidíme, že náš automat akceptuje prázdny ret'azec, ktorý zároveň patrí aj do jazyka L.
   - Nech automat akceptuje ret'azec s aspoň 1 symbolom. Ak je prvý symbol vstupu a, ZA bude po jeho prečítaní v stave $q_{a>b}$. Analogicky, ak by bol prvý symbol vstupu b, ZA by bol po jeho prečítaní v stave $q_{b>a}$.
   - Zo stavu $q_{a>b}$, resp. $q_{b>a}$, ZA **neodíde**, kým sa nepodarí prečítat' tol'ko symbolov a a b, že ich počty sú rovnaké. Ked' nastane táto situácia, automat sa vráti spät' do stavu q_0.
   - Uvedená situácia sa môže v rámci jedného akceptačného výpočtu opakovat', t. j. akceptovaný ret'azec je možné rozdelit' na podret'azce $w_1, w_2, ..., w_n$ obsahujúce rovnaký počet symbolov a a b, pre ktoré platí: $(q_0, w_i, Z_0) \vdash^* (q_0, \varepsilon, Z_0)$, pričom počas ich spracovania sa ZA nachádzal alebo v stave $q_{a>b}$ alebo $q_{b>a}$.



Obr. 5.4: Prechodový diagram ZA z úlohy č. 5.2.2

- Ak teda ZA akceptuje nejaký ret'azec s aspoň 1 symbolom, tento ret'azec pozostáva z podret'azcov $w_1w_2...w_n$, ktoré obsahujú rovnaké počty symbolov a a b, teda aj celý akceptovaný vstupný ret'azec musí obsahovat' rovnaký počet symbolov a a b, čiže patrí do jazyka L, teda platí $L(M) \subseteq L$.

2. Platí $L \subseteq L(M)$?

- Zist'ujeme, či každý ret'azec, ktorý patrí do jazyka L, má zároveň v automate akceptačný výpočet.
- Uvažujme l'ubovol'ný ret'azec w patriaci do jazyka L, teda obsahujúci rovnaký počet symbolov a a b. Takýto ret'azec sa dá rozdelit' na menšie podret'azce, $w = w_1w_2...w_n$, pre ktoré platí, že v rámci w_i je rovnaký počet symbolov a a b, pričom ak podret'azec w_i začína symbolom a, bude končit' k nemu „opačným“ symbolom b a naopak. Napríklad pre $w = abbbaa$ by takéto delenie bolo $w_1 = ab, w_2 = bbaa$.
- V automate určite existuje výpočtová vetva, kde:
  - Pri čítaní každého podret'azca w_i sa automat zo stavu q_0 prepne podl'a prvého symbolu w_i alebo do stavu $q_{a>b}$, ak je v podret'azci w_i prvý symbol a, alebo do stavu $q_{b>a}$, ak je v podret'azci w_i prvý symbol b.

- Následne sa po spracovaní podreťazca w_i, keďže ten obsahuje rovnaký počet symbolov a a b vie ZA dostať späť do stavu q_0, t. j. pre každý podreťazec w_i platí $(q_0, w_i, Z_0) \vdash^* (q_0, \varepsilon, Z_0)$.

- V automate teda určite existuje nasledovný výpočet:

$$(q_0, w_1 w_2 ... w_n, Z_0) \vdash^* (q_0, w_2 ... w_n, Z_0) \vdash^* (q_0, w_n, Z_0) \vdash^* (q_0, \varepsilon, Z_0)$$

- Ak je teda na vstupe slovo z jazyka L, určite v ZA existuje jeho akceptačný výpočet, teda $L \subseteq L(M)$.

Keďže $L(M) \subseteq L$ a zároveň $L \subseteq L(M)$, tak určite platí $L(M) = L$, čo znamená, že jazyk $L(M)$ akceptovaný automatom M je totožný s jazykom L, a teda je náš automat správny.

Len pre úplnosť, zostrojili sme zásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, ktorého stavy $Q = \{q_0, q_{a>b}, q_{b>a}\}$, vstupná abeceda $\Sigma = \{a, b\}$, zásobníková abeceda $\Gamma = \{Z_0, a, b\}$, q_0 je počiatočný stav, Z_0 je počiatočný zásobníkový symbol, množina akceptačných stavov $F = \{q_0\}$ a prechodová funkcia δ je znázornená na obrázku 5.4.

Úloha č. 5.2.3 Je daný jazyk $L = \{w \in \{0,1\}^* \mid \sharp_0(w) = 2 * \sharp_1(w)\}$ nad abecedou $A = \{0, 1\}$. Nájdite zásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, ktorý akceptuje jazyk L.

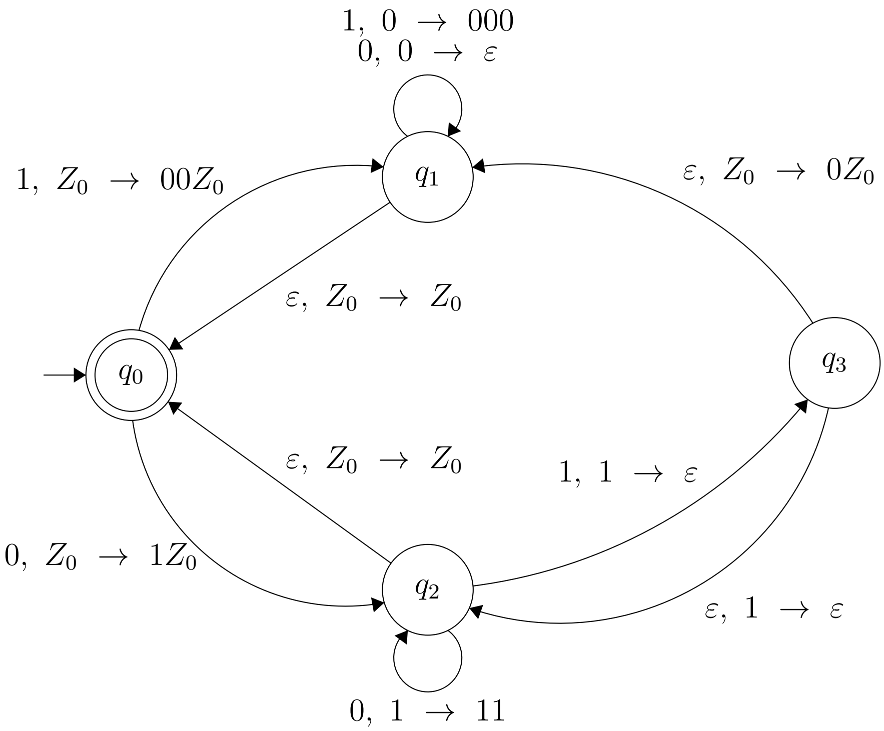
Riešenie:
Do zadaného jazyka $L = \{w \in \{0,1\}^* \mid \sharp_0(w) = 2 * \sharp_1(w)\}$ patria všetky reťazce nad abecedou $A = \{0, 1\}$, v ktorých je počet symbolov 0 dvakrát väčší než počet symbolov 1. Patria sem napríklad reťazce:

- ε (0 výskytov 0 a 1)
- $001, 010, 100$ (2 výskyty 0 a 1 výskyt 1)
- $000011, 000101, 001001, ..., 110000$ (4 výskyty 0 a 2 výskyty 1)
- atď.

Riešením by bol napríklad zásobníkový automat akceptujúci pomocou akceptačných stavov, ktorého množina stavov $Q = \{q_0, q_1, q_2, q_3\}$, vstupná abeceda $\Sigma = \{0, 1\}$, zásobníkové symboly $\Sigma = \{Z_0, 0, 1\}$, počiatočným stavom je q_0, počiatočným zásobníkovým symbolom je Z_0, akceptačné stavy sú $F = \{q_0\}$ a prechodová funkcia je znázornená na obrázku 5.5.

Tento ZA pracuje podľa nasledovného princípu:

- Stav q_0 predstavuje situáciu, že v doteraz prečítanej časti vstupu bol počet núl dvojnásobkom počtu jednotiek. Túto informáciu zároveň ilustruje aj fakt, že sa na vrchu zásobníka nachádza symbol Z_0.



Obr. 5.5: Prechodový diagram ZA z úlohy č. 5.2.3

- Stav q_1 predstavuje situáciu, že sme doteraz prečítali **menší počet núl** než dvojnásobok doteraz prečítaných jednotiek. Túto skutočnosť ilustruje aj výskyt núl v zásobníku — každá nula, ktorá sa v stave q_1 nachádza na vrchu zásobníka predstavuje jednu nulu, ktorú potrebujeme rozpoznať na vstupe.

- Stav q_2 predstavuje situáciu, že sme doteraz prečítali **menší počet jednotiek** než implikuje doteraz prečítaný počet núl. Túto skutočnosť ilustruje výskyt jednotiek v zásobníku — každá navyše prečítaná nula zo vstupu vyrobí v zásobníku jednu jednotku, teda **dve jednotky** v zásobníku predstavujú **jednu jednotku na vstupe**, ktorú musíme prečítať.

  – Ak je v stave q_2 na vstupe nula, prečítame ju a pridáme do zásobníka ďalšiu jednotku.

  – Ak je v stave q_2 na vstupe jednotka, pokúsime sa zo zásobníka odstrániť dve jednotky — nepriamo, prechodom cez stav q_3. Prvú jednotku odstránime zo zásobníka pri prechode do stavu q_3 a druhú sa pokúsime odstrániť v stave q_3 ignorovaním vstupu. Všimnite si, že v stave q_3 môžu nastať 2 situácie.

- Stav q_3:

  – V stave q_3 sa na vrchu zásobníka nachádza jednotka — v takom prípade ju odstránime a vrátime sa do stavu q_2. Tento prechod reprezentuje fakt, že jednotka na vstupe úspešne odstránila dve jednotky zo zásobníka.

– Ak sa v stave q_3 na vrchu zásobníka už jednotka nenachádza, musí to znamenať, že jednotka, ktorú sme zo zásobníka odstránili pri prechode do stavu q_3 bola jedinou jednotkou v zásobníku — teda k jednotke, ktorú sme prečítali pri prechode do q_3, nám vlastne chýba ešte jedna nula na vstupe! Preto, ak sa na vrchu zásobníka zjaví symbol Z_0, je zo stavu q_3 vedený prechod do stavu q_1, ktorý do zásobníka vloží jednu nulu.

Úloha č. 5.2.4 Je daný jazyk $L = \{a^i b^j c^k \mid i, j, k \in \mathbb{Z}_0^+,\ \text{kde } i = j \text{ alebo } j = k\}$ nad abecedou $A = \{a, b, c\}$. Nájdite zásobníkový automat $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, ktorý akceptuje jazyk L.

Riešenie:
Zásobníkový automat najjednoduchšie zostrojíme, ak si uvedomíme, že zadaný jazyk $L = \{a^i b^j c^k \mid i, j, k \in \mathbb{Z}_0^+,\ \text{kde } i = j \text{ alebo } j = k\}$ predstavuje zjednotenie 2 jazykov:

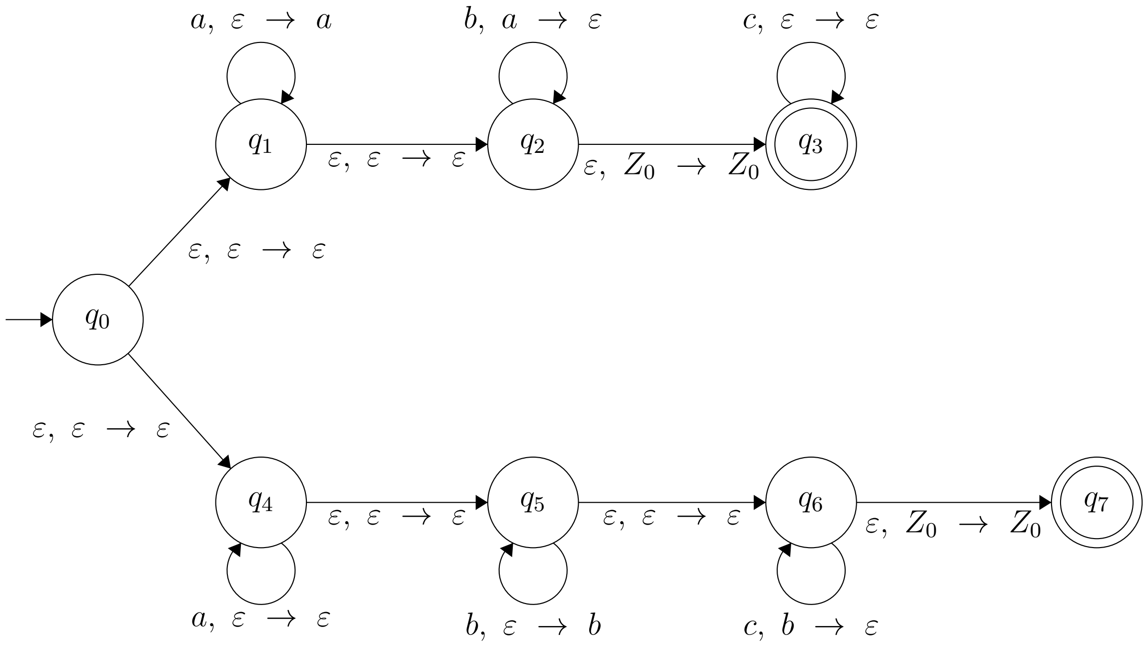
$$\{a^i b^i c^k \mid i, k \in \mathbb{Z}_0^+\} \cup \{a^i b^j c^j \mid i, j \in \mathbb{Z}_0^+\}.$$

Následne zostrojíme časť ZA, ktorá bude akceptovať jazyk $\{a^i b^i c^k \mid i, k \in \mathbb{Z}_0^+\}$, časť ZA, ktorá bude akceptovať jazyk $\{a^i b^j c^j \mid i, j \in \mathbb{Z}_0^+\}$ a z počiatočného stavu ZA sa bude nedeterministickým spôsobom možné prepnúť do jednej z týchto 2 častí.

Riešením by bol napríklad zásobníkový automat akceptujúci pomocou akceptačných stavov, ktorého množina stavov $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$, vstupná abeceda $\Sigma = \{a, b, c\}$, zásobníkové symboly $\Sigma = \{Z_0, a, b\}$, počiatočným stavom je q_0, počiatočným zásobníkovým symbolom je Z_0, akceptačné stavy sú $F = \{q_3, q_7\}$ a prechodová funkcia je znázornená na obrázku 5.6.

Tento ZA pracuje podľa nasledovného princípu:

- Stavy q_1, q_2, q_3 tvoria časť zásobníkového automatu, ktorá akceptuje reťazce patriace do množiny $\{a^i b^i c^k \mid i, k \in \mathbb{Z}_0^+\}$.
- Stavy q_4, q_5, q_6, q_7 tvoria časť zásobníkového automatu, ktorá akceptuje reťazce patriace do množiny $\{a^i b^j c^j \mid i, j \in \mathbb{Z}_0^+\}$.
- Stav q_0 je počiatočným stavom ZA, z ktorého sa automat nedeterministicky prepne alebo do stavu q_1, alebo do stavu q_4.
- Ak je na vstupe reťazec patriaci do množiny $\{a^i b^i c^k \mid i, k \in \mathbb{Z}_0^+\}$, tak ak sa ZA nedeterministicky na začiatku výpočtu prepne do stavu q_1, bude preň existovať akceptačný výpočet, teda takýto reťazec bude ZA akceptovať.
- Analogicky, ak je na vstupe reťazec patriaci do množiny $\{a^i b^j c^j \mid i, j \in \mathbb{Z}_0^+\}$, tak ak sa ZA nedeterministicky na začiatku výpočtu prepne do stavu q_4, bude preň existovať akceptačný výpočet, teda takýto reťazec bude ZA akceptovať.
- Teda uvedený ZA bude akceptovať jazyk $\{a^i b^i c^k \mid i, k \in \mathbb{Z}_0^+\} \cup \{a^i b^j c^j \mid i, j \in \mathbb{Z}_0^+\}$, čiže jazyk $L = \{a^i b^j c^k \mid i, j, k \in \mathbb{Z}_0^+,\ \text{kde } i = j \text{ alebo } j = k\}$.



Obr. 5.6: Prechodový diagram ZA z úlohy č. 5.2.4

Kapitola 6

Lexikálna analýza

6.1 Teoretická konštrukcia lexikálneho analyzátora

Úloha č. 6.1.1 Zostrojte lexikálny analyzátor v tvare DKA, ktorý pomocou akceptačných stavov rozpoznáva nasledujúce lexémy popísané príslušnými regulárnymi výrazmi:

1.	`identifikátor`	$(a\|b\|...\|z\|A\|B\|...\|Z)(a\|b\|...\|z\|A\|B\|...\|Z\|0\|...\|9)^*$
2.	`=`	$(=)$
3.	`+`	$(+)$
4.	`-`	$(-)$
5.	`;`	$(;)$
6.	`konšt_int`	$(+\|-\|\varepsilon)(0\|...\|9)(0\|...\|9)^*$
7.	`konšt_real`	$(+\|-\|\varepsilon)(0\|...\|9)(0\|...\|9)^*(.)(0\|...\|9)(0\|...\|9)^*$

Riešenie:
Lexikálny analyzátor rozpoznávajúci uvedené lexémy zostrojíme tak, že najprv zostrojíme samostatné NKA alebo DKA rozpoznávajúce jednotlivé lexémy, resp. jazyky popísané ich príslušnými regulárnymi výrazmi.

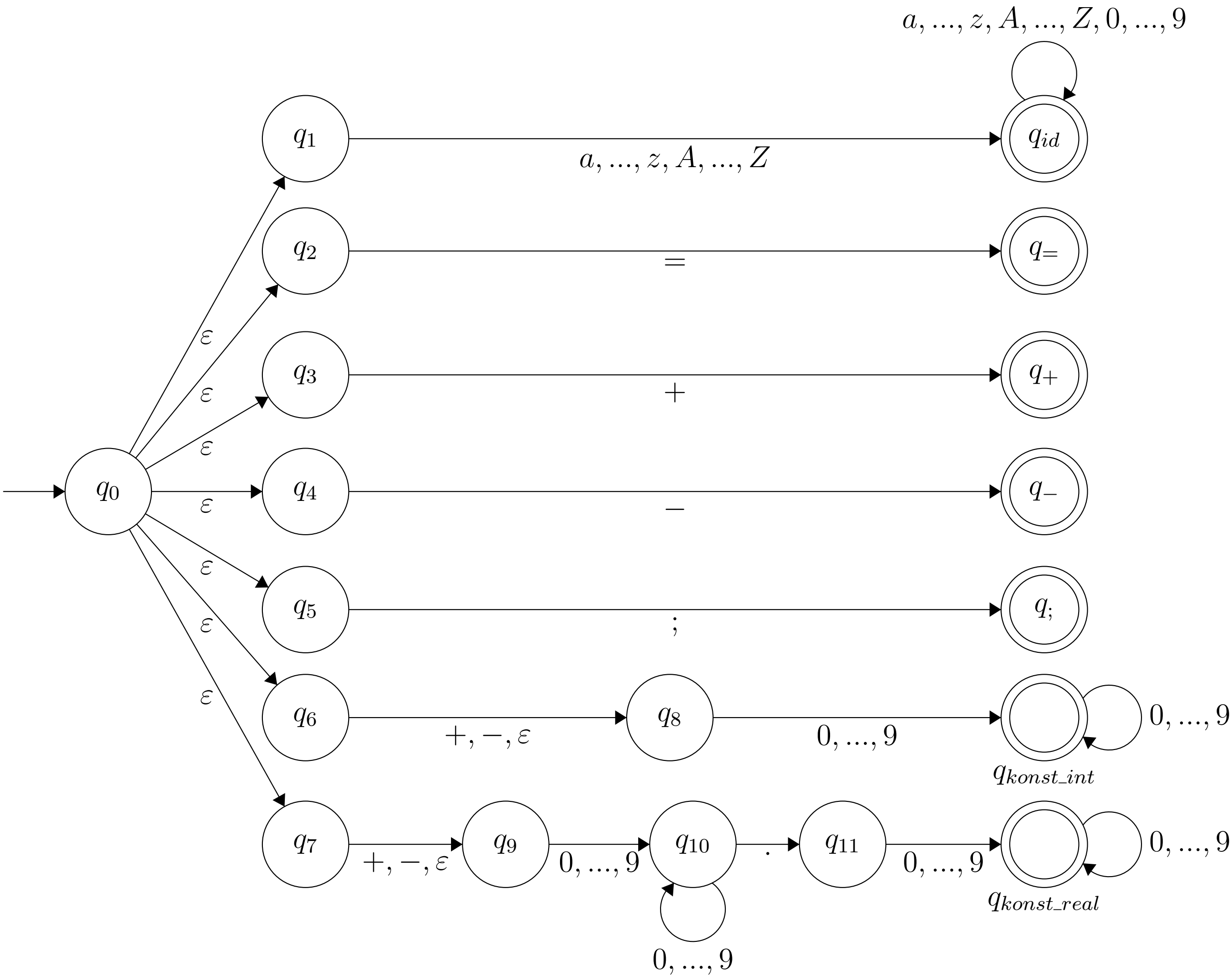
Následne zostrojíme výsledný nedeterministický konečný automat, ktorý bude ich **zjednotením**. V našom prípade je takýto výsledný nedeterministický konečný automat zobrazený na obrázku 6.1.

Na obrázku 6.1 vidíme, že v dôsledku konštrukcie má každá lexéma v NKA vlastný akceptačný stav, t. j. podľa toho, v akom stave sa NKA zastaví počas spracovania vstupu, vieme rozpoznať príslušnú lexému:

1. stav q_{id} rozpoznáva lexému `identifikátor`,

2. stav $q_=$ rozpoznáva lexému `=`,

3. stav q_+ rozpoznáva lexému `+`,

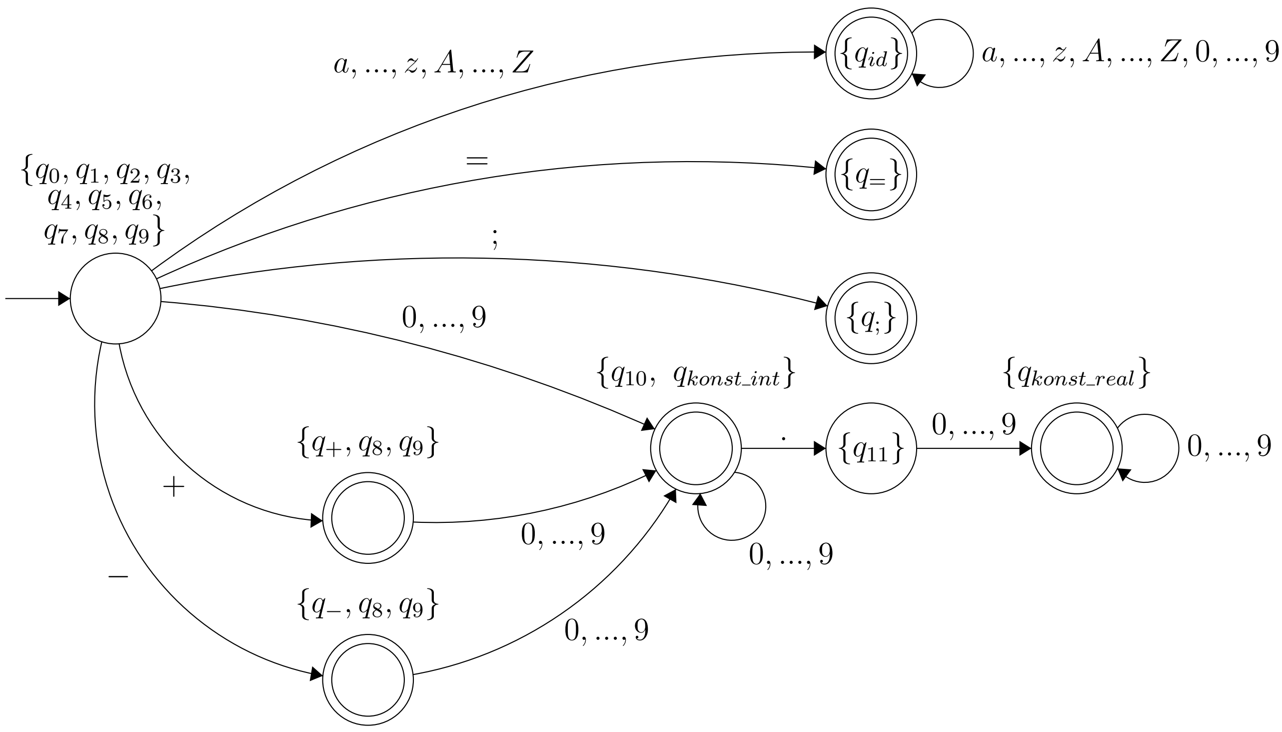


Obr. 6.1: Lexikálny analyzátor v tvare NKA pre úlohu 6.1.1.

4. stav q_- rozpoznáva lexému `-`,
5. stav $q_;$ rozpoznáva lexému `;`,
6. stav q_{konst_int} rozpoznáva lexému `konšt_int`,
7. stav q_{konst_real} rozpoznáva lexému `konšt_real`.

Aby sme zostrojili lexikálny analyzátor v tvare deterministického konečného automatu, stačí determinizovat' NKA na obrázku 6.1, čím dostávame DKA uvedený na obrázku 6.2.

Na obrázku 6.2 vidíme, že v dôsledku determinizácie vzniknú v DKA akceptačné stavy, ktorých názvy v sebe obsahujú akceptačné stavy pôvodného nedeterministického automatu. V tomto prípade platí, že ak sa príslušný lexikálny analyzátor počas spracovania



Obr. 6.2: Lexikálny analyzátor v tvare DKA pre úlohu č. 6.1.1

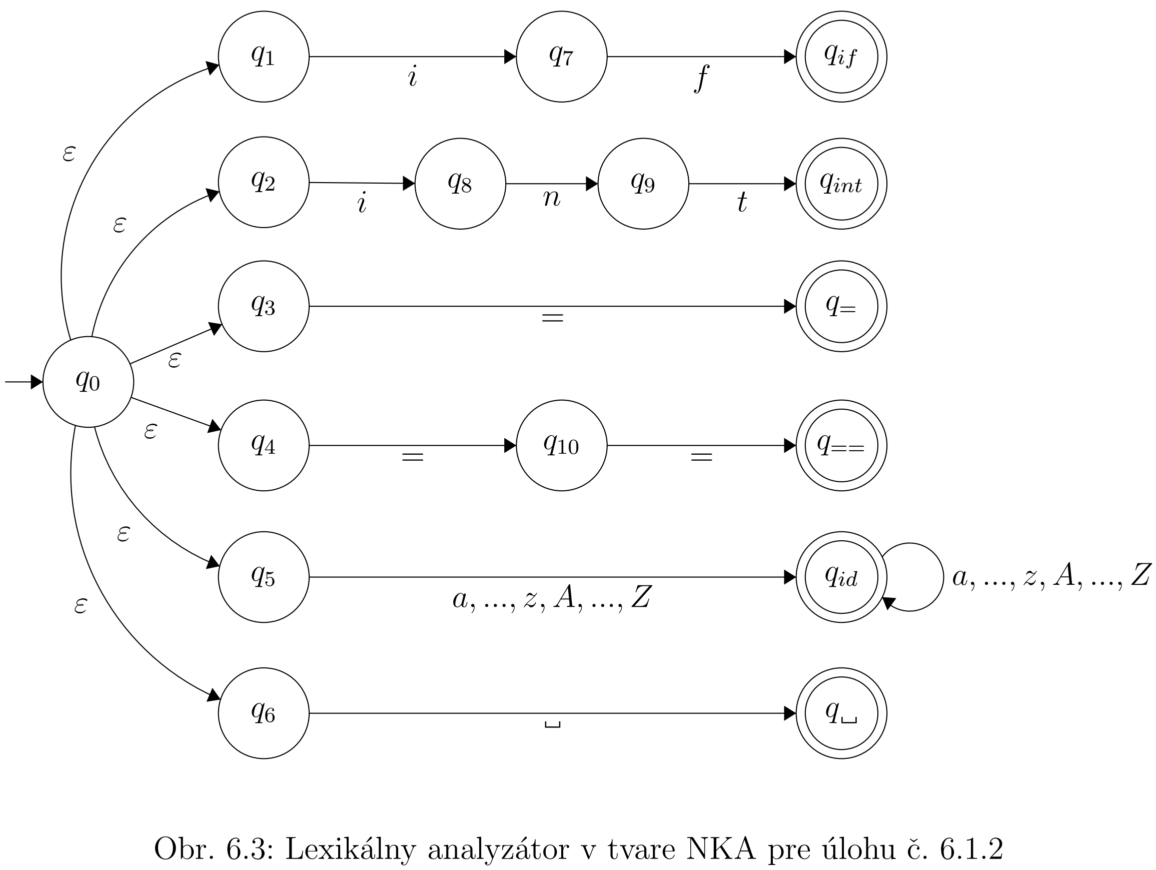
vstupu zastaví v akceptačnom stave, rozpozná sa príslušná lexéma podľa toho, ktorý akceptačný stav pôvodného NKA sa nachádza v danom akceptačnom stave DKA:

1. stav $\{q_{id}\}$ rozpoznáva lexému `identifikátor`,
2. stav $\{q_=\}$ rozpoznáva lexému `=`,
3. stav $\{q_+, q_8, q_9\}$ rozpoznáva lexému `+`,
4. stav $\{q_-, q_8, q_9\}$ rozpoznáva lexému `-`,
5. stav $\{q_;\}$ rozpoznáva lexému `;`,
6. stav $\{q_{10}, q_{konst\_int}\}$ rozpoznáva lexému `konšt_int`,
7. stav $\{q_{konst\_real}\}$ rozpoznáva lexému `konšt_real`.

Úloha č. 6.1.2 Zostrojte lexikálny analyzátor v tvare DKA, ktorý pomocou akceptačných stavov rozpoznáva nasledujúce lexémy[†] popísané príslušnými regulárnymi výrazmi:

1.	`if`	(if)
2.	`int`	(int)
3.	`=`	$(=)$
4.	`==`	$(==)$
5.	`identifikátor`	$(a\|b\|...\|z\|A\|B\|...\|Z)(a\|b\|...\|z\|A\|B\|...\|Z)^*$
6.	`medzera`	(␣)

Riešenie:
Lexikálny analyzátor rozpoznávajúci uvedené lexémy zostrojíme tak, že najprv zostrojíme samostatné NKA alebo DKA rozpoznávajúce jednotlivé lexémy, resp. jazyky popísané ich príslušnými regulárnymi výrazmi. Následne zostrojíme výsledný nedeterministický konečný automat, ktorý bude ich **zjednotením**. V našom prípade je takýto výsledný nedeterministický konečný automat zobrazený na obrázku 6.3.



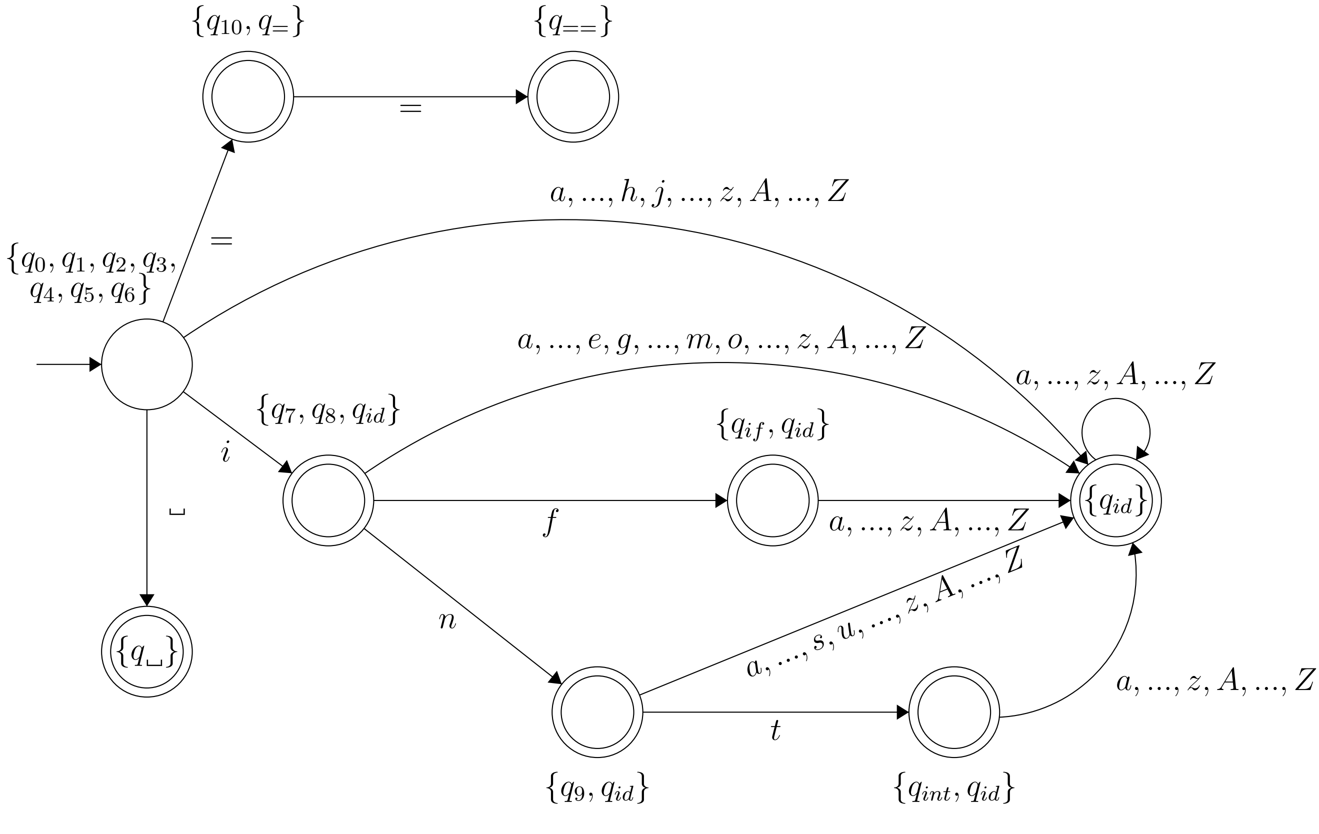
Obr. 6.3: Lexikálny analyzátor v tvare NKA pre úlohu č. 6.1.2

[†] Niekedy je žiadúce mať' lexémy, ktoré vyjadrujú tzv. netlačiteľné (*white-space*) znaky. V tomto príklade budeme uvažovať' medzeru ␣ ako potenciálnu lexému.

Na obrázku 6.3 vidíme, že v dôsledku konštrukcie má každá lexéma v NKA vlastný akceptačný stav, t. j. podľa toho, v akom stave sa NKA zastaví počas spracovania vstupu, vieme rozpoznať príslušnú lexému:

1. stav q_{if} rozpoznáva lexému `if`,
2. stav q_{int} rozpoznáva lexému `int`,
3. stav $q_{=}$ rozpoznáva lexému `=`,
4. stav $q_{==}$ rozpoznáva lexému `==`,
5. stav q_{id} rozpoznáva lexému `identifikátor`,
6. stav $q_{\sqcup}$ rozpoznáva lexému `medzera`.

Aby sme zostrojili lexikálny analyzátor v tvare deterministického konečného automatu, stačí determinizovať NKA na obrázku 6.3, čím dostávame DKA uvedený na obrázku 6.4.



Obr. 6.4: Lexikálny analyzátor v tvare DKA pre úlohu č. 6.1.2

Na obrázku 6.4 vidíme, že v dôsledku determinizácie vzniknú v DKA akceptačné stavy, ktorých názvy v sebe obsahujú akceptačné stavy pôvodného nedeterministického automatu. Znovu platí, že ak sa príslušný lexikálny analyzátor počas spracovania vstupu zastaví v akceptačnom stave, rozpozná sa príslušná lexéma podľa toho, ktorý akceptačný stav pôvodného NKA sa nachádza v danom akceptačnom stave DKA. V tomto DKA však nastáva situácia, že existujú akceptačné stavy, ktoré vo svojom názve obsahujú viacero akceptačných stavov pôvodného NKA:

1. stav $\{q_{if}, q_{id}\}$ zodpovedá rozpoznaným lexémam `identifikátor` a `if`,
2. stav $\{q_{int}, q_{id}\}$ zodpovedá rozpoznaným lexémam `identifikátor` a `int`.

To je spôsobené tým, že existujú reťazce, ktoré súčasne spĺňajú regulárne výrazy pre rôzne lexémy, napríklad reťazec `int` patrí súčasne aj do jazyka popísaného regulárnym výrazom (int), aj $(a|...|Z)(a|...|Z)^*$. Keďže lexikálny analyzátor musí rozpoznávať lexémy jednoznačne, v takomto prípade musíme explicitne určiť, akú lexému má rozpoznávať v prípade, že sa zastaví v stave $\{q_{if}, q_{id}\}$, resp. $\{q_{int}, q_{id}\}$. V praxi sa to rieši tak, že sa niektorej z lexém dá väčšia priorita — v tomto prípade sú lexémy `if` a `int` **špecifickejšie** než lexéma `identifikátor`, preto v stavoch $\{q_{if}, q_{id}\}$, resp. $\{q_{int}, q_{id}\}$ budeme rozpoznávať lexémy `if`, resp. `int`.

1. stav $\{q_{if}, q_{id}\}$ rozpoznáva lexému `if`,
2. stav $\{q_{int}, q_{id}\}$ rozpoznáva lexému `int`,
3. stav $\{q_{10}, q_{=}\}$ rozpoznáva lexému `=`,
4. stav $\{q_{==}\}$ rozpoznáva lexému `==`,
5. stav $\{q_{id}\}$ rozpoznáva lexému `identifikátor`,
6. stav $\{q_9, q_{id}\}$ rozpoznáva lexému `identifikátor`,
7. stav $\{q_7, q_8, q_{id}\}$ rozpoznáva lexému `identifikátor`,
8. stav $\{q_{\sqcup}\}$ rozpoznáva lexému `medzera`.

6.2 Činnosť lexikálneho analyzátora

Úloha č. 6.2.1 Použite lexikálny analyzátor v tvare DKA, ktorý pomocou akceptačných stavov rozpoznáva nasledujúce lexémy popísané regulárnymi výrazmi

1.	`identifikátor`	$(a\|b\|...\|z\|A\|B\|...\|Z)(a\|b\|...\|z\|A\|B\|...\|Z\|0\|...\|9)^*$
2.	`=`	$(=)$
3.	`+`	$(+)$
4.	`-`	$(-)$
5.	`;`	$(;)$
6.	`konšt_int`	$(+\|-\|\varepsilon)(0\|...\|9)(0\|...\|9)^*$
7.	`konšt_real`	$(+\|-\|\varepsilon)(0\|...\|9)(0\|...\|9)^*(.)(0\|...\|9)(0\|...\|9)^*$

na tokenizáciu (lexikálnu analýzu) reťazcov:

1. `i1=1+-1.25`
2. `i1=1+-1..25`
3. `-1;-1PREM1+5.1`

Riešenie:
Lexikálny analyzátor (LA) v tvare DKA k uvedeným lexémam sme zostrojili v úlohe č. 6.1.1 a je uvedený na obrázku 6.2. LA vo forme DKA pracuje v podstate podobne ako klasický DKA, s tým rozdielom, že disponuje vyrovnávacou pamät'ou a jeho činnost' je nasledovná [2]:

1. Postupne číta jednotlivé znaky vstupného ret'azca, kým sa nezasekne, t. j. pokial' existuje prechod z aktuálneho stavu na aktuálny vstupný symbol.
2. Zároveň každý prečítaný znak ukladá do vyrovnávacej pamäte.
3. Ak sa zasekne v akceptačnom stave, vráti lexému, ktorá je asociovaná s príslušným akceptačným stavom. Zároveň sa vo vyrovnávacej pamäti nachádza ret'azec, ktorý bol rozpoznaný ako príslušná lexéma. Následne sa lexikálny analyzátor vráti do počiatočného stavu a pokračuje v rozpoznávaní lexém.
4. Ak sa zasekne v neakceptačnom stave, nájde sa najbližší predchádzajúci akceptačný stav, v ktorom bola rozpoznaná niektorá platná lexéma. Lexikálny analyzátor potom vráti lexému, ktorá je asociovaná s príslušným akceptačným stavom a nasledujúcu lexému sa pokúsi rozpoznat' od konca predchádzajúcej, t. j. zvyšok už prečítaného vstupu vráti z vyrovnávacej pamäte spät' na vstup.
5. Ak sa taká lexéma pri spätnom hl'adaní nenájde, signalizuje to lexikálnu chybu.

Spracovanie ret'azcov:

1. Ret'azec `i1=1+-1.25`

   LA číta vstup a posúva sa po stavoch † kým sa nezasekne:

Symbol vo VP	ε	`i`	`i1`
Stav	$\{q_0, ..., q_9\}$	$\{q_{id}\}$	$\{q_{id}\}$ (zásek)

   LA sa zasekol v stave $\{q_{id}\}$, čo je stav, v ktorom sa rozpoznáva lexéma **`identifikátor`**. LA vyprázdni vyrovnávaciu pamät', vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`=`
Stav	$\{q_0, ..., q_9\}$	$\{q_=\}$ (zásek)

†Počiatočný stav v DKA je označený ako $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}$, z dôvodu obmedzeného miesta ho uvádzame len ako $\{q_0, ..., q_9\}$

LA sa zasekol v stave $\{q_=\}$, čo je stav, v ktorom sa rozpoznáva lexéma `=`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`1`
Stav	$\{q_0, ..., q_9\}$	$\{q_{10}, q_{konst\_int}\}$ (zásek)

LA sa zasekol v stave $\{q_{10}, q_{konst\_int}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `konst_int`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`+`
Stav	$\{q_0, ..., q_9\}$	$\{q_+, q_8, q_9\}$ (zásek)

LA sa zasekol v stave $\{q_+, q_8, q_9\}$, čo je akceptačný stav, ktorého názov síce pozostáva z viacerých stavov pôvodného NKA, avšak len q_+ bol akceptačným stavom v pôvodnom NKA. A keďže bol akceptačným stavom pre lexému `+`, v stave $\{q_+, q_8, q_9\}$ sa teda rozpoznáva lexéma `+`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`-`	`-1`	`-1.`
Stav	$\{q_0, ..., q_9\}$	$\{q_-, q_8, q_9\}$	$\{q_{10}, q_{konst\_int}\}$	$\{q_{11}\}$

Symbol vo VP	`-1.2`	`-1.25`
Stav	$\{q_{konst\_real}\}$	$\{q_{konst\_real}\}$ (zásek)

LA sa zasekol v stave $\{q_{konst\_real}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `konst_real`. LA vyprázdni vyrovnávaciu pamäť a keďže vstup bol celý prečítaný, činnosť lexikálneho analyzátora končí. Vidíme, že vstupný reťazec bol rozdelený na lexémy nasledovným spôsobom:

`i1`	`=`	`1`	`+`	`-1.25`
`identifikátor`	`=`	`konšt_int`	`+`	`konšt_real`

2. Reťazec `i1=1+-1..25`

   Spracovanie tohto reťazca zo začiatku prebehne podobne ako v predchádzajúcom prípade, t. j. rozpoznajú sa lexémy:

   - `identifikátor` (reťazec `i1`)
   - `=` (reťazec `=`)
   - `konšt_int` (reťazec `1`)
   - `+` (reťazec `+`)

   a pokračujeme v spracovaní zvyšku:

Symbol vo VP	ε	`-`	`-1`	`-1.`
Stav	$\{q_0, ..., q_9\}$	$\{q_-, q_8, q_9\}$	$\{q_{10}, q_{konst\_int}\}$	$\{q_{11}\}$ (zásek)

LA sa zasekol v stave $\{q_{11}\}$, čo nie je akceptačný stav. Preto začne hľadať najbližší predchádzajúci akceptačný stav, ktorým bol stav $\{q_{10}, q_{konst\_int}\}$, v ktorom sa vo vyrovnávacej pamäti nachádzal reťazec `-1`. Dôjde teda k rozpoznaniu reťazca `-1` ako lexémy `konšt_int` a zvyšok vstupu, ktorý bol vo vyrovnávacej pamäti (symbol bodka `.`) sa vráti na vstup, teda na vstupe zostal reťazec `..25` a výpočet pokračuje:

Symbol vo VP	ε
Stav	$\{q_0, ..., q_9\}$ (zásek)

LA sa znovu zasekol, pretože zo stavu $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9\}$ nie je v DKA definovaný prechod na symbol `.` (bodka). V tomto prípade sa však nepodarí nájsť najbližší predchádzajúci akceptačný stav, takže to znamená, že sa aktuálne na vstupe nachádza taký reťazec, v ktorom **nie je možné rozpoznať** žiadnu z platných lexém a dochádza k tzv. **lexikálnej chybe**.

Vidíme, že vstupný reťazec bol rozdelený na lexémy nasledovným spôsobom:

`i1`	`=`	`1`	`+`	`-1`	`..25`
`identifikátor`	`=`	`konšt_int`	`+`	`konšt_int`	**Lexikálna chyba**

3. Reťazec `-1;--1PREM1+5.1`

Symbol vo VP	ε	`-`	`-1`
Stav	$\{q_0, ..., q_9\}$	$\{q_-, q_8, q_9\}$	$\{q_{10}, q_{konst\_int}\}$ (zásek)

LA sa zasekol v stave $\{q_{10}, q_{konst\_int}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `konšt_int`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`;`
Stav	$\{q_0, ..., q_9\}$	$\{q_;\}$ (zásek)

LA sa zasekol v stave $\{q_;\}$, čo je stav, v ktorom sa rozpoznáva lexéma `;` (bodkočiarka). LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`-`
Stav	$\{q_0, ..., q_9\}$	$\{q_-, q_8, q_9\}$ (zásek)

LA sa zasekol v stave $\{q_-, q_8, q_9\}$, čo je stav, v ktorom sa rozpoznáva lexéma `-`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`-`	`-1`
Stav	$\{q_0, ..., q_9\}$	$\{q_-, q_8, q_9\}$	$\{q_{10}, q_{konst\_int}\}$ (zásek)

LA sa zasekol v stave $\{q_{10}, q_{konst\_int}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `konst_int`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`P`	`PR`	`PRE`	`PREM`	`PREM1`
Stav	$\{q_0, ..., q_9\}$	$\{q_{id}\}$	$\{q_{id}\}$	$\{q_{id}\}$	$\{q_{id}\}$	$\{q_{id}\}$ (zásek)

LA sa zasekol v stave $\{q_{id}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `identifikátor`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`+`	`+5`	`+5.`	`+5.1`
Stav	$\{q_0, ..., q_9\}$	$\{q_+, q_8, q_9\}$	$\{q_{10}, q_{konst\_int}\}$	$\{q_{11}\}$	$\{q_{konst\_real}\}$ (zásek)

LA sa zasekol v stave $\{q_{konst\_real}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `konst_real`. LA vyprázdni vyrovnávaciu pamäť a keďže vstup bol celý prečítaný, činnosť lexikálneho analyzátora končí. Vidíme, že vstupný reťazec bol rozdelený na lexémy nasledovným spôsobom:

`-1`	`;`	`-`	`-1`	`PREM1`	`+5.1`
`konšt_int`	`;`	`-`	`konšt_int`	`identifikátor`	`konšt_real`

Úloha č. 6.2.2 Použite lexikálny analyzátor v tvare DKA, ktorý pomocou akceptačných stavov rozpoznáva nasledujúce lexémy popísané príslušnými regulárnymi výrazmi

1.	`if`	(if)
2.	`int`	(int)
3.	`=`	$(=)$
4.	`==`	$(==)$
5.	`identifikátor`	$(a\|b\|...\|z\|A\|B\|...\|Z)(a\|b\|...\|z\|A\|B\|...\|Z)^*$
6.	`medzera`	(␣)

na tokenizáciu (lexikálnu analýzu) reťazcov:

1. `int if intif`
2. `if a==in=b`
3. `A===B11`

Riešenie:
Príslušný DKA sme zostrojili v úlohe 6.1.2, pozri obrázok 6.4. Spracovanie reťazcov:

1. Reťazec `int if intif`

   LA číta vstup a posúva sa po stavoch † kým sa nezasekne:

Symbol vo VP	ε	`i`	`in`	`int`
Stav	$\{q_0, ..., q_6\}$	$\{q_7, q_8, q_{id}\}$	$\{q_9, q_{id}\}$	$\{q_{int}, q_{id}\}$ (zásek)

   LA sa zasekol v stave $\{q_{int}, q_{id}\}$, čo je stav, v ktorom sa **prioritne** rozpoznáva lexéma `int`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	␣
Stav	$\{q_0, ..., q_6\}$	$\{q_␣\}$ (zásek)

   LA sa zasekol v stave $\{q_␣\}$, čo je stav, v ktorom sa rozpoznáva lexéma `medzera`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`i`	`if`
Stav	$\{q_0, ..., q_6\}$	$\{q_7, q_8, q_{id}\}$	$\{q_{if}, q_{id}\}$ (zásek)

   LA sa zasekol v stave $\{q_{if}, q_{id}\}$, čo je stav, v ktorom sa **prioritne** rozpoznáva lexéma `if`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	␣
Stav	$\{q_0, ..., q_6\}$	$\{q_␣\}$ (zásek)

   LA sa zasekol v stave $\{q_␣\}$, čo je stav, v ktorom sa rozpoznáva lexéma `medzera`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`i`	`in`	`int`	`inti`	`intif`
Stav	$\{q_0, ..., q_6\}$	$\{q_7, q_8, q_{id}\}$	$\{q_9, q_{id}\}$	$\{q_{int}, q_{id}\}$	$\{q_{id}\}$	$\{q_{id}\}$ (zásek)

   LA sa zasekol v stave $\{q_{id}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `identifikátor`. LA vyprázdni vyrovnávaciu pamäť a keďže vstup bol celý prečítaný, činnosť lexikálneho analyzátora končí. Vidíme, že vstupný reťazec bol rozdelený na lexémy nasledovným spôsobom:

`int`	␣	`if`	␣	`intif`
`int`	`medzera`	`if`	`medzera`	`identifikátor`

†Počiatočný stav v DKA je označený ako $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$, z dôvodu obmedzeného miesta ho uvádzame len ako $\{q_0, ..., q_6\}$

2. Reťazec `if a==in=b`

Symbol vo VP	ε	`i`	`if`
Stav	$\{q_0, ..., q_6\}$	$\{q_7, q_8, q_{id}\}$	$\{q_{if}, q_{id}\}$ (zásek)

LA sa zasekol v stave $\{q_{if}, q_{id}\}$, čo je stav, v ktorom sa **prioritne** rozpoznáva lexéma `if`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	␣
Stav	$\{q_0, ..., q_6\}$	$\{q_{␣}\}$ (zásek)

LA sa zasekol v stave $\{q_{␣}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `medzera`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`a`
Stav	$\{q_0, ..., q_6\}$	$\{q_{id}\}$ (zásek)

LA sa zasekol v stave $\{q_{id}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `identifikátor`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`=`	`==`
Stav	$\{q_0, ..., q_6\}$	$\{q_{10}, q_=\}$	$\{q_{==}\}$ (zásek)

LA sa zasekol v stave $\{q_{==}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `==`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`i`	`in`
Stav	$\{q_0, ..., q_6\}$	$\{q_7, q_8, q_{id}\}$	$\{q_9, q_{id}\}$ (zásek)

LA sa zasekol v stave $\{q_9, q_{id}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `identifikátor`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`=`
Stav	$\{q_0, ..., q_6\}$	$\{q_{10}, q_=\}$ (zásek)

LA sa zasekol v stave $\{q_{10}, q_=\}$, čo je stav, v ktorom sa rozpoznáva lexéma `=`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	`b`
Stav	$\{q_0, ..., q_6\}$	$\{q_{id}\}$ (zásek)

LA sa zasekol v stave $\{q_{id}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `identifikátor`. LA vyprázdni vyrovnávaciu pamäť a keďže vstup bol celý prečítaný, činnosť lexikálneho analyzátora končí. Vidíme, že vstupný reťazec bol rozdelený na lexémy nasledovným spôsobom:

if	␣	a	==	in	=	b
if	medzera	identifikátor	==	identifikátor	=	identifikátor

3. Reťazec `A===B11`

Symbol vo VP	ε	A
Stav	$\{q_0, ..., q_6\}$	$\{q_{id}\}$ (zásek)

LA sa zasekol v stave $\{q_{id}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `identifikátor`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	=	==
Stav	$\{q_0, ..., q_6\}$	$\{q_{10}, q_{=}\}$	$\{q_{==}\}$ (zásek)

LA sa zasekol v stave $\{q_{==}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `==`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	=
Stav	$\{q_0, ..., q_6\}$	$\{q_{10}, q_{=}\}$ (zásek)

LA sa zasekol v stave $\{q_{10}, q_{=}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `=`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε	B
Stav	$\{q_0, ..., q_6\}$	$\{q_{id}\}$ (zásek)

LA sa zasekol v stave $\{q_{id}\}$, čo je stav, v ktorom sa rozpoznáva lexéma `identifikátor`. LA vyprázdni vyrovnávaciu pamäť, vráti sa do počiatočného stavu a výpočet pokračuje so zvyškom vstupu:

Symbol vo VP	ε
Stav	$\{q_0, ..., q_6\}$ (zásek)

LA sa zasekol, pretože zo stavu $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$ nie je v DKA definovaný prechod na symbol `1`. V tomto prípade sa nepodarí nájsť najbližší predchádzajúci akceptačný stav, takže to znamená, že dochádza k tzv. **lexikálnej chybe**. Vidíme, že vstupný reťazec bol rozdelený na lexémy nasledovným spôsobom:

A	==	=	B	11
identifikátor	==	=	identifikátor	**Lexikálna chyba**

Kapitola 7

Syntaktická analýza top-down

7.1 Konštrukcia $LL(1)$ syntaktického analyzátora

Úloha č. 7.1.1 Je daná bezkontextová gramatika $G = (\{S, A, B\}, \{a, b, d\}, P, S)$, ktorej pravidlá P sú:

1. $S \to BAd$
2. $A \to b$
3. $A \to \varepsilon$
4. $B \to a$
5. $B \to \varepsilon$

Zostrojte množiny $PREDICT$ pre jednotlivé pravidlá gramatiky, nájdite rozkladovú tabuľku príslušného $LL(1)$ syntaktického analyzátora a určte, či ide o $LL(1)$ gramatiku.

Riešenie:
Konštrukcia množín $PREDICT$ a rozkladovej tabuľky $LL(1)$ sa vykonáva pre redukovanú gramatiku. Keďže zadaná gramatika je už v redukovanom tvare, môžeme pokračovať s konštrukciou množín $PREDICT$ pre jednotlivé pravidlá. Ak je daná redukovaná bezkontextová gramatika $G = (N, T, P, S)$, potom množina $PREDICT$ pravidla $A \to \alpha$, $A \in N, \alpha \in (N \cup T)^*$ je definovaná predpisom:

$$PREDICT(A \to \alpha) = \begin{cases} FIRST(\alpha) & \text{ak } \varepsilon \notin FIRST(\alpha), \\ (FIRST(\alpha) \setminus \{\varepsilon\}) \cup FOLLOW(A) & \text{ak } \varepsilon \in FIRST(\alpha). \end{cases}$$

Pri konštrukcii množín $PREDICT$ teda potrebujeme poznať množiny $FIRST$ a $FOLLOW$ pre jednotlivé neterminály gramatiky. V tomto prípade majú tieto množiny nasledovné hodnoty:

	S	A	B
$FIRST$	$\{a, b, d\}$	$\{b, \varepsilon\}$	$\{a, \varepsilon\}$
$FOLLOW$	$\{\varepsilon\}$	$\{d\}$	$\{b, d\}$

Množiny $PREDICT$ pravidiel gramatiky teda určíme nasledovným spôsobom:

1. $PREDICT(S \to BAd) = \{a, b, d\}$, pretože:
   - $FIRST(BAd) = \{a, b, d\}$.
   - Keďže $\varepsilon \notin FIRST(BAd)$, tak $PREDICT(S \to BAd) = FIRST(BAd) = \{a, b, d\}$.
2. $PREDICT(A \to b) = \{b\}$, pretože:
   - $FIRST(b) = \{b\}$.
   - Keďže $\varepsilon \notin FIRST(b)$, tak $PREDICT(A \to b) = FIRST(b) = \{b\}$.
3. $PREDICT(A \to \varepsilon) = \{d\}$, pretože:
   - $FIRST(\varepsilon) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak $PREDICT(A \to \varepsilon) = (FIRST(\varepsilon) \setminus \{\varepsilon\}) \cup FOLLOW(A) = \emptyset \cup FOLLOW(A) = FOLLOW(A) = \{d\}$.
4. $PREDICT(B \to a) = \{a\}$, pretože:
   - $FIRST(a) = \{a\}$.
   - Keďže $\varepsilon \notin FIRST(a)$, tak $PREDICT(B \to a) = FIRST(a) = \{a\}$.
5. $PREDICT(B \to \varepsilon) = \{b, d\}$, pretože:
   - $FIRST(\varepsilon) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak $PREDICT(B \to \varepsilon) = (FIRST(\varepsilon) \setminus \{\varepsilon\}) \cup FOLLOW(B) = \emptyset \cup FOLLOW(B) = FOLLOW(B) = \{b, d\}$.

Rozkladovú tabuľku $LL(1)$ syntaktického analyzátora skonštruujeme podľa množín $PREDICT$ nasledovným spôsobom:

Ak $t \in PREDICT(A \to \alpha)$, potom $RT[A, t]$ obsahuje pravidlo $A \to \alpha$.

Na základe nami nájdených množín $PREDICT$ teda vieme povedať, že:

1. $PREDICT(S \to BAd) = \{a, b, d\}$
   - Keďže $a, b, d \in PREDICT(S \to BAd)$, tak v rozkladovej tabuľke sa na pozíciách $RT[S, a], RT[S, b], RT[S, d]$ bude nachádzať pravidlo $S \to BAd$.
2. $PREDICT(A \to b) = \{b\}$
   - Keďže $b \in PREDICT(A \to b)$, tak v rozkladovej tabuľke sa na pozícii $RT[A, b]$ bude nachádzať pravidlo $A \to b$.

3. $PREDICT(A \to \varepsilon) = \{d\}$
   - Keďže $d \in PREDICT(A \to \varepsilon)$, tak v rozkladovej tabuľke sa na pozícii $RT[A, d]$ bude nachádzať pravidlo $A \to \varepsilon$.
4. $PREDICT(B \to a) = \{a\}$
   - Keďže $a \in PREDICT(B \to a)$, tak v rozkladovej tabuľke sa na pozícii $RT[B, a]$ bude nachádzať pravidlo $B \to a$.
5. $PREDICT(B \to \varepsilon) = \{b, d\}$
   - Keďže $b, d \in PREDICT(B \to \varepsilon)$, tak v rozkladovej tabuľke sa na pozíciách $RT[B, b], RT[B, d]$ bude nachádzať pravidlo $B \to \varepsilon$.

Výsledná rozkladová tabuľka $LL(1)$ analyzátora má nasledovný tvar:

RT	a	b	d	ε
S	$S \to BAd$	$S \to BAd$	$S \to BAd$	
A		$A \to b$	$A \to \varepsilon$	
B	$B \to a$	$B \to \varepsilon$	$B \to \varepsilon$	

Gramatika je $LL(1)$ gramatikou práve vtedy, keď sa v jej rozkladovej tabuľke na žiadnej pozícii nenachádza konflikt, teda každá bunka rozkladovej tabuľky obsahuje najviac jedno pravidlo gramatiky. Keďže v nami zostrojenej rozkladovej tabuľke sa v každej bunke nachádza najviac jedno pravidlo, konštatujeme, že zadaná gramatika **je** $LL(1)$ **gramatikou**.

Úloha č. 7.1.2 Je daná bezkontextová gramatika $G = (\{S, A, C\}, \{a, b\}, P, S)$, ktorej pravidlá P sú:

1. $S \to aAb$
2. $A \to Ca$
3. $C \to aA$
4. $C \to aS$
5. $C \to b$

Zostrojte množiny $PREDICT$ pre jednotlivé pravidlá gramatiky, nájdite rozkladovú tabuľku príslušného $LL(1)$ syntaktického analyzátora a určte, či ide o $LL(1)$ gramatiku.

Riešenie:
Keďže gramatika je už redukovaná, môžeme pristúpiť k tvorbe množín $PREDICT$. Množiny $FIRST$ a $FOLLOW$ pre neterminály gramatiky:

	S	A	C
$FIRST$	$\{a\}$	$\{a, b\}$	$\{a, b\}$
$FOLLOW$	$\{\varepsilon, a\}$	$\{a, b\}$	$\{a\}$

Množiny $PREDICT$ pravidiel gramatiky teda určíme nasledovným spôsobom:

1. $PREDICT(S \to aAb) = \{a\}$, pretože:
   - $FIRST(aAb) = \{a\}$.
   - Keďže $\varepsilon \notin FIRST(aAb)$, tak $PREDICT(S \to aAb) = FIRST(aAb) = \{a\}$.
2. $PREDICT(A \to Ca) = \{a, b\}$, pretože:
   - $FIRST(Ca) = \{a, b\}$.
   - Keďže $\varepsilon \notin FIRST(Ca)$, tak $PREDICT(A \to Ca) = FIRST(Ca) = \{a, b\}$.
3. $PREDICT(C \to aA) = \{a\}$, pretože:
   - $FIRST(aA) = \{a\}$.
   - Keďže $\varepsilon \notin FIRST(aA)$, tak $PREDICT(C \to aA) = FIRST(aA) = \{a\}$.
4. $PREDICT(C \to aS) = \{a\}$, pretože:
   - $FIRST(aS) = \{a\}$.
   - Keďže $\varepsilon \notin FIRST(aS)$, tak $PREDICT(C \to aS) = FIRST(aS) = \{a\}$.
5. $PREDICT(C \to b) = \{b\}$, pretože:
   - $FIRST(b) = \{b\}$.
   - Keďže $\varepsilon \notin FIRST(b)$, tak $PREDICT(C \to b) = FIRST(b) = \{b\}$.

Na základe množín $PREDICT$ skonštruujeme rozkladovú tabuľku:

1. $PREDICT(S \to aAb) = \{a\}$
   - Keďže $a \in PREDICT(S \to aAb)$, tak v rozkladovej tabuľke sa na pozícii $RT[S, a]$ bude nachádzať pravidlo $S \to aAb$.
2. $PREDICT(A \to Ca) = \{a, b\}$
   - Keďže $a, b \in PREDICT(A \to Ca)$, tak v rozkladovej tabuľke sa na pozíciách $RT[A, a], RT[A, b]$ bude nachádzať pravidlo $A \to Ca$.
3. $PREDICT(C \to aA) = \{a\}$
   - Keďže $a \in PREDICT(C \to aA)$, tak v rozkladovej tabuľke sa na pozícii $RT[C, a]$ bude nachádzať pravidlo $C \to aA$.

4. $PREDICT(C \rightarrow aS) = \{a\}$
   - Keďže $a \in PREDICT(C \rightarrow aS)$, tak v rozkladovej tabuľke sa na pozícii $RT[C, a]$ bude nachádzať pravidlo $C \rightarrow aS$.
5. $PREDICT(C \rightarrow b) = \{b\}$
   - Keďže $b \in PREDICT(C \rightarrow b)$, tak v rozkladovej tabuľke sa na pozícii $RT[C, b]$ bude nachádzať pravidlo $C \rightarrow b$.

Výsledná rozkladová tabuľka $LL(1)$-syntaktického analyzátora má nasledovný tvar:

RT	a	b	ε
S	$S \rightarrow aAb$		
A	$A \rightarrow Ca$	$A \rightarrow Ca$	
C	$C \rightarrow aA$ $C \rightarrow aS$	$C \rightarrow b$	

Keďže v rozkladovej tabuľke sa na pozícii $RT[C, a]$ nachádza viac ako jedno pravidlo, konkrétne dve pravidlá $C \rightarrow aA$ a $C \rightarrow aS$, v rozkladovej tabuľke je na tejto pozícii tzv. konflikt a príslušná gramatika **nie je** $LL(1)$-gramatikou.

Úloha č. 7.1.3 Je daná bezkontextová gramatika G = ({<príkazy>, <príkaz>, <elseČasť>, <podmienka>}, {`if`, `then`, `else`, `fi`, `p1`, `p2`, `cond`}, P, <príkazy>), ktorej pravidlá P sú:

1. <príkazy>→<príkaz><príkazy>
2. <príkazy>→ ε
3. <príkaz>→ `if` <podmienka> `then` <príkazy><elseČasť> `fi`
4. <príkaz>→ `p1`
5. <príkaz>→ `p2`
6. <elseČasť>→ `else` <príkazy>
7. <elseČasť>→ ε
8. <podmienka>→ `cond`

Zostrojte množiny $PREDICT$ pre jednotlivé pravidlá gramatiky, nájdite rozkladovú tabuľku príslušného $LL(1)$-syntaktického analyzátora a určte, či ide o $LL(1)$-gramatiku.

Riešenie:
Keďže gramatika je redukovaná, môžeme pristúpiť k tvorbe množín $PREDICT$. Množiny $FIRST$ a $FOLLOW$ pre neterminály gramatiky:

	<príkazy>	<príkaz>	<podmienka>	<elseČasť>
$FIRST$	{ε,`if`,`p1`,`p2`}	{`if`,`p1`,`p2`}	{`cond`}	{ε,`else`}
$FOLLOW$	{ε,`else`,`fi`}	{ε,`if`,`else`,`fi`,`p1`,`p2`}	{`then`}	{`fi`}

Množiny $PREDICT$ pravidiel gramatiky:

1. $PREDICT(<\text{príkazy}>\to<\text{príkaz}><\text{príkazy}>) = \{\texttt{if},\texttt{p1},\texttt{p2}\}$, pretože:
   - $FIRST(<\text{príkaz}><\text{príkazy}>) = \{\texttt{if},\texttt{p1},\texttt{p2}\}$.
   - Keďže $\varepsilon \notin FIRST(<\text{príkaz}><\text{príkazy}>)$, tak $PREDICT(<\text{príkazy}>\to<\text{príkaz}><\text{príkazy}>) = FIRST(<\text{príkaz}><\text{príkazy}>) = \{\texttt{if},\texttt{p1},\texttt{p2}\}$.
2. $PREDICT(<\text{príkazy}>\to \varepsilon) = \{\varepsilon, \texttt{else},\texttt{fi}\}$, pretože:
   - $FIRST(\varepsilon) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak $PREDICT(<\text{príkazy}>\to \varepsilon) = (FIRST(\varepsilon) \setminus \{\varepsilon\}) \cup FOLLOW(<\text{príkazy}>) = \emptyset \cup FOLLOW(<\text{príkazy}>) = \{\varepsilon, \texttt{else},\texttt{fi}\}$.
3. $PREDICT(<\text{príkaz}>\to \texttt{if}\ <\text{podmienka}>\ \texttt{then}\ <\text{príkazy}><\text{elseČast'}>\ \texttt{fi}) = \{\texttt{if}\}$, pretože:
   - $FIRST(\texttt{if}\ <\text{podmienka}>\ \texttt{then}\ <\text{príkazy}><\text{elseČast'}>\ \texttt{fi}) = \{\texttt{if}\}$.
   - Keďže $\varepsilon \notin FIRST(\texttt{if}\ <\text{podmienka}>\ \texttt{then}\ <\text{príkazy}><\text{elseČast'}>\ \texttt{fi})$, tak $PREDICT(<\text{príkaz}>\to \texttt{if}\ <\text{podmienka}>\ \texttt{then}\ <\text{príkazy}><\text{elseČast'}>\ \texttt{fi}) = FIRST(\texttt{if}\ <\text{podmienka}>\ \texttt{then}\ <\text{príkazy}><\text{elseČast'}>\ \texttt{fi}) = \{\texttt{if}\}$.
4. $PREDICT(<\text{príkaz}>\to \texttt{p1}) = \{\texttt{p1}\}$, pretože:
   - $FIRST(\texttt{p1}) = \{\texttt{p1}\}$.
   - Keďže $\varepsilon \notin FIRST(\texttt{p1})$, tak $PREDICT(<\text{príkaz}>\to \texttt{p1}) = FIRST(\texttt{p1}) = \{\texttt{p1}\}$.
5. $PREDICT(<\text{príkaz}>\to \texttt{p2}) = \{\texttt{p2}\}$, pretože:
   - $FIRST(\texttt{p2}) = \{\texttt{p2}\}$.
   - Keďže $\varepsilon \notin FIRST(\texttt{p2})$, tak $PREDICT(<\text{príkaz}>\to \texttt{p2}) = FIRST(\texttt{p2}) = \{\texttt{p2}\}$.
6. $PREDICT(<\text{elseČast'}>\to \texttt{else}\ <\text{príkazy}>) = \{\texttt{else}\}$, pretože:
   - $FIRST(\texttt{else}\ <\text{príkazy}>) = \{\texttt{else}\}$.
   - Keďže $\varepsilon \notin FIRST(\texttt{else}\ <\text{príkazy}>)$, tak $PREDICT(<\text{elseČast'}>\to \texttt{else}\ <\text{príkazy}>) = FIRST(\texttt{else}\ <\text{príkazy}>) = \{\texttt{else}\}$.
7. $PREDICT(<\text{elseČast'}>\to \varepsilon) = \{\texttt{fi}\}$, pretože:
   - $FIRST(\varepsilon) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak $PREDICT(<\text{elseČast'}>\to \varepsilon) = (FIRST(\varepsilon) \setminus \{\varepsilon\}) \cup FOLLOW(<\text{elseČast'}>) = \{\texttt{fi}\}$.

8. $PREDICT$(<podmienka>$\rightarrow$ `cond`) = {`cond`}, pretože:
   - $FIRST$(`cond`) = {`cond`}.
   - Keďže $\varepsilon \notin FIRST$(`cond`), tak $PREDICT$(<podmienka>$\rightarrow$ `cond`) = $FIRST$(`cond`) = {`cond`}.

Na základe nami nájdených množín $PREDICT$ skonštruujeme nasledovnú rozkladovú tabuľku, v ktorej pre prehľadnosť uvádzame len poradové čísla pravidiel:

RT	`if`	`then`	`else`	`p1`	`p2`	`cond`	`fi`	ε
<príkazy>	1		2	1	1		2	2
<príkaz>	3			4	5			
<elseČasť'>			6				7	
<podmienka>						8		

Keďže v rozkladovej tabuľke sa na všetkých pozíciách nachádza najviac jedno pravidlo, príslušná gramatika **je $LL(1)$-gramatikou**.

Úloha č. 7.1.4 Je daná bezkontextová gramatika $G = (\{S, A, B, C, D\}, \{b, c, d\}, P, S)$, ktorej pravidlá P sú:

1. $S \rightarrow AdDC$
2. $A \rightarrow BCD$
3. $B \rightarrow D$
4. $B \rightarrow bB$
5. $C \rightarrow cD$
6. $C \rightarrow D$
7. $D \rightarrow \varepsilon$

Zostrojte množiny $PREDICT$ pre jednotlivé pravidlá gramatiky, nájdite rozkladovú tabuľku príslušného $LL(1)$-syntaktického analyzátora a určte, či ide o $LL(1)$-gramatiku.

Riešenie:
Keďže gramatika už je redukovaná, nemusíme ju redukovať a môžeme pristúpiť k tvorbe množín $PREDICT$. Množiny $FIRST$ a $FOLLOW$ pre neterminály gramatiky majú nasledovné hodnoty:

	S	A	B	C	D
$FIRST$	$\{b, c, d\}$	$\{b, c, \varepsilon\}$	$\{b, \varepsilon\}$	$\{c, \varepsilon\}$	$\{\varepsilon\}$
$FOLLOW$	$\{\varepsilon\}$	$\{d\}$	$\{c, d\}$	$\{\varepsilon, d\}$	$\{\varepsilon, c, d\}$

Množiny $PREDICT$ pravidiel gramatiky:

1. $PREDICT(S \to AdDC) = \{b, c, d\}$, pretože:
   - $FIRST(AdDC) = \{b, c, d\}$.
   - Keďže $\varepsilon \notin FIRST(AdDC)$, tak $PREDICT(S \to AdDC) = FIRST(AdDC) = \{b, c, d\}$.
2. $PREDICT(A \to BCD) = \{b, c, d\}$, pretože:
   - $FIRST(BCD) = \{\varepsilon, b, c\}$.
   - Keďže $\varepsilon \in FIRST(BCD)$, tak $PREDICT(A \to BCD) = (FIRST(BCD) \setminus \{\varepsilon\}) \cup FOLLOW(A) = \{b, c\} \cup FOLLOW(A) = \{b, c\} \cup \{d\} = \{b, c, d\}$.
3. $PREDICT(B \to D) = \{c, d\}$, pretože:
   - $FIRST(D) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(D)$, tak $PREDICT(B \to D) = (FIRST(D) \setminus \{\varepsilon\}) \cup FOLLOW(B) = \emptyset \cup FOLLOW(B) = \{c, d\}$.
4. $PREDICT(B \to bB) = \{b\}$, pretože:
   - $FIRST(bB) = \{b\}$.
   - Keďže $\varepsilon \notin FIRST(bB)$, tak $PREDICT(B \to bB) = FIRST(bB) = \{b\}$.
5. $PREDICT(C \to cD) = \{c\}$, pretože:
   - $FIRST(cD) = \{c\}$.
   - Keďže $\varepsilon \notin FIRST(cD)$, tak $PREDICT(C \to cD) = FIRST(cD) = \{c\}$.
6. $PREDICT(C \to D) = \{\varepsilon, d\}$, pretože:
   - $FIRST(D) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(D)$, tak $PREDICT(C \to D) = (FIRST(D) \setminus \{\varepsilon\}) \cup FOLLOW(C) = \emptyset \cup FOLLOW(C) = \{\varepsilon, d\}$.
7. $PREDICT(D \to \varepsilon) = \{\varepsilon, c, d\}$, pretože:
   - $FIRST(D) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak $PREDICT(D \to \varepsilon) = (FIRST(\varepsilon) \setminus \{\varepsilon\}) \cup FOLLOW(D) = \emptyset \cup FOLLOW(D) = \{\varepsilon, c, d\}$.

Na základe nami nájdených množín $PREDICT$ skonštruujeme nasledovnú rozkladovú tabuľku:

RT	b	c	d	ε
S	$S \to AdDC$	$S \to AdDC$	$S \to AdDC$	
A	$A \to BCD$	$A \to BCD$	$A \to BCD$	
B	$B \to bB$	$B \to D$	$B \to D$	
C		$C \to cD$	$C \to D$	$C \to D$
D		$D \to \varepsilon$	$D \to \varepsilon$	$D \to \varepsilon$

Keďže v rozkladovej tabuľke sa na všetkých pozíciách nachádza najviac jedno pravidlo, príslušná gramatika **je $LL(1)$-gramatikou**.

Úloha č. 7.1.5 Je daná bezkontextová gramatika $G = (\{S, A, B\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

1. $S \to AB$
2. $S \to cSc$
3. $A \to aAb$
4. $A \to \varepsilon$
5. $B \to aBa$
6. $B \to bBb$
7. $B \to \varepsilon$

Zostrojte množiny $PREDICT$ pre jednotlivé pravidlá gramatiky, nájdite rozkladovú tabuľku príslušného $LL(1)$-syntaktického analyzátora a určte, či ide o $LL(1)$-gramatiku.

Riešenie:
Keďže gramatika je redukovaná, môžeme pristúpiť k tvorbe množín $PREDICT$. Množiny $FIRST$ a $FOLLOW$ pre neterminály gramatiky:

	S	A	B
$FIRST$	$\{a, b, c, \varepsilon\}$	$\{a, \varepsilon\}$	$\{a, b, \varepsilon\}$
$FOLLOW$	$\{\varepsilon, c\}$	$\{\varepsilon, a, b, c\}$	$\{\varepsilon, a, b, c\}$

Množiny $PREDICT$ pravidiel gramatiky:

1. $PREDICT(S \to AB) = \{a, b, c, \varepsilon\}$, pretože:
   - $FIRST(AB) = \{a, b, \varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(AB)$, tak $PREDICT(S \to AB) =$ $(FIRST(AB) \setminus \{\varepsilon\}) \cup FOLLOW(S) = \{a, b\} \cup \{\varepsilon, c\} = \{a, b, c, \varepsilon\}$.

2. $PREDICT(S \to cSc) = \{c\}$, pretože:
   - $FIRST(cSc) = \{c\}$.
   - Keďže $\varepsilon \notin FIRST(cSc)$, tak $PREDICT(S \to cSc) = FIRST(cSc) = \{c\}$.
3. $PREDICT(A \to aAb) = \{a\}$, pretože:
   - $FIRST(aAb) = \{a\}$.
   - Keďže $\varepsilon \notin FIRST(aAb)$, tak $PREDICT(A \to aAb) = FIRST(aAb) = \{a\}$.
4. $PREDICT(A \to \varepsilon) = \{\varepsilon, a, b, c\}$, pretože:
   - $FIRST(\varepsilon) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak $PREDICT(A \to \varepsilon) = (FIRST(\varepsilon) \setminus \{\varepsilon\}) \cup FOLLOW(A) = \emptyset \cup FOLLOW(A) = \{\varepsilon, a, b, c\}$.
5. $PREDICT(B \to aBa) = \{a\}$, pretože:
   - $FIRST(aBa) = \{a\}$.
   - Keďže $\varepsilon \notin FIRST(aBa)$, tak $PREDICT(B \to aBa) = FIRST(aBa) = \{a\}$.
6. $PREDICT(B \to bBb) = \{b\}$, pretože:
   - $FIRST(bBb) = \{b\}$.
   - Keďže $\varepsilon \notin FIRST(bBb)$, tak $PREDICT(B \to bBb) = FIRST(bBb) = \{b\}$.
7. $PREDICT(B \to \varepsilon) = \{\varepsilon, a, b, c\}$, pretože:
   - $FIRST(\varepsilon) = \{\varepsilon\}$.
   - Keďže $\varepsilon \in FIRST(\varepsilon)$, tak $PREDICT(B \to \varepsilon) = (FIRST(\varepsilon) \setminus \{\varepsilon\}) \cup FOLLOW(B) = \emptyset \cup FOLLOW(B) = \{\varepsilon, a, b, c\}$.

Na základe nami nájdených množín $PREDICT$ skonštruujeme nasledovnú rozkladovú tabuľku, v ktorej uvádzame len poradové čísla pravidiel:

RT	a	b	c	ε
S	1	1	1,2	1
A	3,4	4	4	4
B	5,7	6,7	7	7

Keďže v rozkladovej tabuľke existujú konflikty, konkrétne na pozíciách $RT[S, c], RT[A, a], RT[B, a], RT[B, b]$, zadaná gramatika **nie je** $LL(1)$**-gramatikou**.

7.2 Syntaktická analýza pomocou $LL(1)$-syntaktických analyzátorov

Úloha č. 7.2.1 Je daná bezkontextová gramatika $G = (\{S, A, B\}, \{a, b, d\}, P, S)$, ktorej pravidlá P sú:

1. $S \to BAd$
2. $A \to b$
3. $A \to \varepsilon$
4. $B \to a$
5. $B \to \varepsilon$

Pomocou $LL(1)$-syntaktického analyzátora zistite, či majú nasledovné ret'azce v uvedenej gramatike deriváciu. Ak derivácia ret'azca existuje, zistite, ako vyzerá **l'avá derivácia** príslušného ret'azca:

1. abd
2. aad
3. ba
4. ab

Riešenie:
Ked'že o danej gramatike sme v úlohe č. 7.1.1 zistili, že ide o $LL(1)$-gramatiku, existuje pre ňu príslušný $LL(1)$-syntaktický analyzátor, ktorého rozkladová tabul'ka má tvar:

RT	a	b	d	ε
S	$S \to BAd$	$S \to BAd$	$S \to BAd$	
A		$A \to b$	$A \to \varepsilon$	
B	$B \to a$	$B \to \varepsilon$	$B \to \varepsilon$	

$LL(1)$-syntaktický analyzátor má formu špecifického zásobníkového automatu, ktorého vstupom je ret'azec, o ktorom chceme zistit', či má deriváciu a ktorého počiatočným zásobníkovým symbolom je počiatočný neterminál gramatiky S. Následne syntaktický analyzátor opakuje nasledovné činnosti:

- Ak je na vrchu zásobníka terminálny symbol vykoná sa operácia **porovnanie**. Ak sa terminál na vrchu zásobníka **zhoduje** s aktuálne čítaným vstupným symbolom, terminál zo zásobníka sa odstráni, vstupný symbol sa označí za prečítaný a v d'alšom kroku sa pracuje s nasledovným vstupným symbolom. Ak sa terminál na vrchu zásobníka **nezhoduje** s aktuálne čítaným vstupným symbolom, znamená to **syntaktickú chybu** a vstupný ret'azec **nemá** v uvedenej gramatike deriváciu.

- Ak je na vrchu zásobníka neterminálny symbol, povedzme A, vykoná sa operácia **expanzia**. Na základe aktuálneho vstupného symbolu, povedzme t, sa vykoná expanzia podľa toho pravidla, ktoré sa nachádza v rozkladovej tabuľke na pozícii $RT[A,t]$. Pri tejto expanzii sa **odstráni** zo zásobníka neterminál A a nahradí sa príslušnou pravou stranou podľa expandovaného pravidla tak, že prvý symbol pravej strany bude navrchu zásobníka. Pri expanzii sa vstupný symbol **neoznačí za prečítaný**, t. j. aj v ďalšom kroku zostáva aktuálnym vstupným symbolom.
- Ak sa v rozkladovej tabuľke na pozícii $RT[A,t]$ **žiadne pravidlo nenachádza**, signalizuje to **syntaktickú chybu** a vstupný reťazec **nemá** v uvedenej gramatike deriváciu.
- Ak sa vyprázdni celý zásobník, avšak na vstupe zostali ešte neprečítané vstupné symboly, signalizuje to **syntaktickú chybu** a vstupný reťazec **nemá** v uvedenej gramatike deriváciu.
- Ak výpočet syntaktického analyzátora dospeje do situácie, v ktorej bol vstup **celý prečítaný** a **zásobník celý vyprázdnený**, signalizuje to **akceptáciu vstupného reťazca**, teda že daný reťazec **má v gramatike deriváciu**.
- Postupnosť pravidiel použitých na jednotlivé **expanzie** potom zároveň udáva postupnosť pravidiel, ktoré tvoria **ľavú deriváciu vstupného reťazca**.

1. Syntaktická analýza reťazca *abd*:

   Výpočet znázorníme tak, že uvedieme ešte nespracovanú časť vstupného reťazca, pričom v danom momente vidí syntaktický analyzátor prvý nespracovaný symbol zľava. Obsah zásobníka uvedieme *sklopený doľava*, kde prvý symbol zľava je zároveň symbol navrchu zásobníka.

Zvyšok vstupu	Zásobník $\sqsupset$	Akcia
abd	S	Expanzia, $S \to BAd$
abd	BAd	Expanzia, $B \to a$
abd	aAd	Porovnanie
bd	Ad	Expanzia, $A \to b$
bd	bd	Porovnanie
d	d	Porovnanie
ε	ε	Akceptácia

   Keďže výpočet syntaktického analyzátora dospel do situácie, v ktorej bol vstup celý prečítaný a zásobník celý vyprázdnený, reťazec *abd* má v uvedenej gramatike deriváciu. Ľavá derivácia tohto reťazca vznikne postupnou aplikáciou pravidiel $S \to BAd$, $B \to a$, $A \to b$.

2. Syntaktická analýza ret'azca aad:

Zvyšok vstupu	Zásobník $\sqsupset$	Akcia
aad	S	Expanzia, $S \to BAd$
aad	BAd	Expanzia, $B \to a$
aad	aAd	Porovnanie
ad	Ad	**Syntaktická chyba**

Vznikla situácia, v ktorej je potrebné expandovat' neterminál A, pričom na vstupe sa nachádza terminál a. V rozkladovej tabuľke sa však na pozícii $RT[A, a]$ **nenachádza** žiadne pravidlo. Znamená to teda, že ret'azec aad **nemá** v danej gramatike deriváciu a došlo k syntaktickej chybe.

3. Syntaktická analýza ret'azca ba:

Zvyšok vstupu	Zásobník $\sqsupset$	Akcia
ba	S	Expanzia, $S \to BAd$
ba	BAd	Expanzia, $B \to \varepsilon$
ba	Ad	Expanzia, $A \to b$
ba	bd	Porovnanie
a	d	**Syntaktická chyba**

Vznikla situácia, v ktorej je na vrchu zásobníka terminálny symbol. V takom prípade sa má vykonat' operácia porovnanie, ktorá je úspešná, ak sa terminál na vrchu zásobníka zhoduje so vstupným symbolom. Avšak vidíme, že na vrchu zásobníka je terminál d a na vstupe symbol a. Znamená to teda, že ret'azec ba **nemá** v danej gramatike deriváciu a došlo k syntaktickej chybe.

4. Syntaktická analýza ret'azca ab:

Zvyšok vstupu	Zásobník $\sqsupset$	Akcia
ab	S	Expanzia, $S \to BAd$
ab	BAd	Expanzia, $B \to a$
ab	aAd	Porovnanie
b	Ad	Expanzia, $A \to b$
b	bd	Porovnanie
ε	d	**Syntaktická chyba**

Znovu vznikla situácia, v ktorej by sa mala vykonat' operácia porovnanie, ked'že na vrchu zásobníka je terminálny symbol. Avšak v tomto momente bol už vstup celý prečítaný, takže nie je možné porovnat' terminálny symbol zo zásobníka so žiadnym vstupným symbolom. Znamená to teda, že ret'azec ab **nemá** v danej gramatike deriváciu a došlo k syntaktickej chybe. Všimnite si teda, že prečítanie celého vstupu je len **nutnou podmienkou** akceptácie a platí, že pre akceptáciu vstupu je v $LL(1)$-syntaktickej analýze potrebné zároveň vyprázdnit' celý zásobník, čo v tomto prípade nie je možné.

Úloha č. 7.2.2 Je daná bezkontextová gramatika $G = (\{S, A, B, C, D\}, \{b, c, d\}, P, S)$, ktorej pravidlá P sú:

1. $S \to AdDC$
2. $A \to BCD$
3. $B \to D$
4. $B \to bB$
5. $C \to cD$
6. $C \to D$
7. $D \to \varepsilon$

Pomocou $LL(1)$-syntaktického analyzátora zistite, či majú nasledovné reťazce v uvedenej gramatike deriváciu. Ak derivácia reťazca existuje, zistite, ako vyzerá **ľavá derivácia** príslušného reťazca:

1. cd
2. dd

Riešenie:

Keďže o danej gramatike sme v úlohe č. 7.1.4 zistili, že ide o $LL(1)$-gramatiku, existuje pre ňu príslušný $LL(1)$-syntaktický analyzátor, ktorého rozkladová tabuľka má tvar:

RT	b	c	d	ε
S	$S \to AdDC$	$S \to AdDC$	$S \to AdDC$	
A	$A \to BCD$	$A \to BCD$	$A \to BCD$	
B	$B \to bB$	$B \to D$	$B \to D$	
C		$C \to cD$	$C \to D$	$C \to D$
D		$D \to \varepsilon$	$D \to \varepsilon$	$D \to \varepsilon$

1. Syntaktická analýza reťazca cd:

Zvyšok vstupu	Zásobník $\sqsupset$	Akcia
cd	S	Expanzia, $S \to AdDC$
cd	$AdDC$	Expanzia, $A \to BCD$
cd	$BCDdDC$	Expanzia, $B \to D$
cd	$DCDdDC$	Expanzia, $D \to \varepsilon$
cd	$CDdDC$	Expanzia, $C \to cD$
cd	$cDDdDC$	Porovnanie
d	$DDdDC$	Expanzia, $D \to \varepsilon$
d	$DdDC$	Expanzia, $D \to \varepsilon$
d	dDC	Porovnanie
ε	DC	Expanzia, $D \to \varepsilon$
ε	C	Expanzia, $C \to D$
ε	D	Expanzia, $D \to \varepsilon$
ε	ε	Akceptácia

Keďže výpočet syntaktického analyzátora dospel do situácie, v ktorej bol vstup celý prečítaný a zásobník celý vyprázdnený, reťazec cd má v uvedenej gramatike deriváciu. Ľavá derivácia tohto reťazca vznikne postupnou aplikáciou pravidiel použitých na expanzie.

Počas výpočtu sme na expanzie použili pravidlá, ktoré sú v rozkladovej tabuľke na pozíciách $RT[C, \varepsilon], RT[D, \varepsilon]$. Len pripomíname, že v tomto prípade označuje ε situáciu, v ktorej je **celý vstup prečítaný**, teda tieto pravidlá sú na expanziu použiteľné len v prípade, že bol vstup celý prečítaný!

Na základe syntaktickej analýzy sme zistili, že ľavú deriváciu reťazca cd dostaneme postupnou aplikáciou tých pravidiel, ktoré $LL(1)$-syntaktický analyzátor použil pri expanziách, vždy na prvý neterminál zľava:

$$S \Rightarrow_l \mathbf{A}dDC \Rightarrow_l \mathbf{B}CDdDC \Rightarrow_l \mathbf{D}CDdDC \Rightarrow_l \mathbf{C}DdDC \Rightarrow_l c\mathbf{D}DdDC \Rightarrow_l$$
$$\Rightarrow_l c\mathbf{D}dDC \Rightarrow_l cd\mathbf{D}C \Rightarrow_l cd\mathbf{C} \Rightarrow_l cd\mathbf{D} \Rightarrow_l cd$$

2. Syntaktická analýza reťazca dd:

Zvyšok vstupu	Zásobník $\sqsupset$	Akcia
dd	S	Expanzia, $S \rightarrow AdDC$
dd	$AdDC$	Expanzia, $A \rightarrow BCD$
dd	$BCDdDC$	Expanzia, $B \rightarrow D$
dd	$DCDdDC$	Expanzia, $D \rightarrow \varepsilon$
dd	$CDdDC$	Expanzia, $C \rightarrow D$
dd	$DDdDC$	Expanzia, $D \rightarrow \varepsilon$
dd	$DdDC$	Expanzia, $D \rightarrow \varepsilon$
dd	dDC	Porovnanie
d	DC	Expanzia, $D \rightarrow \varepsilon$
d	C	Expanzia, $C \rightarrow D$
d	D	Expanzia, $D \rightarrow \varepsilon$
d	ε	**Syntaktická chyba**

Keďže výpočet syntaktického analyzátora dospel do situácie, v ktorej bol obsah zásobníka vyprázdnený, avšak na vstupe zostali nespracované symboly, dochádza k syntaktickej chybe a reťazec dd teda nemá v uvedenej gramatike deriváciu.

Kapitola 8

Syntaktická analýza bottom-up

8.1 Konštrukcia $LR(0)$-syntaktického analyzátora

Úloha č. 8.1.1 Je daná bezkontextová gramatika $G = (\{S, A, B\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

1. $S \to aSb$
2. $S \to A$
3. $A \to ab$
4. $A \to B$
5. $B \to c$

Pre uvedenú gramatiku zostrojte $LR(0)$-automat a tabuľky $ACTION$ a $GOTO$. Určte, či ide o $LR(0)$-gramatiku.

Riešenie:
Zostrojenie $LR(0)$-automatu začneme tým, že do gramatiky pridáme nový počiatočný neterminál S' a do gramatiky pridáme pravidlo $S' \to S$, kde S je pôvodný počiatočný neterminál gramatiky.

Následne zostrojíme jednotlivé stavy $LR(0)$-automatu, spolu s príslušnými $LR(0)$-položkami.

1. Počiatočný stav $LR(0)$-automatu, s_0:
   - Počiatočný stav $LR(0)$-automatu, označíme ho s_0, vznikne ako **uzáver množiny $LR(0)$-položiek** pre položku v tvare $S' \to \bullet S$, t. j. počítame výsledok operácie `CLOSURE0`$(\{S' \to \bullet S\})$.

- Pri počítaní uzáveru množiny $LR(0)$-položiek postupujeme nasledovne:
  (a) Ak je v množine položiek položka tvaru $B \to \alpha \bullet A\beta$, kde A je neterminál, do množiny pridáme všetky (chýbajúce) položky tvaru $A \to \bullet\gamma$, kde $A \to \gamma \in P$, teda do množiny pridáme všetky pravidlá, ktorých ľavú stranu tvorí neterminál A, pričom na začiatok pravej strany dáme symbol $\bullet$.
  (b) Postup opakujeme, pokiaľ nám pribúdajú nové položky.
- V prípade položky $S' \to \bullet S$ pridáme do množiny všetky chýbajúce položky, ktoré vzniknú zo všetkých pravidiel, ktoré majú na ľavej strane neterminál S, pričom symbol $\bullet$ dáme na začiatok pravej strany, t. j. pridáme položky:
  - $S \to \bullet aSb$
  - $S \to \bullet A$
- Keďže sme pridali položku $S \to \bullet A$, teda takú, kde je za symbolom $\bullet$ neterminál, v tomto prípade A, pridáme do množiny všetky chýbajúce položky, ktoré vzniknú zo všetkých pravidiel, ktoré majú na ľavej strane neterminál A, pričom symbol $\bullet$ dáme na začiatok pravej strany, t. j. pridáme položky:
  - $A \to \bullet ab$
  - $A \to \bullet B$
- Keďže sme pridali položku $A \to \bullet B$, teda takú, kde je za symbolom $\bullet$ neterminál, v tomto prípade B, pridáme do množiny všetky chýbajúce položky, ktoré vzniknú zo všetkých pravidiel, ktoré majú na ľavej strane neterminál B, pričom symbol $\bullet$ dáme na začiatok pravej strany, t. j. pridáme položku:
  - $B \to \bullet c$
- Keďže nám už nepribudla nová položka, v ktorej by za symbolom $\bullet$ bol neterminál, výsledný uzáver $LR(0)$-položky $S' \to \bullet S$ je množina:
  - $S' \to \bullet S$
  - $S \to \bullet aSb$
  - $S \to \bullet A$
  - $A \to \bullet ab$
  - $A \to \bullet B$
  - $B \to \bullet c$
- Táto množina tvorí **počiatočný stav** s_0 v hľadanom $LR(0)$-automate.
- Z tohto stavu budú následne v $LR(0)$-automate viesť prechody na tie symboly gramatiky, ktoré v príslušných položkách stoja za symbolom $\bullet$. Prechod bude viesť do stavu, ktorý vznikne ako uzáver množiny položiek, v ktorých presunieme symbol $\bullet$ za ten symbol gramatiky, na ktorý vedieme prechod:
  - Prechod na symbol S pre položku $S' \to \bullet S$ do stavu s_1 daného uzáverom $s_1 = \texttt{CLOSURE0}(\{S' \to S\bullet\})$.

   - Prechod na symbol a pre položky $S \to \bullet aSb$, $A \to \bullet ab$ do stavu s_2 daného uzáverom $s_2 = \texttt{CLOSURE0}(\{S \to a \bullet Sb,\ A \to a \bullet b\})$.
   - Prechod na symbol A pre položku $S \to \bullet A$ do stavu s_3 daného uzáverom $s_3 = \texttt{CLOSURE0}(\{S \to A\bullet\})$.
   - Prechod na symbol B pre položku $A \to \bullet B$ do stavu s_4 daného uzáverom $s_4 = \texttt{CLOSURE0}(\{A \to B\bullet\})$.
   - Prechod na symbol c pre položku $B \to \bullet c$ do stavu s_5 daného uzáverom $s_5 = \texttt{CLOSURE0}(\{B \to c\bullet\})$.

2. Stav $s_1 = \texttt{CLOSURE0}(\{S' \to S\bullet\})$:

   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $S' \to S\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
   - V takom prípade platí, že stav s_1 je tvorený len položkou $S' \to S\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu nebudú viesť žiadne prechody.

3. Stav $s_2 = \texttt{CLOSURE0}(\{S \to a \bullet Sb,\ A \to a \bullet b\})$:

   - Na základe položky $S \to a \bullet Sb$ pridáme do množiny položiek, ktorá bude tvoriť stav s_2, položky $S \to \bullet aSb, S \to \bullet A$.
   - Na základe položky $A \to a \bullet b$ do množiny nepridáme žiadne nové položky, pretože za symbolom $\bullet$ stojí v položke terminálny symbol.
   - Keďže sme do množiny pridali položku $S \to \bullet A$, kde sa za symbolom $\bullet$ nachádza neterminál A, pridáme do množiny položky $A \to \bullet ab, A \to \bullet B$.
   - Keďže sme do množiny pridali položku $A \to \bullet B$, kde sa za symbolom $\bullet$ nachádza neterminál B, pridáme do množiny položku $B \to \bullet c$.
   - Výsledný uzáver $\texttt{CLOSURE0}(\{S \to a \bullet Sb,\ A \to a \bullet b\})$ je teda tvorený množinou položiek:
     - $S \to a \bullet Sb$
     - $A \to a \bullet b$
     - $S \to \bullet aSb$
     - $S \to \bullet A$
     - $A \to \bullet ab$
     - $A \to \bullet B$
     - $B \to \bullet c$
   - Z tohto stavu budú následne v $LR(0)$-automate viesť prechody na tie symboly gramatiky, ktoré v príslušných položkách stoja za symbolom $\bullet$. Prechod bude viesť do stavu, ktorý vznikne ako uzáver množiny položiek, v ktorých presunieme symbol $\bullet$ za ten symbol gramatiky, na ktorý vedieme prechod.

Samozrejme, ak v $LR(0)$-automate už **existuje** stav, ktorý by obsahoval tú istú množinu položiek, tak potom prechod vedieme do už existujúceho stavu.

- Prechod na symbol S pre položku $S \to a \bullet Sb$ do nového stavu, označme ho s_6, daného uzáverom $s_6 = \texttt{CLOSURE0}(\{S \to aS \bullet b\})$.
- Prechod na symbol b pre položku $A \to a \bullet b$ do nového stavu, označme ho s_7, daného uzáverom $s_7 = \texttt{CLOSURE0}(\{A \to ab\bullet\})$.
- Prechod na symbol a pre položky $S \to \bullet aSb$, $A \to \bullet ab$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{S \to a \bullet Sb,\ A \to a \bullet b\})$. **Takýto stav už však máme**, vznikol pri vytváraní prechodov zo stavu s_0, je označený ako s_2. Preto z aktuálne vyšetrovaného stavu, s_2, bude pre symbol a viesť **slučka** do stavu s_2.
- Prechod na symbol A pre položku $S \to \bullet A$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{S \to A\bullet\})$. **Takýto stav nám už vznikol** pri vytváraní prechodov zo stavu s_0, označili sme ho s_3. Preto bude v tomto prípade prechod zo stavu s_2 na symbol A do stavu s_3.
- Prechod na symbol B pre položku $A \to \bullet B$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{A \to B\bullet\})$. **Takýto stav nám už vznikol** pri vytváraní prechodov zo stavu s_0, označili sme ho s_4. Preto bude v tomto prípade prechod zo stavu s_2 na symbol B do stavu s_4.
- Prechod na symbol c pre položku $B \to \bullet c$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{B \to c\bullet\})$. **Takýto stav nám už vznikol** pri vytváraní prechodov zo stavu s_0, označili sme ho s_5. Preto bude v tomto prípade prechod zo stavu s_2 na symbol c do stavu s_5.

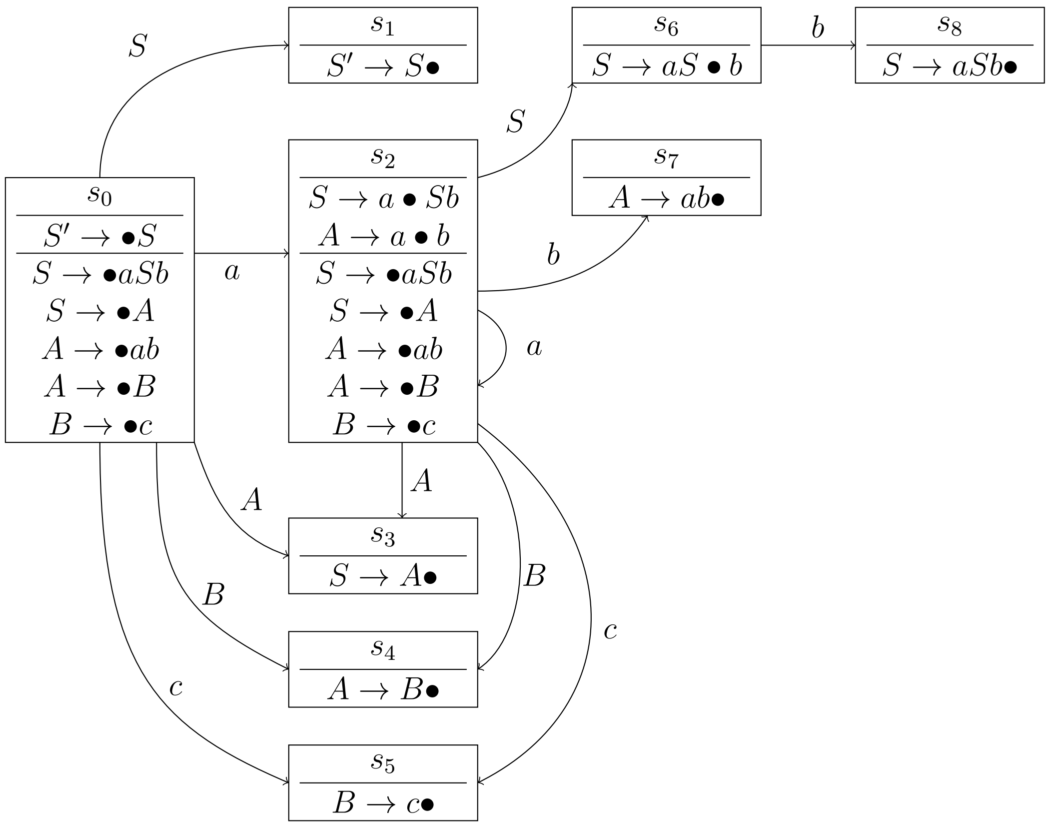
4. Stav $s_3 = \texttt{CLOSURE0}(\{S \to A\bullet\})$:

   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $S \to A\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
   - V takom prípade platí, že stav s_3 je tvorený len položkou $S \to A\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu s_3 nebudú viesť žiadne prechody.

5. Stav $s_4 = \texttt{CLOSURE0}(\{A \to B\bullet\})$:

   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $A \to B\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
   - V takom prípade platí, že stav s_4 je tvorený len položkou $A \to B\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu s_4 nebudú viesť žiadne prechody.

6. Stav $s_5 = \texttt{CLOSURE0}(\{B \to c\bullet\})$:
   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $B \to c\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
   - V takom prípade platí, že stav s_5 je tvorený len položkou $B \to c\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu s_5 nebudú viesť žiadne prechody.
7. Stav $s_6 = \texttt{CLOSURE0}(\{S \to aS \bullet b\})$:
   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $S \to aS \bullet b$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
   - V takom prípade platí, že stav s_6 je tvorený len položkou $S \to aS \bullet b$.
   - Z tohto stavu budú následne v $LR(0)$-automate viesť prechody na tie symboly gramatiky, ktoré v príslušných položkách stoja za symbolom $\bullet$. Prechod bude viesť do stavu, ktorý vznikne ako uzáver množiny položiek, v ktorých presunieme symbol $\bullet$ za ten symbol gramatiky, na ktorý vedieme prechod. Samozrejme, ak v $LR(0)$-automate už **existuje** stav, ktorý by obsahoval tú istú množinu položiek, tak potom prechod vedieme do už existujúceho stavu.
     - Prechod na symbol b pre položku $S \to aS \bullet b$ do nového stavu, označme ho s_8, daného uzáverom $s_8 = \texttt{CLOSURE0}(\{S \to aSb\bullet\})$.
8. Stav $s_7 = \texttt{CLOSURE0}(\{A \to ab\bullet\})$:
   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $A \to ab\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
   - V takom prípade platí, že stav s_7 je tvorený len položkou $A \to ab\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu s_7 nebudú viesť žiadne prechody.
9. Stav $s_8 = \texttt{CLOSURE0}(\{S \to aSb\bullet\})$:
   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $S \to aSb\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
   - V takom prípade platí, že stav s_8 je tvorený len položkou $S \to aSb\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu s_8 nebudú viesť žiadne prechody.



Obr. 8.1: $LR(0)$-automat ku gramatike z úlohy 8.1.1

10. Keďže sme vyšetrili všetky stavy, ktoré počas konštrukcie $LR(0)$-automatu postupne vznikli, dostávame výsledný $LR(0)$-automat, ktorý je znázornený na obrázku 8.1.

Tabuľky $ACTION$ a $GOTO$ zostrojíme na základe $LR(0)$-automatu nasledovným spôsobom:

- Tabuľka $ACTION$ — ide o tabuľku akcií, ktoré vykonáva $LR(0)$-syntaktický analyzátor v závislosti na tom, aký stavový symbol sa nachádza na vrchu jeho zásobníka. Túto tabuľku zostrojíme v prípade $LR(0)$-syntaktického analyzátora pre jednotlivé stavy $LR(0)$-automatu nasledovným spôsobom:
  - Ak stav s obsahuje aspoň 1 položku, v ktorej sa za symbolom $\bullet$ nachádza **terminálny symbol**, bude sa v tabuľke $ACTION$ na pozícii $ACTION[s]$ nachádzať akcia **Presun**, teda je signalizovaný presun terminálneho symbolu do zásobníka.
  - Ak stav s obsahuje položku v tvare $A \to \alpha\bullet$, t. j. symbol $\bullet$ sa nachádza na konci nejakého pravidla gramatiky, vo všeobecnosti $A \to \alpha$, bude sa v tabuľke $ACTION$ na pozícii $ACTION[s]$ nachádzať akcia **Redukcia** $A \to \alpha$, teda je signalizovaná redukcia podľa pravidla $A \to \alpha$.

   - Ak stav s obsahuje položku $S' \to S\bullet$, bude sa v tabuľke $ACTION$ na pozícii $ACTION[s]$ nachádzať akcia **Akceptácia**.

- Tabuľka $GOTO$ — ide o prechodovú tabuľku $LR(0)$-automatu, t. j. zachytáva prechody z jednotlivých stavov na jednotlivé symboly gramatiky.

Tabuľka $ACTION$:

- V stave s_0 je signalizovaný presun, pretože obsahuje položky $S \to \bullet aSb$, $A \to \bullet ab, B \to \bullet c$, v ktorých je za symbolom $\bullet$ terminál.
- V stave s_1 je signalizovaná akceptácia, pretože obsahuje položku $S' \to S\bullet$.
- V stave s_2 je signalizovaný presun, pretože obsahuje položky $A \to a \bullet b$, $S \to \bullet aSb, A \to \bullet ab, B \to \bullet c$, v ktorých je za symbolom $\bullet$ terminál.
- V stave s_3 je signalizovaná redukcia podľa pravidla $S \to A$, pretože obsahuje položku $S \to A\bullet$.
- V stave s_4 je signalizovaná redukcia podľa pravidla $A \to B$, pretože obsahuje položku $A \to B\bullet$.
- V stave s_5 je signalizovaná redukcia podľa pravidla $B \to c$, pretože obsahuje položku $B \to c\bullet$.
- V stave s_6 je signalizovaný presun, pretože obsahuje položku $S \to aS \bullet b$, v ktorej je za symbolom $\bullet$ terminál.
- V stave s_7 je signalizovaná redukcia podľa pravidla $A \to ab$, pretože obsahuje položku $A \to ab\bullet$.
- V stave s_8 je signalizovaná redukcia podľa pravidla $S \to aSb$, pretože obsahuje položku $S \to aSb\bullet$.

$ACTION$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
	P	A	P	$R2$	$R4$	$R5$	P	$R3$	$R1$

Tabuľka 8.1: Tabuľka $ACTION$ $LR(0)$-analyzátora pre gramatiku z úlohy 8.1.1

Tabuľka $GOTO$:

- V $LR(0)$-automate vedú nasledovné prechody zo stavu s_0:
   - na symbol S do stavu s_1, teda v $GOTO$ tabuľke $GOTO[s_0, S] = s_1$.
   - na symbol a do stavu s_2, teda v $GOTO$ tabuľke $GOTO[s_0, a] = s_2$.
   - na symbol A do stavu s_3, teda v $GOTO$ tabuľke $GOTO[s_0, A] = s_3$.
   - na symbol B do stavu s_4, teda v $GOTO$ tabuľke $GOTO[s_0, B] = s_4$.

- na symbol c do stavu s_5, teda v $GOTO$ tabuľke $GOTO[s_0, c] = s_5$.

- V $LR(0)$-automate nevedú žiadne prechody zo stavu s_1, preto sú v $GOTO$ tabuľke všetky bunky pre stav s_1 prázdne.

- V $LR(0)$-automate vedú nasledovné prechody zo stavu s_2:
  - na symbol a do stavu s_2, teda v $GOTO$ tabuľke $GOTO[s_2, a] = s_2$.
  - na symbol A do stavu s_3, teda v $GOTO$ tabuľke $GOTO[s_2, A] = s_3$.
  - na symbol B do stavu s_4, teda v $GOTO$ tabuľke $GOTO[s_2, B] = s_4$.
  - na symbol c do stavu s_5, teda v $GOTO$ tabuľke $GOTO[s_2, c] = s_5$.
  - na symbol S do stavu s_6, teda v $GOTO$ tabuľke $GOTO[s_2, S] = s_6$.
  - na symbol b do stavu s_7, teda v $GOTO$ tabuľke $GOTO[s_2, b] = s_7$.

- V $LR(0)$-automate nevedú žiadne prechody zo stavu s_3, preto sú v $GOTO$ tabuľke všetky bunky pre stav s_3 prázdne.

- V $LR(0)$-automate nevedú žiadne prechody zo stavu s_4, preto sú v $GOTO$ tabuľke všetky bunky pre stav s_4 prázdne.

- V $LR(0)$-automate nevedú žiadne prechody zo stavu s_5, preto sú v $GOTO$ tabuľke všetky bunky pre stav s_5 prázdne.

- V $LR(0)$-automate vedú nasledovné prechody zo stavu s_6:
  - na symbol b do stavu s_8, teda v $GOTO$ tabuľke $GOTO[s_6, b] = s_8$.

- V $LR(0)$-automate nevedú žiadne prechody zo stavu s_7, preto sú v $GOTO$ tabuľke všetky bunky pre stav s_7 prázdne.

- V $LR(0)$-automate nevedú žiadne prechody zo stavu s_8, preto sú v $GOTO$ tabuľke všetky bunky pre stav s_8 prázdne.

$GOTO$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
a	s_2		s_2						
b			s_7				s_8		
c	s_5		s_5						
S'									
S	s_1		s_6						
A	s_3		s_3						
B	s_4		s_4						

Tabuľka 8.2: Tabuľka $GOTO$ $LR(0)$-analyzátora pre gramatiku z úlohy 8.1.1

Pri určení, či je zadaná gramatika $LR(0)$-gramatikou vychádzame z toho, či príslušná tabuľka $ACTION$ obsahuje konflikty alebo nie. V princípe môže tabuľka $ACTION$ obsahovať 2 typy konfliktov:

- Presun/redukcia — ak sa v tej istej bunke tabuľky $ACTION$ súčasne nachádzajú aj akcia presun, aj akcia redukcia podľa nejakého pravidla.
- Redukcia/redukcia — ak sa v tej istej bunke tabuľky $ACTION$ súčasne nachádzajú aspoň 2 rôzne redukcie, t. j. redukcie podľa rôznych pravidiel.

Keďže v tabuľke $ACTION$, ktorú sme zostrojili, sa **nenachádza konflikt**, t. j. v každej bunke je signalizovaná najviac jedna akcia, zadaná gramatika **je $LR(0)$-gramatikou**.

Úloha č. 8.1.2 Je daná bezkontextová gramatika $G = (\{S, A\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

1. $S \rightarrow Aa$
2. $S \rightarrow b$
3. $A \rightarrow cAbA$
4. $A \rightarrow a$

Pre uvedenú gramatiku zostrojte $LR(0)$-automat a tabuľky $ACTION$ a $GOTO$. Určte, či ide o $LR(0)$-gramatiku.

Riešenie:
Zostrojenie $LR(0)$-automatu začneme tým, že do gramatiky pridáme nový počiatočný neterminál S' a do gramatiky pridáme pravidlo $S' \rightarrow S$, kde S je pôvodný počiatočný neterminál gramatiky.

Následne zostrojíme jednotlivé stavy $LR(0)$-automatu, spolu s príslušnými $LR(0)$-položkami.

1. Počiatočný stav $LR(0)$-automatu, s_0:
   - Počiatočný stav $LR(0)$-automatu s_0 vznikne ako **uzáver množiny $LR(0)$-položiek** pre položku v tvare $S' \rightarrow \bullet S$, t. j. počítame výsledok operácie $\texttt{CLOSURE0}(\{S' \rightarrow \bullet S\})$.
   - Na základe položky $S' \rightarrow \bullet S$ pridáme do množiny položky $S \rightarrow \bullet Aa$, $S \rightarrow \bullet b$.
   - Keďže sme do množiny pridali položku $S \rightarrow \bullet Aa$, kde sa za symbolom $\bullet$ nachádza neterminál A, pridáme do množiny položky $A \rightarrow \bullet cAbA$, $A \rightarrow \bullet a$.

- Keďže už nepribudli položky, v ktorých by za symbolom $\bullet$ bol neterminál, výsledný uzáver $LR(0)$-položky $S' \to \bullet S$ je množina:
  - $S' \to \bullet S$
  - $S \to \bullet Aa$
  - $S \to \bullet b$
  - $A \to \bullet cAbA$
  - $A \to \bullet a$
- Táto množina tvorí **počiatočný stav** s_0 v hľadanom $LR(0)$-automate.
- Z tohto stavu budú následne v $LR(0)$-automate viesť prechody na tie symboly gramatiky, ktoré v príslušných položkách stoja za symbolom $\bullet$. Prechod bude viesť do stavu, ktorý vznikne ako uzáver množiny položiek, v ktorých presunieme symbol $\bullet$ za ten symbol gramatiky, na ktorý vedieme prechod:
  - Prechod na symbol A pre položku $S \to \bullet Aa$ do stavu s_1 daného uzáverom $s_1 = \texttt{CLOSURE0}(\{S \to A \bullet a\})$.
  - Prechod na symbol S pre položku $S' \to \bullet S$ do stavu s_2 daného uzáverom $s_2 = \texttt{CLOSURE0}(\{S' \to S\bullet\})$.
  - Prechod na symbol c pre položku $A \to \bullet cAbA$ do stavu s_3 daného uzáverom $s_3 = \texttt{CLOSURE0}(\{A \to c \bullet AbA\})$.
  - Prechod na symbol b pre položku $S \to \bullet b$ do stavu s_4 daného uzáverom $s_4 = \texttt{CLOSURE0}(\{S \to b\bullet\})$.
  - Prechod na symbol a pre položku $A \to \bullet a$ do stavu s_5 daného uzáverom $s_5 = \texttt{CLOSURE0}(\{A \to a\bullet\})$.

2. Stav $s_1 = \texttt{CLOSURE0}(\{S \to A \bullet a\})$:

- V prípade počítania uzáveru množiny, ktorá obsahuje len položku $S \to A \bullet a$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
- V takom prípade platí, že stav s_1 je tvorený len položkou $S \to A \bullet a$.
- Z tohto stavu bude následne v $LR(0)$-automate viesť prechod:
  - Prechod na symbol a pre položku $S \to A \bullet a$ do stavu s_6 daného uzáverom $s_6 = \texttt{CLOSURE0}(\{S \to Aa\bullet\})$.

3. Stav $s_2 = \texttt{CLOSURE0}(\{S' \to S\bullet\})$:

- V prípade počítania uzáveru množiny, ktorá obsahuje len položku $S' \to S\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
- V takom prípade platí, že stav s_2 je tvorený len položkou $S' \to S\bullet$.
- Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu nebudú viesť žiadne prechody.

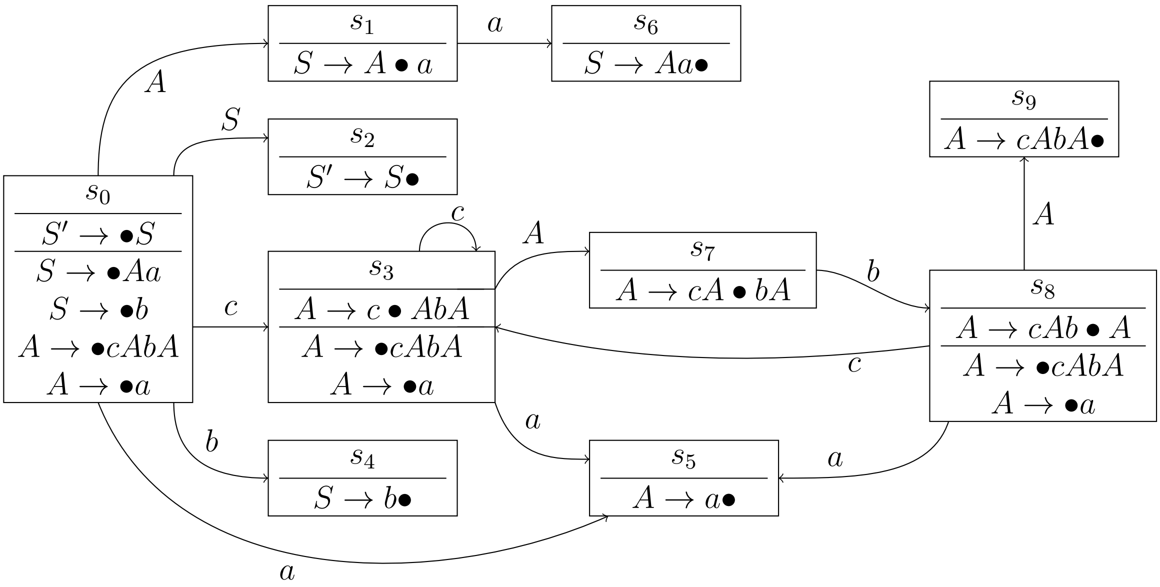
4. Stav $s_3 = \texttt{CLOSURE0}(\{A \to c \bullet AbA\})$:
   - Na základe položky $A \to c \bullet AbA$ pridáme do množiny položiek, ktorá bude tvoriť stav s_3, položky $A \to \bullet cAbA, A \to \bullet a$.
   - Keďže v pridaných položkách za symbolom $\bullet$ nestojí neterminál, ďalšie položky už nepridáme a stav s_3 bude tvorený položkami:
     - $A \to c \bullet AbA$
     - $A \to \bullet cAbA$
     - $A \to \bullet a$
   - Z tohto stavu budú následne v $LR(0)$-automate viesť nasledovné prechody:
     - Prechod na symbol A pre položku $A \to c \bullet AbA$ do stavu s_7 daného uzáverom $s_7 = \texttt{CLOSURE0}(\{A \to cA \bullet bA\})$.
     - Prechod na symbol c pre položku $A \to \bullet cAbA$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{A \to c \bullet AbA\})$. Takýto stav sme už vytvorili, je ním stav s_3, teda v stave s_3 bude slučka na symbol c.
     - Prechod na symbol a pre položku $A \to \bullet a$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{A \to a\bullet\})$. Takýto stav sme už vytvorili, je ním stav s_5, teda zo stavu s_3 bude viesť prechod na symbol a do stavu s_5.
5. Stav $s_4 = \texttt{CLOSURE0}(\{S \to b\bullet\})$:
   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $S \to b\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky, resp. v tomto prípade stojí $\bullet$ na konci položky.
   - Stav s_4 bude teda tvorený len položkou $S \to b\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu nebudú viesť žiadne prechody.
6. Stav $s_5 = \texttt{CLOSURE0}(\{A \to a\bullet\})$:
   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $A \to a\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky, resp. v tomto prípade stojí $\bullet$ na konci položky.
   - Stav s_5 bude teda tvorený len položkou $A \to a\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu nebudú viesť žiadne prechody.
7. Stav $s_6 = \texttt{CLOSURE0}(\{S \to Aa\bullet\})$:
   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $S \to Aa\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky, resp. v tomto prípade stojí $\bullet$ na konci položky.
   - Stav s_6 bude teda tvorený len položkou $A \to a\bullet$.

- Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu nebudú viesť žiadne prechody.

8. Stav $s_7 = \texttt{CLOSURE0}(\{A \to cA \bullet bA\})$:
   - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $A \to cA \bullet bA$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky.
   - V takom prípade platí, že stav s_7 je tvorený len položkou $A \to cA \bullet bA$.
   - Z tohto stavu bude následne v $LR(0)$-automate viesť prechod:
     - Prechod na symbol b pre položku $A \to cA \bullet bA$ do stavu s_8 daného uzáverom $s_8 = \texttt{CLOSURE0}(\{A \to cAb \bullet A\})$.

9. Stav $s_8 = \texttt{CLOSURE0}(\{A \to cAb \bullet A\})$:
   - Na základe položky $A \to cAb \bullet A$ do stavu s_8 pridáme položky $A \to \bullet cAbA$, $A \to \bullet a$.
   - Keďže v pridaných položkách za symbolom $\bullet$ nestojí neterminál, ďalšie položky už nepridáme a stav s_8 bude tvorený položkami:
     - $A \to cAb \bullet A$
     - $A \to \bullet cAbA$
     - $A \to \bullet a$
   - Z tohto stavu budú následne v $LR(0)$-automate viesť nasledovné prechody:
     - Prechod na symbol A pre položku $A \to cAb \bullet A$ do stavu s_9 daného uzáverom $s_9 = \texttt{CLOSURE0}(\{A \to cAbA\bullet\})$.
     - Prechod na symbol c pre položku $A \to \bullet cAbA$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{A \to c \bullet AbA\})$. Takýto stav sme už vytvorili, je ním stav s_3, teda zo stavu s_8 bude prechod na symbol c do stavu s_3.
     - Prechod na symbol a pre položku $A \to \bullet a$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{A \to a\bullet\})$. Takýto stav sme už vytvorili, je ním stav s_5, teda zo stavu s_8 bude viesť prechod na symbol a do stavu s_5.

10. Stav $s_9 = \texttt{CLOSURE0}(\{A \to cAbA\bullet\})$:
    - V prípade počítania uzáveru množiny, ktorá obsahuje len položku $A \to cAbA\bullet$ nemusíme nič počítať, pretože sa za symbolom $\bullet$ nenachádza neterminál gramatiky, resp. v tomto prípade stojí $\bullet$ na konci položky.
    - Stav s_9 bude teda tvorený len položkou $A \to cAbA\bullet$.
    - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu nebudú viesť žiadne prechody.



Obr. 8.2: $LR(0)$-automat ku gramatike z úlohy 8.1.2

11. Keďže sme vyšetrili všetky stavy, ktoré počas konštrukcie $LR(0)$-automatu postupne vznikli, dostávame výsledný $LR(0)$-automat, ktorý je znázornený na obrázku 8.2.

Tabuľka $ACTION$:

- V stave s_0 je signalizovaný presun, pretože obsahuje položky $S \to \bullet b$, $A \to \bullet cAbA$, $A \to \bullet a$, v ktorých je za symbolom $\bullet$ terminál.
- V stave s_1 je signalizovaný presun, pretože obsahuje položku $S \to A \bullet a$, v ktorej je za symbolom $\bullet$ terminál.
- V stave s_2 je signalizovaná akceptácia, pretože obsahuje položku $S' \to S\bullet$.
- V stave s_3 je signalizovaný presun, pretože obsahuje položky $A \to \bullet cAbA$, $A \to \bullet a$, v ktorých je za symbolom $\bullet$ terminál.
- V stave s_4 je signalizovaná redukcia podľa pravidla $S \to b$, pretože obsahuje položku $S \to b\bullet$.
- V stave s_5 je signalizovaná redukcia podľa pravidla $A \to a$, pretože obsahuje položku $A \to a\bullet$.
- V stave s_6 je signalizovaná redukcia podľa pravidla $S \to Aa$, pretože obsahuje položku $S \to Aa\bullet$.
- V stave s_7 je signalizovaný presun, pretože obsahuje položku $A \to cA \bullet bA$, v ktorej je za symbolom $\bullet$ terminál.

- V stave s_8 je signalizovaný presun, pretože obsahuje položky $A \to \bullet cAbA$, $A \to \bullet a$, v ktorých je za symbolom $\bullet$ terminál.
- V stave s_9 je signalizovaná redukcia podľa pravidla $A \to cAbA$, pretože obsahuje položku $A \to cAbA\bullet$.

$ACTION$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
	P	P	A	P	$R2$	$R4$	$R1$	P	P	$R3$

Tabuľka 8.3: Tabuľka $ACTION$ $LR(0)$-analyzátora pre gramatiku z úlohy 8.1.2

$GOTO$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
a	s_5	s_6		s_5					s_5	
b	s_4									
c	s_3			s_3					s_3	
S'										
S	s_2									
A	s_1			s_7					s_9	

Tabuľka 8.4: Tabuľka $GOTO$ $LR(0)$-analyzátora pre gramatiku z úlohy 8.1.2

Keďže v tabuľke $ACTION$, ktorú sme zostrojili, sa **nenachádza konflikt**, t. j. v každej bunke je signalizovaná najviac jedna akcia, zadaná gramatika **je $LR(0)$-gramatikou**.

Úloha č. 8.1.3 Je daná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c, d\}, P, S)$, ktorej pravidlá P sú:

1. $S \to aAb$
2. $S \to aA$
3. $A \to Bb$
4. $A \to Cc$
5. $A \to d$
6. $B \to aA$
7. $C \to aA$

Pre uvedenú gramatiku zostrojte $LR(0)$-automat a tabuľky $ACTION$ a $GOTO$. Určte, či ide o $LR(0)$-gramatiku.

Riešenie:
Zostrojenie $LR(0)$-automatu začneme tým, že do gramatiky pridáme nový počiatočný neterminál S' a do gramatiky pridáme pravidlo $S' \to S$, kde S je pôvodný počiatočný neterminál gramatiky. Následne zostrojíme jednotlivé stavy $LR(0)$-automatu, spolu s príslušnými $LR(0)$-položkami.

1. Počiatočný stav $LR(0)$-automatu, s_0:
   - Pre počiatočný stav $LR(0)$-automatu bude platiť $s_0 = \texttt{CLOSURE0}(\{S' \to \bullet S\})$.
   - Na základe položky $S' \to \bullet S$ pridáme do množiny položky $S \to \bullet aAb, S \to \bullet aA$.
   - Keďže v pridaných položkách sú za symbolom $\bullet$ vždy len terminály a nepribudla nám položka, v ktorej by za symbolom $\bullet$ bol neterminál, výsledný uzáver $LR(0)$-položky $S' \to \bullet S$ je množina:
     - $S' \to \bullet S$
     - $S \to \bullet aAb$
     - $S \to \bullet aA$
   - Z tohto stavu budú následne v $LR(0)$-automate viesť nasledovné prechody:
     - Prechod na symbol S pre položku $S' \to \bullet S$ do stavu s_1 daného uzáverom $s_1 = \texttt{CLOSURE0}(\{S' \to S\bullet\})$.
     - Prechod na symbol a pre položky $S \to \bullet aAb, S \to \bullet aA$ do stavu s_2 daného uzáverom $s_2 = \texttt{CLOSURE0}(\{S \to a \bullet Ab, S \to a \bullet A\})$.
2. Stav $s_1 = \texttt{CLOSURE0}(\{S' \to S\bullet\})$:
   - Stav s_1 bude tvorený len položkou $S' \to S\bullet$.
   - Keďže v tomto stave neexistuje položka, ktorá by za symbolom $\bullet$ mala nejaký symbol gramatiky, z tohto stavu nebudú viesť žiadne prechody.
3. Stav $s_2 = \texttt{CLOSURE0}(\{S \to a \bullet Ab, S \to a \bullet A\})$:
   - Na základe položky $S \to a \bullet Ab$ pridáme do množiny položiek, ktorá bude tvoriť stav s_3, položky $A \to \bullet Bb, A \to \bullet Cc, A \to \bullet d$.
   - Na základe položky $S \to a \bullet A$ by sme pridali položky $A \to \bullet Bb$, $A \to \bullet Cc, A \to \bullet d$, avšak tie sme tam už pridali v predchádzajúcom kroku.
   - Na základe pridanej položky $A \to \bullet Bb$ ďalej pridáme položku $B \to \bullet aA$.
   - Na základe pridanej položky $A \to \bullet Cc$ ďalej pridáme položku $C \to \bullet aA$.
   - Výsledný stav s_2 bude tvorený položkami:
     - $S \to a \bullet Ab$
     - $S \to a \bullet A$
     - $A \to \bullet Bb$
     - $A \to \bullet Cc$
     - $A \to \bullet d$
     - $B \to \bullet aA$
     - $C \to \bullet aA$

- Z tohto stavu budú v $LR(0)$-automate viesť nasledovné prechody:
  - Prechod na symbol A pre položky $S \to a \bullet Ab, S \to a \bullet A$ do stavu s_3 daného uzáverom $s_3 = \texttt{CLOSURE0}(\{S \to aA \bullet b, S \to aA\bullet\})$.
  - Prechod na symbol B pre položku $A \to \bullet Bb$ do stavu s_4 daného uzáverom $s_4 = \texttt{CLOSURE0}(\{A \to B \bullet b\})$.
  - Prechod na symbol C pre položku $A \to \bullet Cc$ do stavu s_5 daného uzáverom $s_5 = \texttt{CLOSURE0}(\{A \to C \bullet c\})$.
  - Prechod na symbol d pre položku $A \to \bullet d$ do stavu s_6 daného uzáverom $s_6 = \texttt{CLOSURE0}(\{A \to d\bullet\})$.
  - Prechod na symbol a pre položky $B \to \bullet aA, C \to \bullet aA$ do stavu s_7 daného uzáverom $s_7 = \texttt{CLOSURE0}(\{B \to a \bullet A, C \to a \bullet A\})$.

4. Stav $s_3 = \texttt{CLOSURE0}(\{S \to aA \bullet b, S \to aA\bullet\})$:
   - V tomto prípade uzáverová operácia nepridá nové položky.
   - Stav s_3 bude teda tvorený len položkami $S \to aA \bullet b, S \to aA\bullet$.
   - Z tohto stavu bude následne v $LR(0)$-automate viesť prechod:
     - Prechod na symbol b pre položku $S \to aA \bullet b$ do stavu s_8 daného uzáverom $s_8 = \texttt{CLOSURE0}(\{S \to aAb\bullet\})$.

5. Stav $s_4 = \texttt{CLOSURE0}(\{A \to B \bullet b\})$:
   - V tomto prípade uzáverová operácia nepridá nové položky.
   - Stav s_4 bude teda tvorený len položkou $A \to B \bullet b$.
   - Z tohto stavu bude následne v $LR(0)$-automate viesť prechod:
     - Prechod na symbol b pre položku $A \to B \bullet b$ do stavu s_9 daného uzáverom $s_9 = \texttt{CLOSURE0}(\{A \to Bb\bullet\})$.

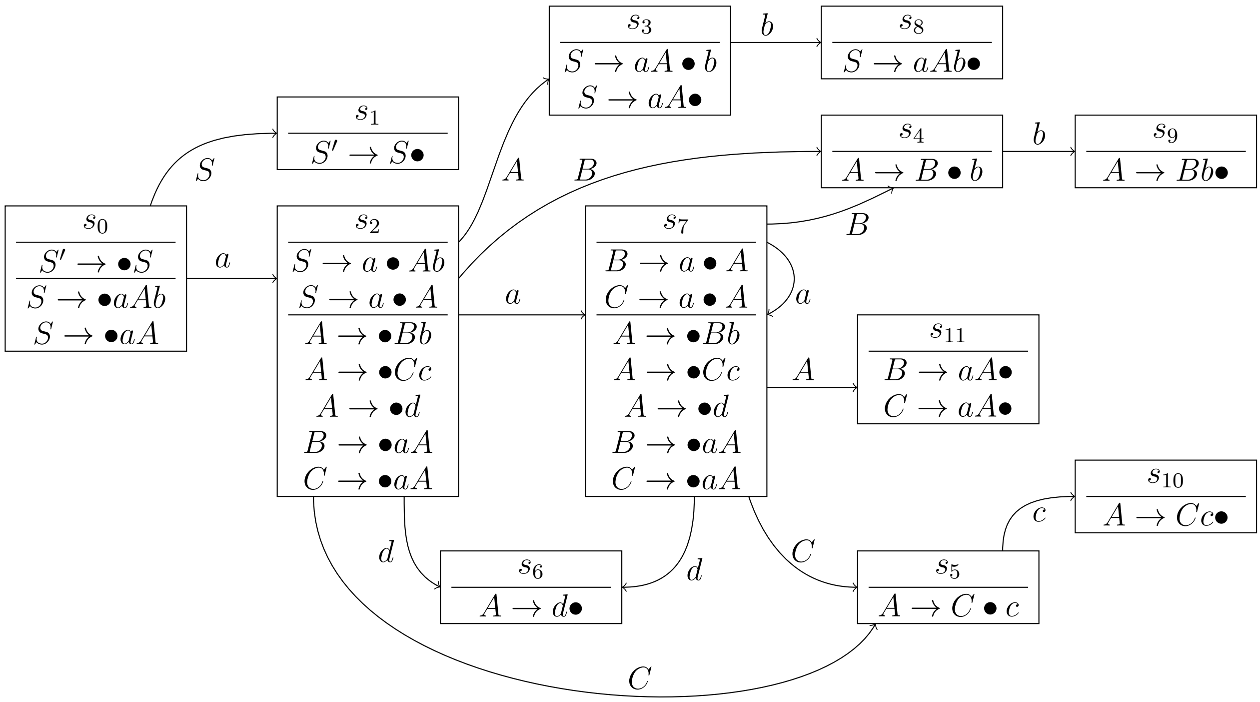
6. Stav $s_5 = \texttt{CLOSURE0}(\{A \to C \bullet c\})$:
   - V tomto prípade uzáverová operácia nepridá nové položky.
   - Stav s_5 bude teda tvorený len položkou $A \to C \bullet c$.
   - Z tohto stavu bude následne v $LR(0)$-automate viesť prechod:
     - Prechod na symbol c pre položku $A \to C \bullet c$ do stavu s_{10} daného uzáverom $s_{10} = \texttt{CLOSURE0}(\{A \to Cc\bullet\})$.

7. Stav $s_6 = \texttt{CLOSURE0}(\{A \to d\bullet\})$:
   - V tomto prípade uzáverová operácia nepridá nové položky.
   - Stav s_6 bude teda tvorený len položkou $A \to d\bullet$.
   - Z tohto stavu nebudú v $LR(0)$-automate viesť prechody.

8. Stav $s_7 = \texttt{CLOSURE0}(\{B \to a \bullet A, C \to a \bullet A\})$:
   - Na základe položiek $B \to a \bullet A$, resp. $C \to a \bullet A$ pribudnú v stave s_7 položky $A \to \bullet Bb, A \to \bullet Cc, A \to \bullet d$.
   - Na základe pridanej položky $A \to \bullet Bb$ pribudne do stavu s_7 položka $B \to \bullet aA$.
   - Na základe pridanej položky $A \to \bullet Cc$ pribudne do stavu s_7 položka $C \to \bullet aA$.
   - Stav s_7 bude tvorený položkami:
     - $B \to a \bullet A$
     - $C \to a \bullet A$
     - $A \to \bullet Bb$
     - $A \to \bullet Cc$
     - $A \to \bullet d$
     - $B \to \bullet aA$
     - $C \to \bullet aA$
   - Z tohto stavu budú následne v $LR(0)$-automate viesť prechody:
     - Prechod na symbol A pre položky $B \to a \bullet A, C \to a \bullet A$ do stavu s_{11} daného uzáverom $s_{11} = \texttt{CLOSURE0}(\{B \to aA\bullet, C \to aA\bullet\})$.
     - Prechod na symbol B pre položku $A \to \bullet Bb$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{A \to B \bullet b\})$, teda do stavu s_4.
     - Prechod na symbol C pre položku $A \to \bullet Cc$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{A \to C \bullet c\})$, teda do stavu s_5.
     - Prechod na symbol d pre položku $A \to \bullet d$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{A \to d\bullet\})$, teda do stavu s_6.
     - Prechod na symbol a pre položky $B \to \bullet aA, C \to \bullet aA$ do stavu daného uzáverom $\texttt{CLOSURE0}(\{B \to a \bullet A, C \to a \bullet A\})$, teda v stave s_7 bude slučka pre symbol a.

9. Stav $s_8 = \texttt{CLOSURE0}(\{S \to aAb\bullet\})$:
   - V tomto prípade uzáverová operácia nepridá nové položky.
   - Stav s_8 bude teda tvorený len položkou $S \to aAb\bullet$.
   - Z tohto stavu nebudú v $LR(0)$-automate viesť prechody.

10. Stav $s_9 = \texttt{CLOSURE0}(\{A \to Bb\bullet\})$:
    - V tomto prípade uzáverová operácia nepridá nové položky.
    - Stav s_9 bude teda tvorený len položkou $A \to Bb\bullet$.
    - Z tohto stavu nebudú v $LR(0)$-automate viesť prechody.



Obr. 8.3: $LR(0)$-automat ku gramatike z úlohy 8.1.3

11. Stav $s_{10} = \texttt{CLOSURE0}(\{A \to Cc\bullet\})$:
    - V tomto prípade uzáverová operácia nepridá nové položky.
    - Stav s_{10} bude teda tvorený len položkou $A \to Cc\bullet$.
    - Z tohto stavu nebudú v $LR(0)$-automate viesť prechody.
12. Stav $s_{11} = \texttt{CLOSURE0}(\{B \to aA\bullet, C \to aA\bullet\})$:
    - V tomto prípade uzáverová operácia nepridá nové položky.
    - Stav s_{11} bude teda tvorený len položkami $B \to aA\bullet, C \to aA\bullet$.
    - Z tohto stavu nebudú v $LR(0)$-automate viesť prechody.
13. Keďže sme vyšetrili všetky stavy, ktoré počas konštrukcie $LR(0)$-automatu postupne vznikli, dostávame výsledný $LR(0)$-automat, ktorý je znázornený na obrázku 8.3.

Tabuľka $ACTION$:

- V stave s_0 je signalizovaný presun položkami $S \to \bullet aAb, S \to \bullet aA$.
- V stave s_1 je signalizovaná akceptácia položkou $S' \to S\bullet$.
- V stave s_2 je signalizovaný presun položkami $A \to \bullet d, B \to \bullet aA, C \to \bullet aA$.

- V stave s_3 je signalizovaný presun, pretože obsahuje položku $S \to aA \bullet b$. **Súčasne je v tomto stave** signalizovaná redukcia podľa pravidla $S \to aA$ položkou $S \to aA\bullet$.
- V stave s_4 je signalizovaný presun položkou $A \to B \bullet b$.
- V stave s_5 je signalizovaný presun položkou $A \to C \bullet c$.
- V stave s_6 je signalizovaná redukcia podľa pravidla $A \to d$ položkou $A \to d\bullet$.
- V stave s_7 je signalizovaný presun položkami $A \to \bullet d, B \to \bullet aA, C \to \bullet aA$.
- V stave s_8 je signalizovaná redukcia podľa pravidla $S \to aAb$ položkou $S \to aAb\bullet$.
- V stave s_9 je signalizovaná redukcia podľa pravidla $A \to Bb$ položkou $A \to Bb\bullet$.
- V stave s_{10} je signalizovaná redukcia podľa pravidla $A \to Cc$ položkou $A \to Cc\bullet$.
- V stave s_{11} je signalizovaná redukcia podľa pravidla $B \to aA$ položkou $B \to aA\bullet$. **Súčasne je v tomto stave** signalizovaná redukcia podľa pravidla $C \to aA$ položkou $C \to aA\bullet$.

$ACTION$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}
	P	A	P	$P/R2$	P	P	$R5$	P	$R1$	$R3$	$R4$	$R6/R7$

Tabuľka 8.5: Tabuľka $ACTION$ $LR(0)$-analyzátora pre gramatiku z úlohy 8.1.3

Tabuľka $GOTO$:

$GOTO$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}
a	s_2		s_7					s_7				
b				s_8	s_9							
c						s_{10}						
d			s_6					s_6				
S'												
S	s_1											
A			s_3					s_{11}				
B			s_4					s_4				
C			s_5					s_5				

Tabuľka 8.6: Tabuľka $GOTO$ $LR(0)$-analyzátora pre gramatiku z úlohy 8.1.3

V tabuľke $ACTION$, ktorú sme zostrojili, sa **nachádzajú konflikty**, t. j. existujú v nej bunky, ktorých sú signalizované aspoň 2 rôzne akcie:

- $ACTION[s_3]$ obsahuje presun a redukciu podľa pravidla $S \to aA$, t. j. obsahuje konflikt typu presun/redukcia.

- $ACTION[s_{11}]$ obsahuje redukciu podľa pravidla $B \to aA$ a redukciu podľa pravidla $C \to aA$, t. j. obsahuje konflikt typu redukcia/redukcia.

Keďže $LR(0)$-analyzátor zostrojený k danej gramatike, resp. jeho tabuľka $ACTION$, obsahuje konflikty, príslušná gramatika **nie je** $LR(0)$-gramatikou.

8.2 Konštrukcia $SLR(1)$-syntaktického analyzátora

Úloha č. 8.2.1 Je daná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c, d\}, P, S)$, ktorej pravidlá P sú:

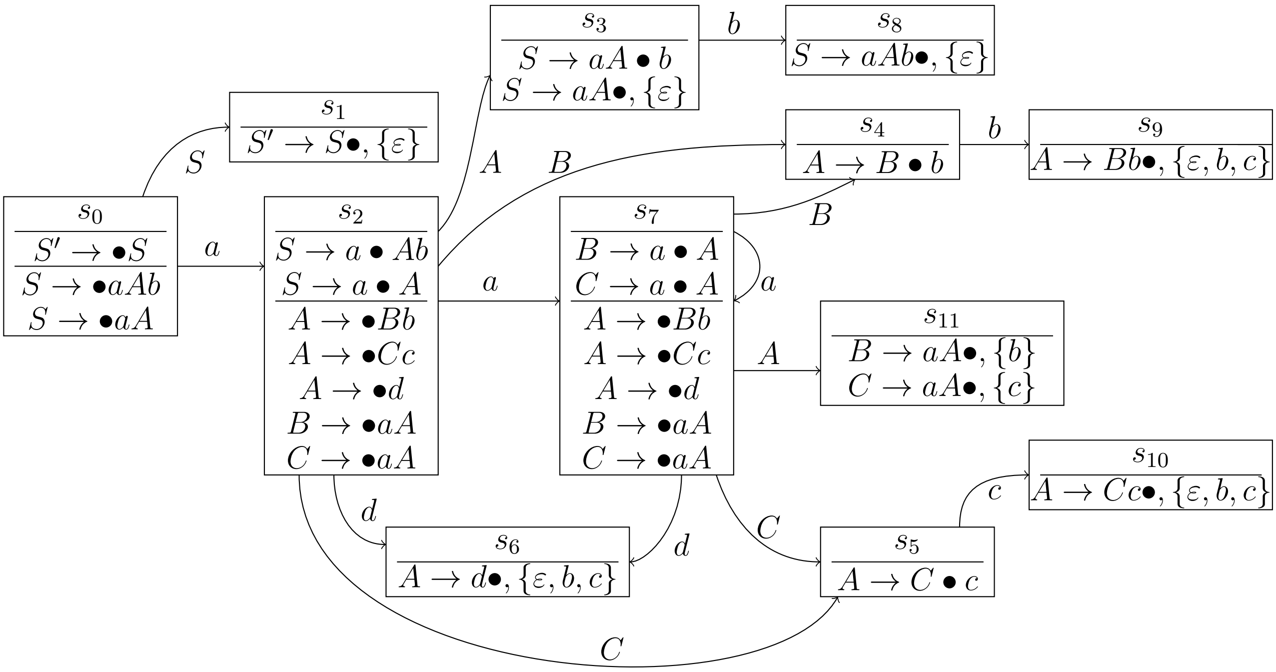
1. $S \to aAb$
2. $S \to aA$
3. $A \to Bb$
4. $A \to Cc$
5. $A \to d$
6. $B \to aA$
7. $C \to aA$

Pre uvedenú gramatiku zostrojte $SLR(1)$-syntaktický analyzátor a určte, či ide o $SLR(1)$-gramatiku.

Riešenie:
Zostrojenie $SLR(1)$-syntaktického analyzátora vychádza z konštrukcie $LR(0)$-automatu k zadanej gramatike. Zostrojí sa teda $LR(0)$-automat rovnakým spôsobom ako pri $LR(0)$-syntaktickom analyzátore, avšak navyše sa k tým položkám, ktoré signalizujú **redukciu** alebo **akceptáciu**, pridajú symboly z množiny $FOLLOW$ **neterminálu stojaceho v pravidle na ľavej strane**. Množiny $FIRST$ a $FOLLOW$ pre neterminály tejto gramatiky sú:

	S	A	B	C
$FIRST$	$\{a\}$	$\{a, d\}$	$\{a\}$	$\{a\}$
$FOLLOW$	$\{\varepsilon\}$	$\{\varepsilon, b, c\}$	$\{b\}$	$\{c\}$

$LR(0)$-automat, doplnený o symboly z množiny $FOLLOW$ pri redukčných a akceptačnej položke, je zobrazený na obrázku 8.4. Keďže pri tvorbe $LR(0)$-automatu do gramatiky dopĺňame neterminál S' ako nový počiatočný neterminál a pravidlo $S' \to S$, pre množinu $FOLLOW(S')$ vždy platí, že množina $FOLLOW(S') = \{\varepsilon\}$.



Obr. 8.4: $LR(0)$-automat ku gramatike z úlohy 8.2.1 doplnený o symboly z množiny $FOLLOW$

Tabuľky $ACTION$ a $GOTO$ zostrojíme pre $SLR(1)$-syntaktický analyzátor na základe uvedeného $LR(0)$-automatu nasledovným spôsobom:

- Tabuľka $ACTION$ — v tomto prípade bude o akcii rozhodovať nielen príslušný stav, ale aj vstupný symbol:
  - Ak stav s obsahuje aspoň 1 položku, v ktorej sa za symbolom $\bullet$ nachádza **terminálny symbol** t, bude sa v tabuľke $ACTION$ na pozícii $ACTION[s, t]$ nachádzať akcia **Presun**, teda je signalizovaný presun terminálneho symbolu do zásobníka v prípade, že sa na vstupe analyzátora nachádza terminál t.
  - Ak stav s obsahuje položku v tvare $A \to \alpha\bullet, \{t, ..., \}$, t. j. symbol $\bullet$ sa nachádza na konci nejakého pravidla gramatiky, vo všeobecnosti $A \to \alpha$ a $t \in FOLLOW(A)$, potom sa bude v tabuľke $ACTION$ na pozícii $ACTION[s, t]$ nachádzať akcia **Redukcia** $A \to \alpha$, teda je signalizovaná redukcia podľa pravidla $A \to \alpha$, ak je na vstupe terminál t.
  - Rovnako, ak do množiny $FOLLOW(A)$ patrí ε, tak ak stav s obsahuje položku v tvare $A \to \alpha\bullet, \{\varepsilon, ..., \}$, t. j. symbol $\bullet$ sa nachádza na konci nejakého pravidla gramatiky, vo všeobecnosti $A \to \alpha$ a $\in FOLLOW(A)$, potom sa v tabuľke $ACTION$ na pozícii $ACTION[s, \varepsilon]$ nachádzať akcia **Redukcia** $A \to \alpha$, teda je signalizovaná redukcia podľa pravidla $A \to \alpha$, ak je **vstup celý prečítaný**.

- Ak stav s obsahuje položku $S' \to S\bullet, \{\varepsilon\}$, bude sa v tabuľke $ACTION$ na pozícii $ACTION[s, \varepsilon]$ nachádzať akcia **Akceptácia**. Túto akciu je možné vykonať len ak je vstup celý prečítaný.

- Tabuľka $GOTO$ — tá sa zostrojí identicky ako pri $LR(0)$-analyzátore.

Tabuľka $ACTION$:

$ACTION$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}
a	P		P					P				
b				P	P		$R5$			$R3$	$R4$	$R6$
c						P	$R5$			$R3$	$R4$	$R7$
d			P					P				
ε		A		$R2$			$R5$		$R1$	$R3$	$R4$	

Tabuľka 8.7: Tabuľka $ACTION$ $SLR(1)$-analyzátora pre gramatiku z úlohy 8.2.1

Tabuľka $GOTO$:

$GOTO$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}
a	s_2		s_7					s_7				
b				s_8	s_9							
c						s_{10}						
d			s_6					s_6				
S'												
S	s_1											
A			s_3					s_{11}				
B			s_4					s_4				
C			s_5					s_5				

Tabuľka 8.8: Tabuľka $GOTO$ $SLR(1)$-analyzátora pre gramatiku z úlohy 8.2.1

Pri určení, či je zadaná gramatika $SLR(1)$-gramatikou vychádzame z toho, či príslušná tabuľka $ACTION$ obsahuje konflikty alebo nie. V princípe môže tabuľka $ACTION$ obsahovať 2 typy konfliktov:

- Presun/redukcia — ak sa v tej istej bunke tabuľky $ACTION[s, t]$ súčasne nachádzajú aj akcia presun, aj akcia redukcia podľa nejakého pravidla.

- Redukcia/redukcia — ak sa v tej istej bunke tabuľky $ACTION[s, t]$ súčasne nachádzajú aspoň 2 rôzne redukcie, t. j. redukcie podľa rôznych pravidiel.

Keďže v tabuľke $ACTION$, ktorú sme zostrojili, sa **nenachádza konflikt**, t. j. v každej bunke je signalizovaná najviac jedna akcia, zadaná gramatika **je** $SLR(1)$**-gramatikou**.

Úloha č. 8.2.2 Je daná bezkontextová gramatika $G = (\{S, A\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

1. $S \rightarrow Aa$
2. $S \rightarrow \varepsilon$
3. $A \rightarrow cAbA$
4. $A \rightarrow \varepsilon$

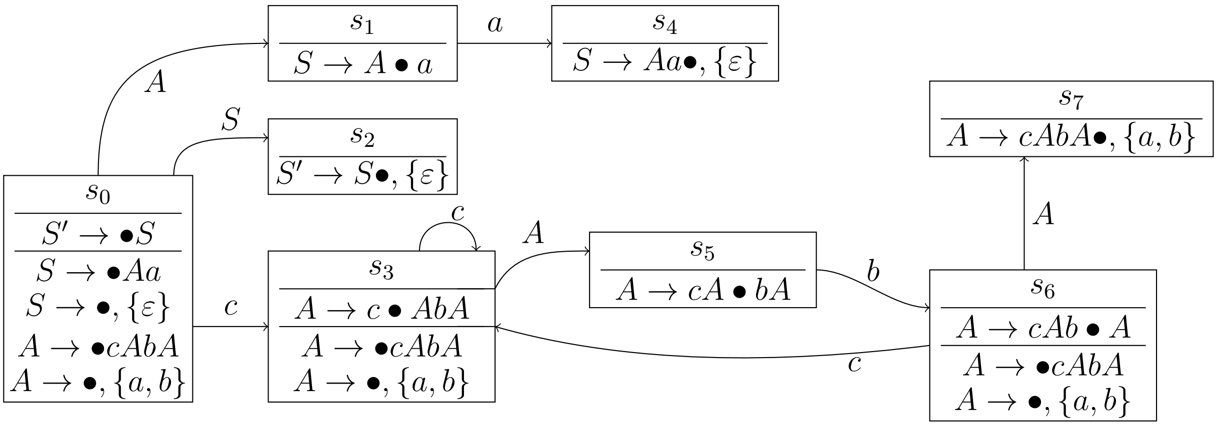
Pre uvedenú gramatiku zostrojte $SLR(1)$-syntaktický analyzátor a určte, či ide o $SLR(1)$-gramatiku.

Riešenie:
Množiny $FIRST$ a $FOLLOW$ pre neterminály tejto gramatiky sú:

	S	A
$FIRST$	$\{\varepsilon, a, c\}$	$\{c, \varepsilon\}$
$FOLLOW$	$\{\varepsilon\}$	$\{a, b\}$

Pri konštrukcii $LR(0)$-automatu sa v tomto príklade stretneme s položkou odvodenou z pravidla, ktoré má na pravej strane prázdny reťazec. Ukážeme si teraz na príklade počiatočného stavu, ako sa s tým vysporiadať:

- Ako sme písali už pri $LR(0)$-analyzátore, počiatočný stav $LR(0)$-automatu bude tvorený uzáverom položky $S' \rightarrow \bullet S$.
- V uvedenej gramatike to znamená, že do stavu s_0 pribudnú položky $S \rightarrow \bullet Aa$ a $S \rightarrow \bullet\varepsilon$.
- Keďže v položke $S \rightarrow \bullet\varepsilon$ je ε prázdny reťazec, často sa táto položka píše bez symbolu ε ako $S \rightarrow \bullet$, prípadne sa symbol $\bullet$ môže vložiť **za** symbol ε, t. j. $S \rightarrow \varepsilon\bullet$.
- Keďže pravidlo $S \rightarrow \varepsilon$ **nemá** na pravej strane žiadne symboly gramatiky, môžeme predpokladať, že v danom stave sa rozpoznala celá jeho pravá strana, čo zároveň korešponduje so situáciou, že položka $S \rightarrow \bullet$ bude priamo signalizovať redukciu podľa pravidla $S \rightarrow \varepsilon$.
- Pre úplnosť dodávame, že keďže v pridanej položke $S \rightarrow \bullet Aa$ stojí za symbolom $\bullet$ neterminál A, do stavu s_0 pridáme taktiež položky $A \rightarrow \bullet cAbA$, $A \rightarrow \bullet\varepsilon$.
- Aj položku $A \rightarrow \bullet\varepsilon$ môžeme prepísať ako $A \rightarrow \bullet$.
- Upozorňujeme, že v položkách $S \rightarrow \bullet\varepsilon, A \rightarrow \bullet\varepsilon$ je ε prázdny reťazec! To znamená, že v tomto prípade **nemôžeme** pre tento stav / tieto položky uvažovať prechody na symbol ε, keďže ε **nie je symbol gramatiky**, resp. by to znamenalo, že konštruujeme **nedeterministický syntaktický analyzátor**, čo je v priamom rozpore s našim cieľom.



Obr. 8.5: $LR(0)$-automat ku gramatike z úlohy 8.2.2 doplnený o symboly z množiny $FOLLOW$

Výsledný $LR(0)$-automat, ku ktorému sme k redukčným položkám, resp. k akceptačnej položke, doplnili symboly z množiny $FOLLOW$ pre neterminály na ľavých stranách, je zobrazený na obrázku 8.5.
Tabuľka $ACTION$:

$ACTION$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
a	$R4$	P		$R4$			$R4$	$R3$
b	$R4$			$R4$		P	$R4$	$R3$
c	P			P			P	
ε	$R2$		A		$R1$			

Tabuľka 8.9: Tabuľka $ACTION$ $SLR(1)$-analyzátora pre gramatiku z úlohy 8.2.2

Tabuľka $GOTO$:

$GOTO$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
a		s_4						
b						s_6		
c	s_3			s_3			s_3	
S'								
S	s_2							
A	s_1			s_5			s_7	

Tabuľka 8.10: Tabuľka $GOTO$ $SLR(1)$-analyzátora pre gramatiku z úlohy 8.2.2

Keďže v tabuľke $ACTION$, ktorú sme zostrojili, sa **nenachádza konflikt**, t. j. v každej bunke je signalizovaná najviac jedna akcia, zadaná gramatika **je** $SLR(1)$**-gramatikou**.

Úloha č. 8.2.3 Je daná bezkontextová gramatika $G = (\{S, A, B, C\}, \{a, b, c, d\}, P, S)$, ktorej pravidlá P sú:

1. $S \to aAb$
2. $S \to aA$
3. $S \to Bc$
4. $A \to Bb$
5. $A \to Cc$
6. $A \to d$
7. $B \to aA$
8. $C \to aA$

Pre uvedenú gramatiku zostrojte $SLR(1)$-syntaktický analyzátor a určte, či ide o $SLR(1)$-gramatiku.

Riešenie:
Množiny $FIRST$ a $FOLLOW$ pre neterminály tejto gramatiky sú:

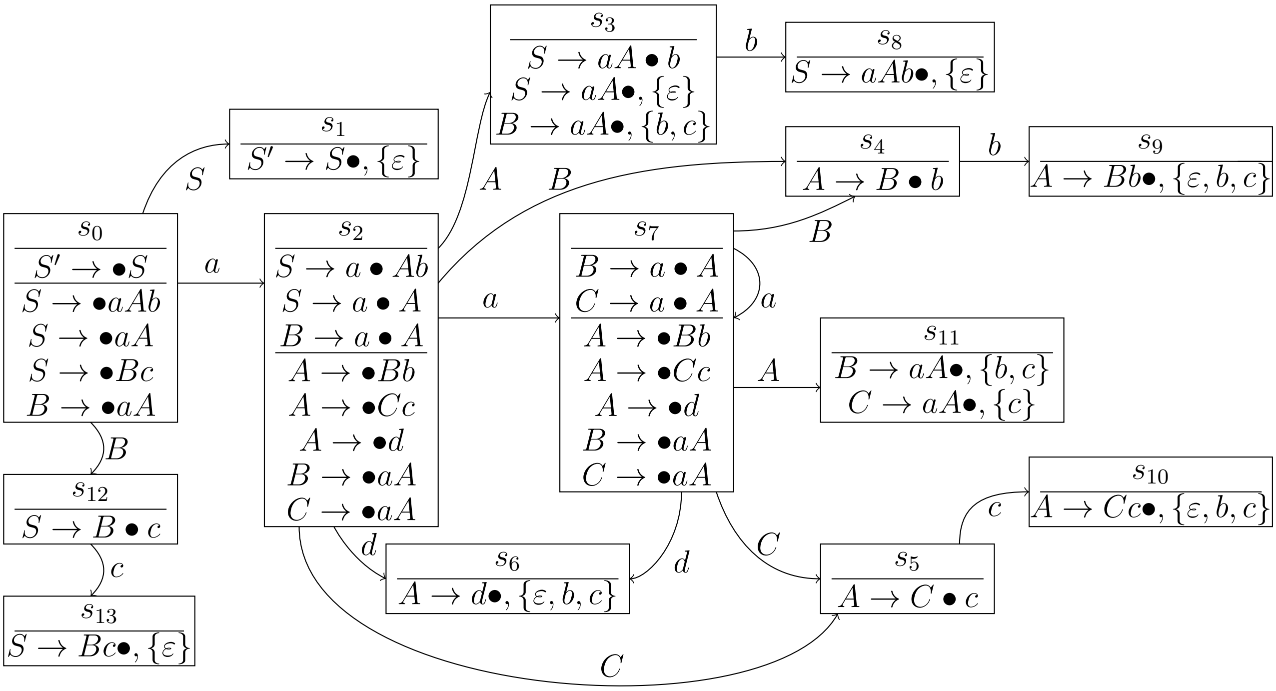
	S	A	B	C
$FIRST$	$\{a\}$	$\{a, d\}$	$\{a\}$	$\{a\}$
$FOLLOW$	$\{\varepsilon\}$	$\{b, c, \varepsilon\}$	$\{b, c\}$	$\{c\}$

$LR(0)$-automat, ku ktorému sme k redukčným položkám, resp. k akceptačnej položke, doplnili symboly z množiny $FOLLOW$ pre neterminály na ľavých stranách, je zobrazený na obrázku 8.6.

Tabuľka $ACTION$:

$ACTION$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}
a	P		P					P						
b				$P/R7$	P		$R6$			$R4$	$R5$	$R7$		
c				$R7$		P	$R6$			$R4$	$R5$	$R7/R8$	P	
d			P					P						
ε		A		$R2$			$R6$		$R1$	$R4$	$R5$			$R3$

Tabuľka 8.11: Tabuľka $ACTION$ $SLR(1)$-analyzátora pre gramatiku z úlohy 8.2.3



Obr. 8.6: $LR(0)$-automat ku gramatike z úlohy 8.2.3 doplnený o symboly z množiny $FOLLOW$

Tabuľka $GOTO$:

$GOTO$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9	s_{10}	s_{11}	s_{12}	s_{13}
a	s_2		s_7					s_7						
b				s_8	s_9									
c						s_{10}							s_{13}	
d			s_6					s_6						
S'														
S	s_1													
A			s_3					s_{11}						
B	s_{12}		s_4					s_4						
C			s_5					s_5						

Tabuľka 8.12: Tabuľka $GOTO$ $SLR(1)$-analyzátora pre gramatiku z úlohy 8.2.3

Keďže v tabuľke $ACTION$, ktorú sme zostrojili, sa **nachádzajú konflikty**:

- presun/redukcia v bunke $ACTION[s_3, b]$,
- redukcia/redukcia v bunke $ACTION[s_{11}, c]$,

zadaná gramatika **nie je** $SLR(1)$**-gramatikou**.

8.3 Syntaktická analýza pomocou LR syntaktického analyzátora

Úloha č. 8.3.1 Je daná bezkontextová gramatika $G = (\{S, A, B\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

1. $S \to aSb$
2. $S \to A$
3. $A \to ab$
4. $A \to B$
5. $B \to c$

Pre uvedenú gramatiku zostrojte nejaký typ LR syntaktického analyzátora a zistite, či majú nasledovné reťazce v uvedenej gramatike deriváciu: $acb, aabb, aab, bca, aba$.

Ak derivácia reťazca existuje, zistite, ako vyzerá **pravá derivácia** príslušného reťazca.

Riešenie:
Keďže o danej gramatike sme v úlohe č. 8.1.1 zistili, že ide o $LR(0)$-gramatiku, existuje pre ňu príslušný $LR(0)$-syntaktický analyzátor, ktorého tabuľky $ACTION$ a $GOTO$ majú tvar:

$ACTION$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
	P	A	P	$R2$	$R4$	$R5$	P	$R3$	$R1$

$GOTO$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
a	s_2		s_2						
b			s_7				s_8		
c	s_5		s_5						
S'									
S	s_1		s_6						
A	s_3		s_3						
B	s_4		s_4						

$LR(0)$-syntaktický analyzátor má formu špecifického zásobníkového automatu, ktorého vstupom je reťazec, o ktorom chceme zistiť, či má deriváciu a ktorého počiatočným zásobníkovým symbolom je počiatočný stav príslušného $LR(0)$-automatu s_0. Následne $LR(0)$-syntaktický analyzátor opakuje nasledovné činnosti:

- Ak je na vrchu zásobníka stavový symbol s_i, pre ktorý je v tabuľke $ACTION$ uvedená akcia **presun** a na vstupe sa nachádza vstupný symbol t, syntaktický analyzátor vloží do zásobníka stavový symbol s_j, ktorý sa nachádza v tabuľke $GOTO$ na pozícii $GOTO[s_i, t]$ a vstupný symbol t označí za prečítaný.
- Ak je na vrchu zásobníka stavový symbol s_i, pre ktorý je v tabuľke $ACTION$ uvedená akcia **redukcia** podľa pravidla $A \to \alpha$, najprv sa zo zásobníka odstráni $|\alpha|$ stavových symbolov, t. j. toľko symbolov, koľko symbolov gramatiky sa nachádza na pravej strane pravidla $A \to \alpha$. Po tomto odstránení sa na vrch zásobníka dostane nejaký stavový symbol, povedzme s_j. Následne sa na vrch zásobníka vloží stavový symbol s_k, ktorý sa nachádza v tabuľke $GOTO$ na pozícii $GOTO[s_j, A]$.
- Ak je na vrchu zásobníka stavový symbol s_i, pre ktorý je v tabuľke $ACTION$ uvedená akcia **akceptácia** a **vstup bol celý prečítaný**, syntaktický analyzátor akceptuje vstupný reťazec, teda vstupný reťazec **má v uvedenej gramatike deriváciu**.
- Ak je signalizovaná akcia presun, avšak vstup už bol celý prečítaný alebo je signalizovaná akcia akceptácia a na vstupe zostali neprečítané symboly, znamená to **syntaktickú chybu**.
- Podobne, ak sa pri akciách presun alebo redukcia má použiť symbol z tabuľky $GOTO[s, t]$, avšak na uvedenej pozícii sa v tabuľke $GOTO$ nenachádza žiaden stavový symbol, znamená to **syntaktickú chybu**.

- Syntaktická chyba znamená, že vstupný ret'azec **nemá v gramatike deriváciu.**

1. Syntaktická analýza ret'azca acb:

   Výpočet znázorníme tak, že uvedieme ešte nespracovanú čast' vstupného ret'azca, pričom v danom momente vidí syntaktický analyzátor prvý nespracovaný symbol zl'ava. Obsah zásobníka uvedieme *sklopený doprava*, kde prvý symbol sprava je zároveň symbol na vrchu zásobníka.

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	acb	Presun
s_0s_2	cb	Presun
$s_0s_2s_5$	b	Redukcia, $B \to c$
$s_0s_2s_4$	b	Redukcia, $A \to B$
$s_0s_2s_3$	b	Redukcia, $S \to A$
$s_0s_2s_6$	b	Presun
$s_0s_2s_6s_8$	ε	Redukcia, $S \to aSb$
s_0s_1	ε	Akceptácia

   Ked'že výpočet syntaktického analyzátora dospel do situácie, v ktorej bol vstup celý prečítaný a na vrchu zásobníka je stavový symbol s_1 signalizujúci akceptáciu, ret'azec acb má v uvedenej gramatike deriváciu. Pravá derivácia tohto ret'azca vznikne postupnou aplikáciou pravidiel použitých na redukcie, ak sa použijú v **opačnom poradí**, t. j. $S \to aSb, S \to A, A \to B, B \to c$.

2. Syntaktická analýza ret'azca $aabb$:

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	$aabb$	Presun
s_0s_2	abb	Presun
$s_0s_2s_2$	bb	Presun
$s_0s_2s_2s_7$	b	Redukcia, $A \to ab$
$s_0s_2s_3$	b	Redukcia, $S \to A$
$s_0s_2s_6$	b	Presun
$s_0s_2s_6s_8$	ε	Redukcia, $S \to aSb$
s_0s_1	ε	Akceptácia

   Ked'že výpočet syntaktického analyzátora dospel do situácie, v ktorej bol vstup celý prečítaný a navrchu zásobníka je stavový symbol s_1 signalizujúci akceptáciu, ret'azec $aabb$ má v uvedenej gramatike deriváciu. Pravá derivácia tohto ret'azca vznikne postupnou aplikáciou pravidiel použitých na redukcie, ak sa použijú v **opačnom poradí**, t. j. $S \to aSb, S \to A, A \to ab$.

3. Syntaktická analýza ret'azca aab:

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	aab	Presun
s_0s_2	ab	Presun
$s_0s_2s_2$	b	Presun
$s_0s_2s_2s_7$	ε	Redukcia, $A \to ab$
$s_0s_2s_3$	ε	Redukcia, $S \to A$
$s_0s_2s_6$	ε	**Syntaktická chyba**

Vznikla situácia, v ktorej je signalizovaný presun, pretože na vrchu zásobníka je stavový symbol s_6, avšak na vstupe sa už **nenachádza** žiaden symbol. Znamená to teda, že prišlo k syntaktickej chybe a ret'azec aab **nemá** v danej gramatike deriváciu.

4. Syntaktická analýza ret'azca bca:

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	bca	**Syntaktická chyba**

Vznikla situácia, v ktorej je signalizovaný presun, pretože na vrchu zásobníka je stavový symbol s_0. Do zásobníka by sa mal vložit' stavový symbol, ktorý je v tabul'ke *GOTO* na pozícii $GOTO[s_0, b]$. Ked'že táto pozícia je v tabul'ke *GOTO* **prázdna**, znamená to, že prišlo k syntaktickej chybe a ret'azec bca **nemá** v danej gramatike deriváciu.

5. Syntaktická analýza ret'azca aba:

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	aba	Presun
s_0s_2	ba	Presun
$s_0s_2s_7$	a	Redukcia, $A \to ab$
s_0s_3	a	Redukcia, $S \to A$
s_0s_1	a	**Syntaktická chyba**

Vznikla situácia, v ktorej je signalizovaná akceptácia, pretože na vrchu zásobníka sa nachádza stavový symbol s_1. Avšak akceptácia je možná len v prípade, že bol vstup celý prečítaný. Ked'že v aktuálnej situácii zostal na vstupe neprečítaný symbol a, znamená to, že došlo k syntaktickej chybe a ret'azec aba **nemá** v gramatike deriváciu.

Úloha č. 8.3.2 Je daná bezkontextová gramatika $G = (\{S, A\}, \{a, b, c\}, P, S)$, ktorej pravidlá P sú:

1. $S \to Aa$
2. $S \to \varepsilon$
3. $A \to cAbA$
4. $A \to \varepsilon$

Pre uvedenú gramatiku zostrojte nejaký typ LR syntaktického analyzátora a zistite, či majú nasledovné reťazce v uvedenej gramatike deriváciu: $cba, \varepsilon, cbaa, cb$.

Ak derivácia reťazca existuje, zistite, ako vyzerá **pravá derivácia** príslušného reťazca.

Riešenie:
Keďže o danej gramatike sme v úlohe č. 8.2.2 zistili, že ide o $SLR(1)$-gramatiku, existuje pre ňu príslušný $SLR(1)$-syntaktický analyzátor, ktorého tabuľky $ACTION$ a $GOTO$ majú tvar:

$ACTION$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
a	$R4$	P		$R4$			$R4$	$R3$
b	$R4$			$R4$		P	$R4$	$R3$
c	P			P			P	
ε	$R2$		A		$R1$			

$GOTO$	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
a		s_4						
b						s_6		
c	s_3			s_3			s_3	
S'								
S	s_2							
A	s_1			s_5			s_7	

$SLR(1)$-syntaktický analyzátor má formu špecifického zásobníkového automatu, ktorého vstupom je reťazec, o ktorom chceme zistiť, či má deriváciu a ktorého počiatočným zásobníkovým symbolom je počiatočný stav príslušného $SLR(1)$-automatu s_0. Následne $SLR(1)$-syntaktický analyzátor opakuje nasledovné činnosti:

- Ak je na vrchu zásobníka stavový symbol s_i a na vstupe terminálny symbol t, pre ktoré je v tabuľke $ACTION$ na pozícii $ACTION[s_i, t]$ uvedená akcia **presun**, syntaktický analyzátor vloží do zásobníka stavový symbol s_j, ktorý sa nachádza v tabuľke $GOTO$ na pozícii $GOTO[s_i, t]$ a vstupný symbol t označí za prečítaný.

- Ak je na vrchu zásobníka stavový symbol s_i a na vstupe terminálny symbol t (v situácii, že bol už vstup celý prečítaný, berieme $t = \varepsilon$), pre ktorý je v tabuľke $ACTION$ na pozícii $ACTION[s_i, t]$ uvedená akcia **redukcia** podľa pravidla $A \to \alpha$, najprv sa zo zásobníka odstráni $|\alpha|$ stavových symbolov, t. j. toľko symbolov, koľko symbolov gramatiky sa nachádza na pravej strane pravidla $A \to \alpha$. Po tomto odstránení sa na vrch zásobníka dostane nejaký stavový symbol, povedzme s_j. Následne sa na vrch zásobníka vloží stavový symbol s_k, ktorý sa nachádza v tabuľke $GOTO$ na pozícii $GOTO[s_j, A]$.
- Ak je na vrchu zásobníka stavový symbol s_i a vstup bol celý prečítaný, t. j. $t = \varepsilon$ a v tabuľke $ACTION$ je na pozícii $ACTION[s_i, \varepsilon]$ uvedená akcia **akceptácia**, syntaktický analyzátor akceptuje vstupný reťazec, teda vstupný reťazec **má v uvedenej gramatike deriváciu**.
- Ak pre aktuálny stavový symbol na vrchu zásobníka s a aktuálny vstupný symbol t (prípadne $t = \varepsilon$ ak bol vstup celý prečítaný) nie je v tabuľke $ACTION$ na pozícii $ACTION[s, t]$ žiadna akcia, došlo k syntaktickej chybe.
- Syntaktická chyba znamená, že vstupný reťazec **nemá v gramatike deriváciu.**

1. Syntaktická analýza reťazca cba:

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	cba	Presun
s_0s_3	ba	Redukcia, $A \to \varepsilon$
$s_0s_3s_5$	ba	Presun
$s_0s_3s_5s_6$	a	Redukcia, $A \to \varepsilon$
$s_0s_3s_5s_6s_7$	a	Redukcia, $A \to cAbA$
s_0s_1	a	Presun
$s_0s_1s_4$	ε	Redukcia, $S \to Aa$
s_0s_2	ε	Akceptácia

Len pre istotu upozorňujeme, že v situácii, keď je signalizovaná redukcia podľa pravidiel $S \to \varepsilon, A \to \varepsilon$ a zo zásobníka je potrebné najprv odstrániť toľko symbolov, aká je veľkosť pravej strany pravidiel, tak v tomto prípade sa zo zásobníka odstráni nula symbolov, keďže veľkosť pravej strany, ktorú tvorí ε, je nula, a teda sa priamo pristúpi ku vkladaniu symbolu do zásobníka na základe tabuľky $GOTO$.

Keďže výpočet syntaktického analyzátora dospel do situácie, v ktorej je navrchu zásobníka symbol s_2, vstup je celý prečítaný ($t = \varepsilon$) a v tabuľke $ACTION$ je na pozícii $ACTION[s_2, \varepsilon]$ akcia akceptácia, $SLR(1)$-syntaktický analyzátor reťazec acb akceptuje, a teda reťazec acb má v uvedenej gramatike deriváciu. Pravá derivácia tohto reťazca vznikne postupnou aplikáciou pravidiel použitých na redukcie, ak sa použijú v **opačnom poradí**, vždy na prvý neterminál sprava, t. j.:

$$S \Rightarrow_r Aa \Rightarrow_r cAbAa \Rightarrow_r cAba \Rightarrow_r cba$$

2. Syntaktická analýza ret'azca ε:

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	ε	Redukcia, $S \to \varepsilon$
$s_0 s_2$	ε	Akceptácia

Vidíme, že v $SLR(1)$-syntaktickom analyzátore je možné aj v prípade, že je na vstupe ε, vykonat' nejaké akcie, konkrétne redukciu podľa pravidla $S \to \varepsilon$, ak je na vrchu zásobníka stav s_0. Vd'aka tomu vidíme, že ret'azec ε má v gramatike deriváciu, ked'že ho $SLR(1)$-syntaktický analyzátor akceptoval.

3. Syntaktická analýza ret'azca $cbaa$:

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	$cbaa$	Presun
$s_0 s_3$	baa	Redukcia, $A \to \varepsilon$
$s_0 s_3 s_5$	baa	Presun
$s_0 s_3 s_5 s_6$	aa	Redukcia, $A \to \varepsilon$
$s_0 s_3 s_5 s_6 s_7$	aa	Redukcia, $A \to cAbA$
$s_0 s_1$	aa	Presun
$s_0 s_1 s_4$	a	**Syntaktická chyba**

Došlo k syntaktickej chybe, pretože na vrchu zásobníka sa nachádza stavový symbol s_4, na vstupe symbol a, avšak v tabuľke $ACTION$ nie je na pozícii $ACTION[s_4, a]$ uvedená žiadna akcia. Ret'azec $cbaa$ teda nemá v gramatike deriváciu.

Len pre istotu upozorňujeme, že v danej situácii **nie je aplikovateľná** akcia na pozícii $ACTION[s_4, \varepsilon]$, ked'že riadok označený ε v tabuľke $ACTION$ sa týka **len situácie, že bol vstup celý prečítaný.**

4. Syntaktická analýza ret'azca cb:

Zásobník $\sqsubset$	Zvyšok vstupu	Akcia
s_0	cb	Presun
$s_0 s_3$	b	Redukcia, $A \to \varepsilon$
$s_0 s_3 s_5$	b	Presun
$s_0 s_3 s_5 s_6$	ε	**Syntaktická chyba**

Vidíme, že došlo k syntaktickej chybe, pretože na vrchu zásobníka sa nachádza stavový symbol s_6 a vstup bol celý prečítaný ($t = \varepsilon$), avšak v tabuľke $ACTION$ nie je na pozícii $ACTION[s_6, \varepsilon]$ uvedená žiadna akcia. Ret'azec cb teda nemá v gramatike deriváciu.

Literatúra

[1] AHO, A. V., et. al. 2006. *Compilers : Principles, Techniques, & Tools*. 2. vyd. Reading : Addison-Wesley, 2006. 1038 s. ISBN 0-321-48681-1.

[2] DEDERA, Ľ. 2014. *Počítačové jazyky a ich spracovanie*. Liptovský Mikuláš : Akadémia ozbrojených síl generála M. R. Štefánika, 2014. 286 s. ISBN 978-80-8040-503-8.

[3] HOPCROFT, J. E. - MOTWANI, R. - ULLMAN, J. D. 2006. *Introduction to Automata Theory, Languages, and Computation*. 3. vyd. Boston : Addison-Wesley, 2006. 550 s. ISBN 0-321-45536-3.

[4] LINZ, P. 2017. *An Introduction to Formal Languages and Automata*. 6. vyd. Burlington : Jones & Bartlett Learning, 2017. 454 s. ISBN 978-1-284-07724-7.

[5] MOLNÁR, Ľ. - ČEŠKA, M. - MELICHAR, B. 1987. *Gramatiky a jazyky*. Bratislava : Alfa - vydavateľstvo technickej a ekonomickej literatúry, 1987. 192 s.

[6] SIPSER, M. 2013. *Introduction to the Theory of Computation*. 3. vyd. Boston : Cengage Learning, 2013. 482 s. ISBN 978-1-133-18779-0.

Ing. Viliam Hromada, PhD.

ZBIERKA RIEŠENÝCH ÚLOH Z PREDMETU AUTOMATY A FORMÁLNE JAZYKY

Vydala Slovenská technická univerzita v Bratislave vo Vydavateľstve SPEKTRUM STU, Bratislava, Vazovova 5, v roku 2023.

Edícia skrípt

Rozsah 210 strán, 35 obrázkov, 16 tabuliek 10,224 AH, 10,488 VH, 1. vydanie, edičné číslo 6155, vydané v elektronickej forme.

85 – 217 – 2023

ISBN 978-80-227-5320-3