Cvičenie 5 - Regulárne výrazy a konečné automaty

Ing. Viliam Hromada, PhD.

C-510
Ústav informatiky a matematiky
FEI STU

viliam.hromada@stuba.sk

21.10.2020

Regulárne výrazy

Popíšte pomocou regulárneho výrazu:

1. Reťazce z písmen a, b, ktoré začínajú alebo predponou ab alebo predponou ba.
2. Reťazce z písmen a, b, ktoré obsahujú ako podreťazec $abba$.
3. Reťazce z písmen a, b, ktoré obsahujú ako podreťazec aaa alebo bbb.
4. Reťazce z číslic $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$, ktoré predstavujú kladné nepárne čísla.
5. Reťazce z číslic $0, 1$ ktoré predstavujú binárne vyjadrenie nepárnych čísiel.
6. Reťazce z číslic $0, 1$, ktoré obsahujú párny počet symbolov 0.
7. Reťazce z písmen a, b, ktoré sú párnej dĺžky.

Reťazce z písmen a, b, ktoré začínajú alebo predponou ab alebo predponou ba.

- Reťazce sú teda všetky v tvare, že začínajú alebo ab, alebo ba a za týmto prefixom môže nasledovať hocijaká postupnosť symbolov a, b.
- Regulárny výraz, reprezentujúci **hocijakú** postupnosť a, b vytvoríme ľahko:
  - Zjednotíme $a \cup b$, t.j. regex: **a+b**
  - A následne toto zjednotenie iterujeme, t.j. regex: **(a+b)***, čím dostávame **všetky reťazce** zložené z a a b
- Pripúšťame 2 prefixy: ab alebo ba, čiže zjednotenie ab a ba. T.j. regex: **ab + ba**
- Aby sme vyjadrili ,že požadované reťazce sú v tvare, že začínajú alebo ab, alebo ba, za ktorými môže už ísť hocičo, tak vlastne potrebujeme zreťaziť **ab + ba** a **(a+b)***. Takže výsledný regulárny výraz:

$$R = \textbf{(ab + ba)(a+b)}^*$$

Reťazce z písmen a, b, ktoré obsahujú ako podreťazec $abba$.

- To, že reťazec obsahuje nejaký konkrétny podreťazec znamená, že sa dá rozdeliť na 3 časti:
  1. Ľubovoľný prefix, t.j. ľubovoľná postupnosť a, b
  2. Konkrétna časť $abba$
  3. Ľubovoľný sufix, t.j. ľubovoľná postupnosť a, b.
- Už vieme, že ľubovoľná postupnosť a, b je daná regexom: **(a+b)***.
- Reťazec $abba$ je daný regexom: **abba**.
- Keď ich zreťazíme tak, aby popisovali reťazce, ktoré v sebe vnútri obsahujú $abba$, ale teoreticky môžu pred a za $abba$ obsahovať ľubovoľnú postupnosť z a a b, tak dostávame:

$$R = \textbf{(a+b)}^*\textbf{(abba)}\textbf{(a+b)}^*$$

Reťazce z písmen a, b**, ktoré obsahujú ako podreťazec** *aaa* **alebo** *bbb***.**

- Táto úloha je len jednoduchým rozšírením predchádzajúcej úlohy, kde uvažujeme alebo reťazce obsahujúce *aaa*, alebo reťazce obsahujúce *bbb*.
- T.j. "vnútorná časť" je popísateľná regexom: **(aaa + bbb)**.
- Znovu platí, že pred a za **aaa + bbb** môže byť ľubovoľná postupnosť symbolov a, b.
- Teda výsledok:

$$R = \textbf{(a+b)}^*\textbf{(aaa + bbb)(a+b)}^*$$

Reťazce z číslic $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$**, ktoré predstavujú nezáporné párne čísla.**

- Každé párne číslo končí $0, 2, 4, 6, 8$.
- Navyše by sme chceli, aby čísla nemohli začínať nulou, tak ako sme zvyknutí z bežného života.
- T.j. prvá číslica nemôže byť nula a posledná musí byť $0, 2, 4, 6, 8$. Ostatné môžu byť ľubovoľné.

$$\mathbf{(1+2+3+4+5+6+7+8+9)(1+2+3+4+5+6+7+8+9+0)^*(0+2+4+6+8)}$$

- Takýto výraz však neobsahuje jednociferné párne čísla. Tie vieme pridať explicitne:

$$\mathbf{(1+2+3+4+5+6+7+8+9)(1+2+3+4+5+6+7+8+9+0)^*(0+2+4+6+8) +(0+2+4+6+8)}$$

- Prípadne efektívnejší zápis:

$R = (\varepsilon$ **+ (1+2+3+4+5+6+7+8+9)(1+2+3+4+5+6+7+8+9+0)*)(0+2+4+6+8)**

Reťazce z číslic $0, 1$ **ktoré predstavujú binárne vyjadrenie nepárnych čísiel.**

- Každé nepárne binárne číslo končí 1.
- Navyše by sme chceli, aby čísla nemohli začínať nulou, t.j. musia začínať jednotkou
- T.j. prvá číslica nemôže byť nula a posledná musí byť 1. Ostatné môžu byť ľubovoľné.

$$R = \mathbf{1\ (\ 0 + 1)^*1}$$

Reťazce z číslic $0, 1$, ktoré obsahujú párny počet symbolov 0.

- Takéto reťazce môžu na ľubovoľnom mieste obsahovať nuly alebo jednotky, avšak počet núl musí byť deliteľný dvomi (t.j. napr $\varepsilon, 1, 11, 1001, 0011, 0110, 0000, ...$)
- Ak chceme popísať takéto reťazce regulárnym výrazom, musíme si dávať pozor, aby **výraz nepopísal niečo iné!!!**
- Napríklad výraz $(0 + 1)^*$ určite popisuje všetky výrazy, ktoré majú v sebe párny počet núl. Avšak popisuje aj reťazce, v ktorých je počet núl nepárny! Napríklad 101. Preto tento popis je **zle!**.
- Iný príklad, výraz $(1 + 00)^*$, je už lepším pokusom o riešenie. Každý ním popísaný reťazec určite obsahuje párne veľa núl. Problém je, že nedokáže popísať **všetky reťazce s párnym počtom núl**, napríklad 1010 tento regex nepopisuje.
- Preto je potrebné zabezpečiť nielen to, aby počet núl vo výslednom reťazci bol párny, ale aby aj vedel výraz popísať **každý výraz s párnym počtom núl**.

- Kľúčom k riešeniu je uvedomiť si, že každý reťazec, ktorý obsahuje párne veľa núl, sa dá rozdeliť na bloky, v ktorých sú dve nuly, pre ktoré platí, že medzi sebou, pred sebou a za sebou môžu mať ľubovoľný počet jednotiek.

  $1^*01^*01^*$
- A takéto bloky dvoch núl obklopených jednotkami sa môžu ľubovoľne opakovať.

  $(1^*01^*01^*)^*$
- Takýto výraz dokáže korektne popísať všetky reťazce, kde sa nachádza párny počet núl. Jediný problém je, že nedokáže popísať reťazce, kde nuly nie sú, t.j. kde sú samé jednotky. Ale to vieme vyriešiť elegantne:

  $(1^*01^*01^*)^* + 1^*$
- Korektných riešení existuje viacej, iné je napríklad:

  $(1 + 01^*0)^*$

Reťazce z písmen a, b, ktoré sú párnej dĺžky.

- Takéto reťazce sú dĺžky 0, 2, 4, 6, 8, ...
- Žiaľ, regulárny výraz podľa definície neumožňuje napísať niečo ako $(a+b)^{2k}$.
- Avšak znovu je riešenie pomerne jednoduché, keď si uvedomíme, že reťazce párnej dĺžky sa dajú napísať ako opakovanie nejakej skupiny symbolov...
- Každý reťazec párnej dĺžky sa dá rozdeliť na skupiny dvoch symbolov (lebo je párnej dĺžky a tá je teda deliteľná 2).
- Keďže uvažujeme všetky možné reťazce, skupina 2 symbolov môže obsahovať ľubovoľnú kombináciu 2 symbolov: aa, ab, ba, bb.
- Preto výsledok by mohol vyzerať aj:

$$R = \mathbf{(aa + ab + ba + bb)}^*$$

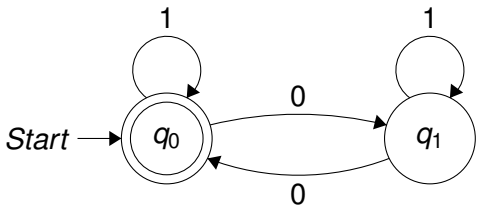
Konverzia DKA $\rightarrow$ Regex

- Na prednáške sme si spomínali, že každý DKA sa dá previesť na ekvivalentný regulárny výraz.
  1. **Doplň do DKA nový akceptačný stav** a z každého pôvodného akceptačného stavu doň veď ε**-prechod**. Zároveň pôvodné **akceptačné** stavy zmeň na **neakceptačné**.
  2. **Odstraňuj** všetky neakceptačné a nepočiatočné stavy z DKA. Zároveň po odstránení stavu **doplň** do automatu **nové prechody** tak, aby sa nezmenil jazyk, akceptovaný príslušným automatom.
  3. Keď ti **zostane** len konečný automat s **1 počiatočným a 1 akceptačným stavom**, popíš jazyk ním akceptovaný **regulárnym výrazom**.

  Počas spomínaného procesu budú prechody v automate popisované teraz už nie len symbolmi abecedy, ale regulárnymi výrazmi - ale tak, aby stále zostával zachovaný jazyk akceptovaný automatom.

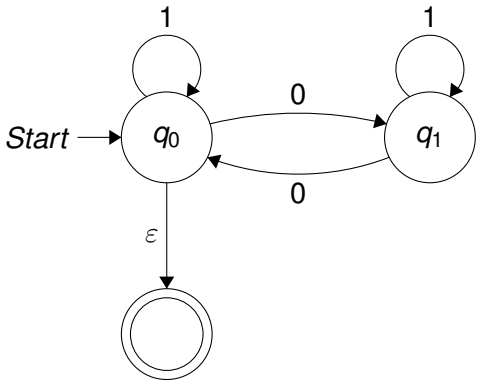
Príklad

Popíšte jazyk akceptovaný nasledovným automatom pomocou regexu:



(je to práve DKA, ktorý akceptuje jazyk, ktorý tvoria reťazce z núl a jednotiek, kde je párny počet núl)

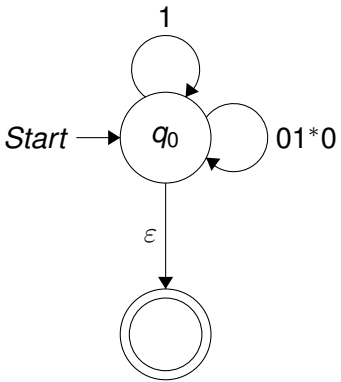
Podľa postupu, vložíme nový akceptačný stav (na mene nezáleží) a vedieme doň ε-prechody zo všetkých pôvodných akceptačných stavov (q_0). Zároveň pôvodné akceptačné stavy zmeníme na neakceptačné.



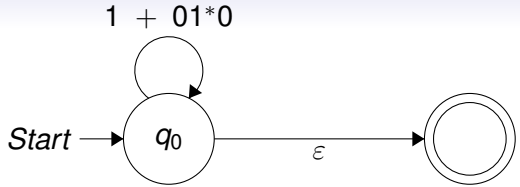
V ďalšom kroku chceme odstrániť stavy, ktoré nie sú počiatočné, ani akceptačné - t.j. stav q_1. Po jeho odstránení však musí zostať zachovaný jazyk, ktorý automat akceptuje. To znamená, že budeme musieť pridať do automatu nasledovné prechody:

- Zo stavu q_0 do stavu q_0 pre všetky také reťazce, ktoré sú v tvare 01^*0. Pretože týmito reťazcami bolo možné v automate zo stavu q_0 prejsť cez **odstraňovaný stav** q_1 do stavu q_0.
- Vznikne tým pádom druhá slučka v q_0, teraz ohodnotená výrazom 01^*0.
- Podobne by sme museli vyšetriť **všetky ďalšie možné prechody** cez stav q_1. Avšak v danom automate už nie sú iné stavy, ktorých by sa odstránenie q_1 dotklo.

Dostávame:



Pre jednoduchosť zjednotíme hrany, ktoré majú rovnaký začiatok a koniec, t.j. zjednotíme slučky v stave q_0 do 1 slučky, pričom prechod na ňu bude ohodnotený $1 + 01^*0$. Výsledok na ďalšom slajde (upravili sme polohu akceptačného stavu).



- Vidíme, že všetky reťazce, ktorými sa automat vie z počiatočného stavu dostať do stavu akceptačného sú v tvare, že na začiatku môže byť ľubovoľné opakovanie symbolov 1 alebo 01^*0, ktoré je nasledované ε.
- Všetky reťazce teda spĺňajú nasledovný regex:

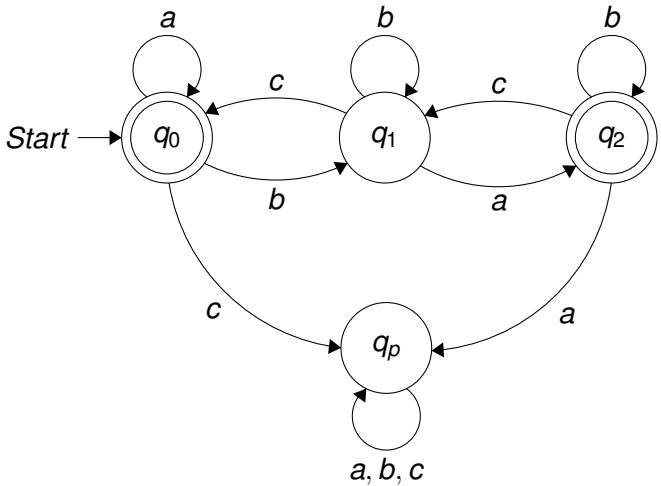
$$R = (1 + 01^*0)^*\varepsilon$$

- Čo je ekvivalentné s:

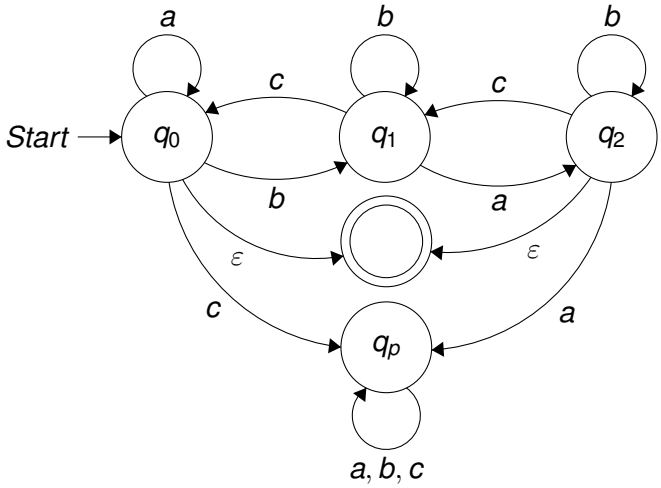
$$R = (1 + 01^*0)^*$$

Príklad 2

Preveďte nasledovný DKA na regulárny výraz:

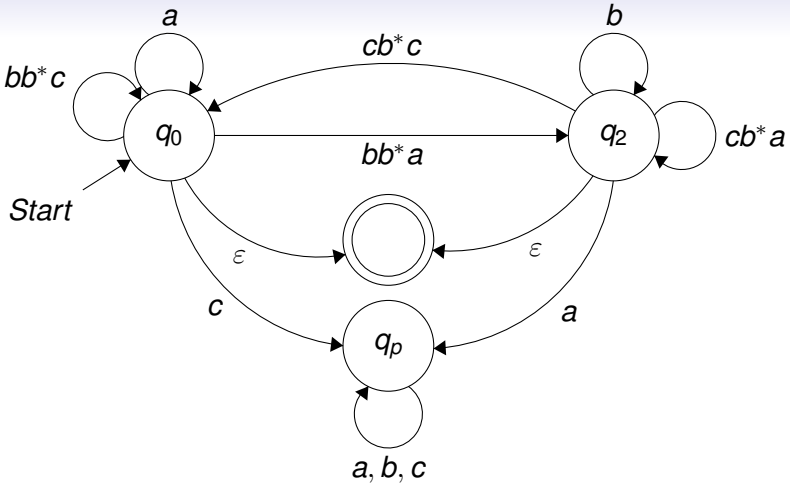


Prvý krok je vloženie nového akceptačného stavu, zo starých akceptačných stavov q_0, q_2 doň vedieme ε-prechody a zároveň z q_0, q_2 urobíme neakceptačné:

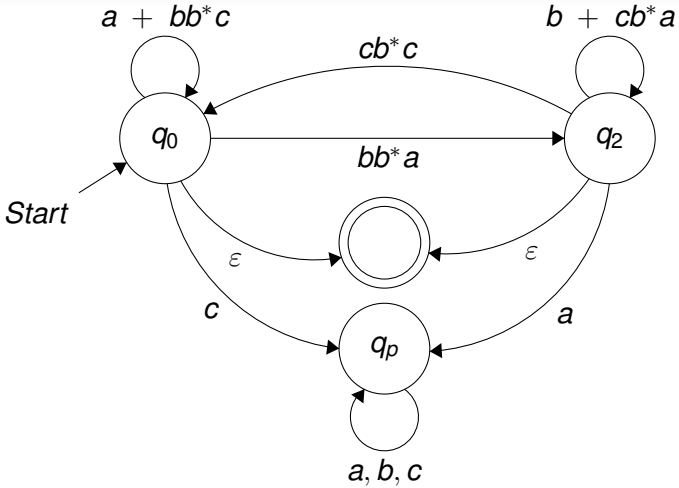


Odstráňme napríklad q_1. To znamená, že do automatu musíme doplniť prechody, ktoré to ovplyvní. Doplníme prechody:

- Z q_0 do q_2 na regulárny výraz bb^*a. Zdôvodnenie: keďže v automate je možné cez stav q_1 prejsť zo stavu q_0 do stavu q_2 na reťazec začínajúci b (prechod $q_0 \to q_1$), potom je možné slučkou čítať b-čka ($q_1 \to q_1$) a následne a-čkom prejsť z q_1 do q_2 ($q_1 \to q_2$).
- Podobne z q_2 do q_0 na regulárny výraz cb^*c.
- Z q_0 do q_0 na regulárny výraz bb^*c. Zdôvodnenie: V automate existuje cesta z q_0 do q_0 cez q_1 v tvare: znakom b prechod $q_0 \to q_1$, potom postupnosť b-čok v slučke $q_1 \to q_1$ a následne znakom c návrat $q_1 \to q_0$.
- Podobne z q_2 do q_2 na regulárny výraz cb^*a.
- Iné stavy nie sú priamo prepojené s q_1, a teda sa ich odstránenie q_1 nedotkne.

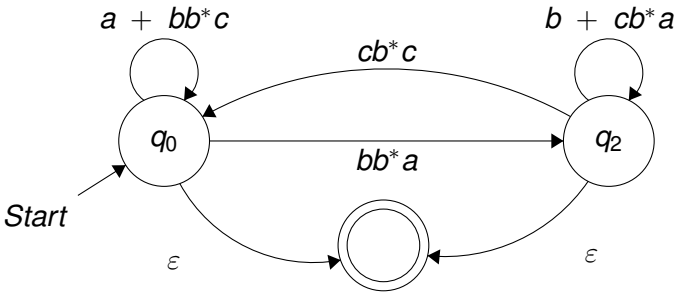


Môžeme zjednotiť regulárne výrazy na hranách so spoločným začiatkom a koncom (slučky v q_0 a q_2)


a + bb*c
b + cb*a
cb*c
q0
q2
bb*a
Start
ε
ε
c
a
qp
a, b, c

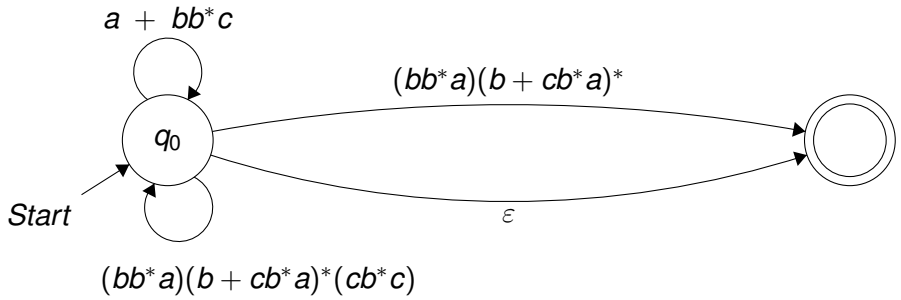
Ďalej odstráňme napríklad q_p. Zamyslime sa, o aké prechody v automate prídeme jeho odstránením:

- Do q_p idú prechody z q_0 a q_2. Avšak, z q_p žiaden prechod **nevychádza** a navyše, q_p **nie je akceptačný stav**.
- Preto v danom automate **ničím neprispieva** do jazyka, ktorý akceptuje daný automat.
- Pretože žiaden akceptovaný reťazec **nemá** takú výpočtovú cestu, že by sa počas jeho spracovania automat ocitol v stave q_p.
- Preto odstránenie takéhoto stavu **nijako nezmení jazyk automatu** a teda nemusíme dopĺňať žiadne hrany a stačí len stav odobrať.


a + bb*c
b + cb*a
cb*c
q0
q2
bb*a
Start
ε
ε

V automate zostali už len počiatočný stav q_0, akceptačný stav a neakceptačný nepočiatočný stav q_2. Preto už len odstránime q_2. Keďže s týmto stavom sú hranami prepojené stavy q_0 a akceptačný stav, jeho odstránenie musíme nahradiť nasledovnými hranami:

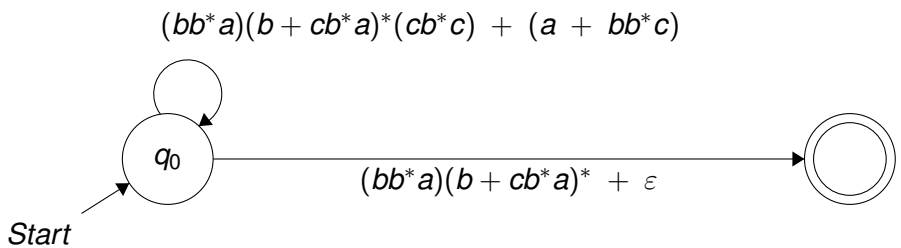
- Zo stavu q_0 do akceptačného stavu hranou, na ktorej je regulárny výraz: $(bb^*a)(b + cb^*a)^*\varepsilon$, prípadne len $(bb^*a)(b + cb^*a)^*$
- Zo stavu q_0 do stavu q_0 slučka, ohodnotená $(bb^*a)(b + cb^*a)^*(cb^*c)$


$a + bb^*c$
$(bb^*a)(b + cb^*a)^*$
q_0
Start
ε
$(bb^*a)(b + cb^*a)^*(cb^*c)$

Znovu môžeme zjednotiť hrany, ktoré majú spoločný začiatok a spoločný koniec:

- Slučky v stave q_0
- Hrany z q_0 do akceptačného stavu.

Dostávame výsledný KA:

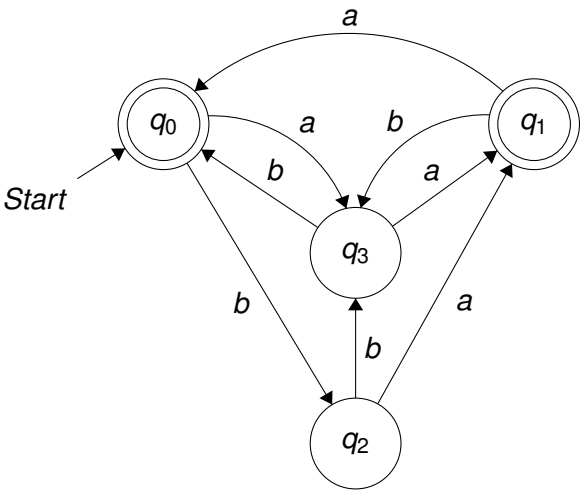


Výsledný regulárny výraz pôvodného DKA teda získame na základe upraveného prechodového diagramu tak, že ziterujeme regulárny výraz na slučke v q_0 a následne k tomu priraďazíme regulárny výraz z q_0 do akceptačného stavu:

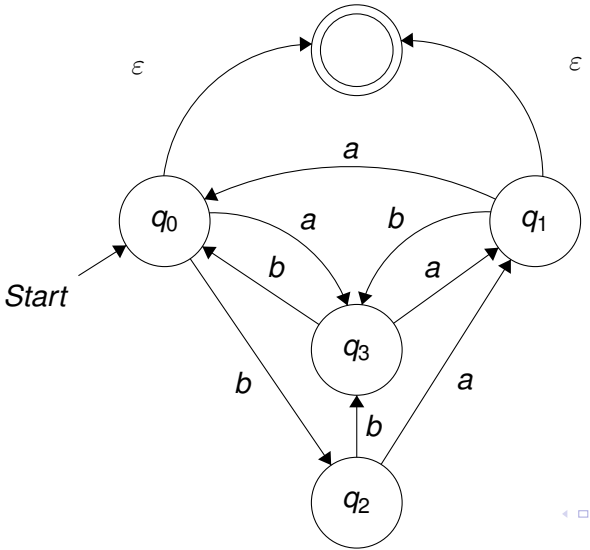
$$R = ((bb^*a)(b + cb^*a)^*(cb^*c) + (a + bb^*c))^*(\varepsilon + (bb^*a)(b + cb^*a)^*)$$

Príklad 3

Nájdite regulárny výraz popisujúci jazyk nasledovného DKA:

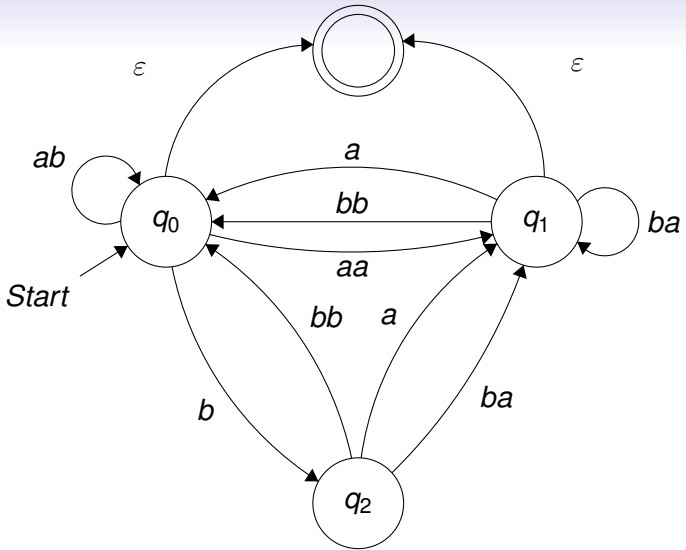


Najprv pridáme nový akceptačný stav, do ktorého vedieme ε-prechody zo starých akceptačných stavov q_0, q_1, zároveň zmeníme q_0, q_1 na neakceptačné:

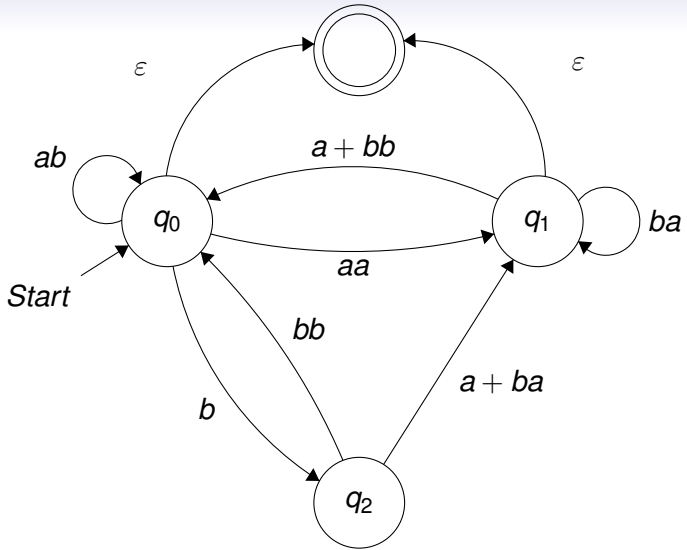


Odstráňme q_3. Všetky stavy, ktoré sú s q_3 spojené hranami sú q_0, q_1, q_2, preto musíme zohľadniť všetky možné cesty medzi týmito stavmi cez q_3. Takže súčasne s odstránením q_3 do automatu doplníme hrany:

- Z q_0 do q_0 na ab
- Z q_0 do q_1 na aa
- Z q_1 do q_1 na ba
- Z q_1 do q_0 na bb
- Z q_2 do q_0 na bb
- Z q_2 do q_1 na ba.

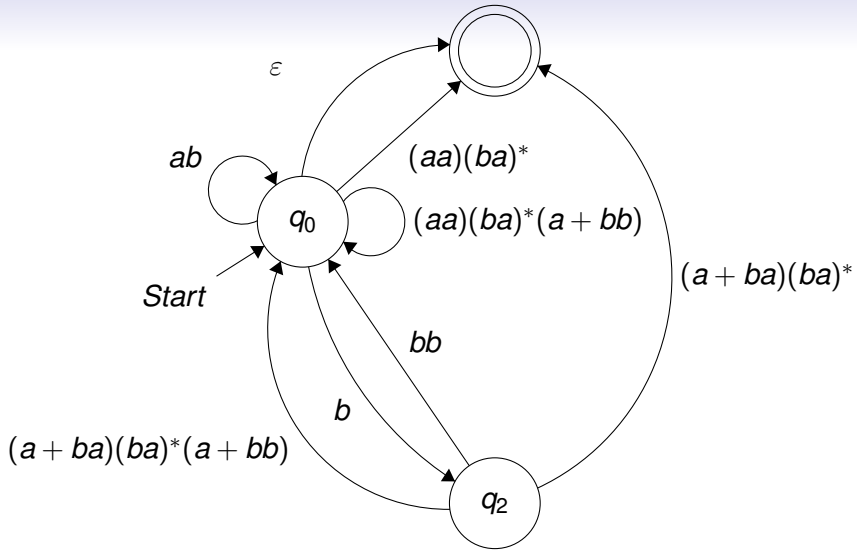


Zjednoťme hrany so spoločným začiatkom a koncom:

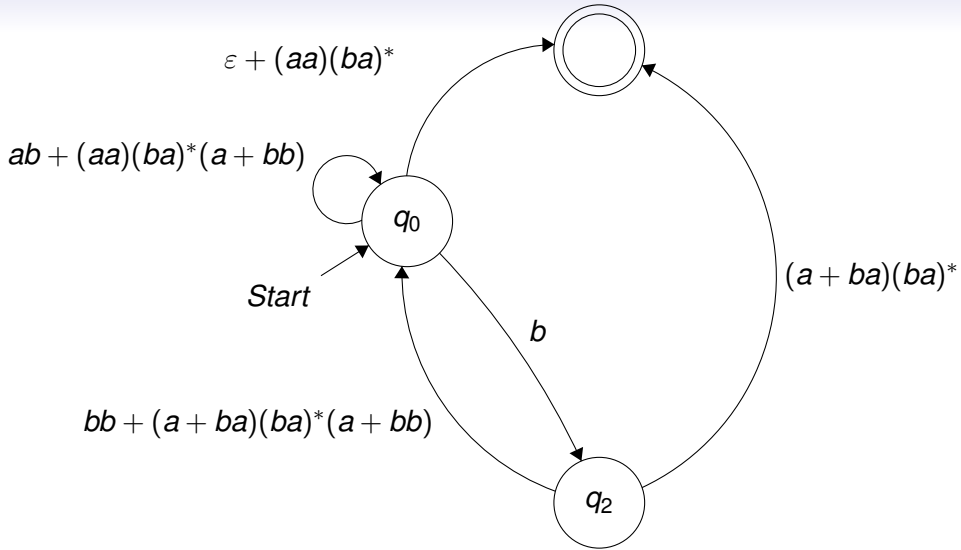

ε
ε
ab
a + bb
q0
q1
ba
aa
Start
bb
a + ba
b
q2

Odstráňme q_1. S q_1 sú hranami spojené stavy q_0, q_2 a akceptačný stav. Preto po odstránení q_1 musíme doplniť hrany:

- Slučku v q_0 pre $(aa)(ba)^*(a + bb)$
- Prechod z q_0 do akceptačného stavu pre $(aa)(ba)^*$
- Prechod z q_2 do akceptačného stavu pre $(a + ba)(ba)^*$
- Prechod z q_2 do q_0 pre $(a + ba)(ba)^*(a + bb)$



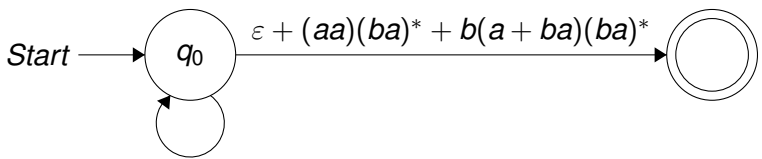
Zjednotíme hrany so spoločnými začiatkami a koncami.


$\varepsilon + (aa)(ba)^*$
$ab + (aa)(ba)^*(a + bb)$
q_0
Start
b
$(a + ba)(ba)^*$
$bb + (a + ba)(ba)^*(a + bb)$
q_2

A posledný stav, ktorý odstránime, je q_2. Po jeho odstránení je potrebné doplniť do prechodového diagramu:

- Slučku v q_0 pre $b(bb + (a + ba)(ba)^*(a + bb))$
- Prechod z q_0 do akceptačného stavu pre $b(a + ba)(ba)^*$

Po pridaní a **zlúčení novej a starej slučky v** q_0 dostávame finálnu verziu:


Start
q_0
$\varepsilon + (aa)(ba)^* + b(a + ba)(ba)^*$
$ab + (aa)(ba)^*(a + bb) + b(bb + (a + ba)(ba)^*(a + bb))$

Výsledný regulárny výraz:

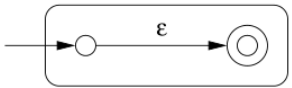
$$(ab+(aa)(ba)^*(a+bb)+b(bb+(a+ba)(ba)^*(a+bb)))^*(\varepsilon+(aa)(ba)^*+b(a+ba)(ba)^*)$$

Konverzia Regex $\rightarrow$ *DKA*

Na prednáške sme si spomínali, že každý regulárny výraz sa dá previesť na ε-NKA, ktorý akceptuje rovnaký jazyk, ako popisuje regulárny výraz, pričom konštrukcia KA vychádza z nasledovného:

- Zostroja sa elementárne ε-NKA ktoré akceptujú jednotlivé symboly gramatiky, ε, prípadne $\emptyset$.
- Následne sa z jednoduchších ε-NKA zostrojujú zložitejšie ε-NKA pre operácie zjednotenie, zreťazenie, iterácia.

ε-NKA pre symboly abecedy, ε, $\emptyset$

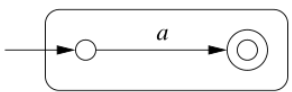


(a)

R = ε

(b)

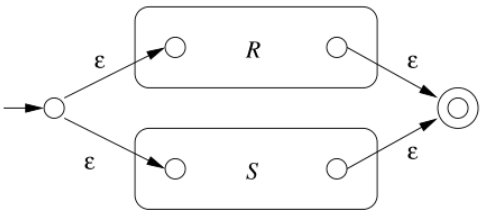
R = ∅



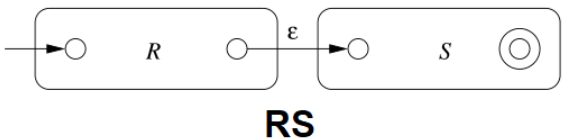
(c)

R = a

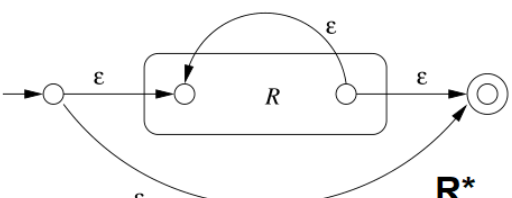
ε-NKA pre zjednotenie, zreťazenie, iteráciu



R+S



RS

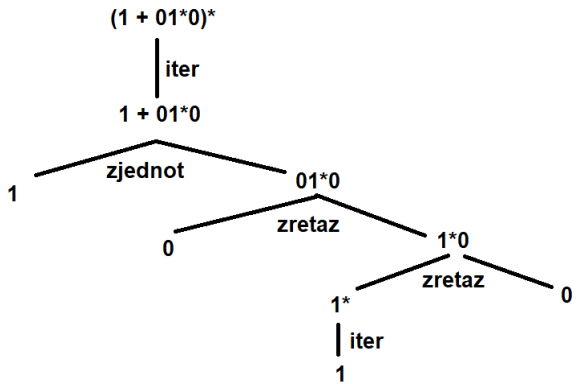


R*

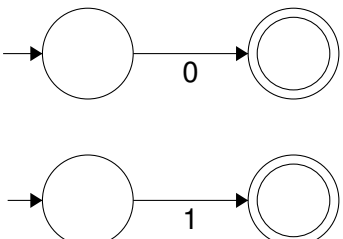
Príklad 1

Konvertujte regulárny výraz $(1 + 01^*0)^*$ na ε-NKA, ktorý akceptuje práve jazyk popísaný regulárnym výrazom.

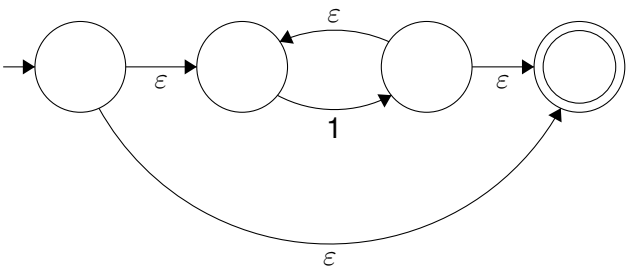
Regulárny výraz $(1 + 01^*0)^*$ si rozpíšeme ako postupnosť aplikácií operátorov zjednotenie, zreťazenie, iterácia na menšie výrazy, aby sme videli, ako bude prebiehať konštrukcia výsledného ε-NKA. Tým dostaneme v podstate strom aplikácií operátorov:



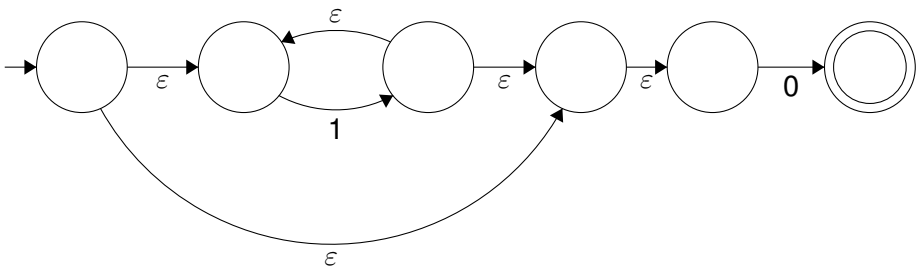
Začneme teda so základnými elementami, ktoré budú ε-NKA pre akceptáciu symbolov abecedy 0 a 1:



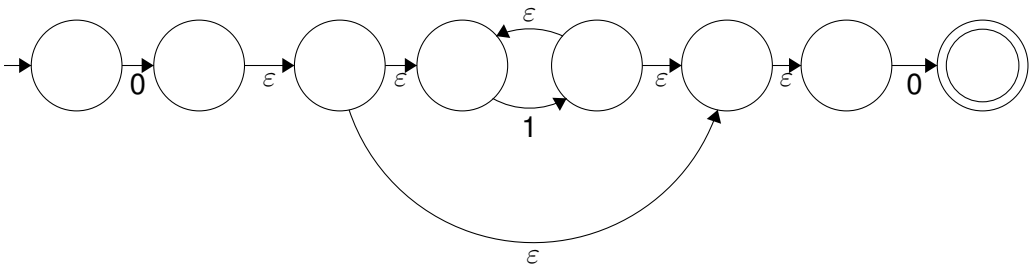
Pokračujeme tým, že konštruujeme automaty pre výrazy v danom strome aplikácií operátorov smerom zdola-nahor, od listov ku koreňu. Teda v ďalšom kroku zostrojíme ε-NKA pre výraz 1^* tým, že aplikujeme konštrukciu automatu pre operátor iterácie nad automatom pre výraz 1.



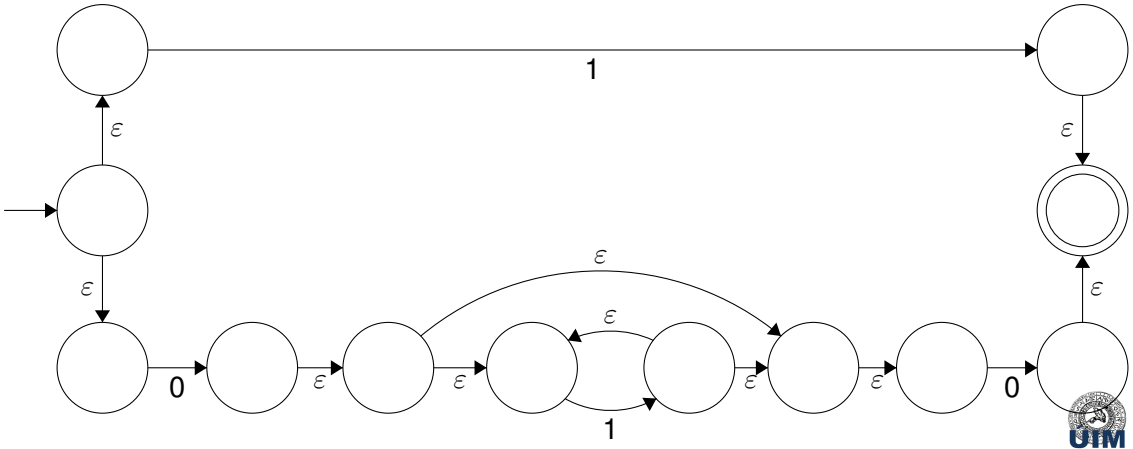
Keď už máme automaty pre výraz 1^* a výraz 0, môžeme zostrojiť automat pre ich zreťazenie 1^*0:



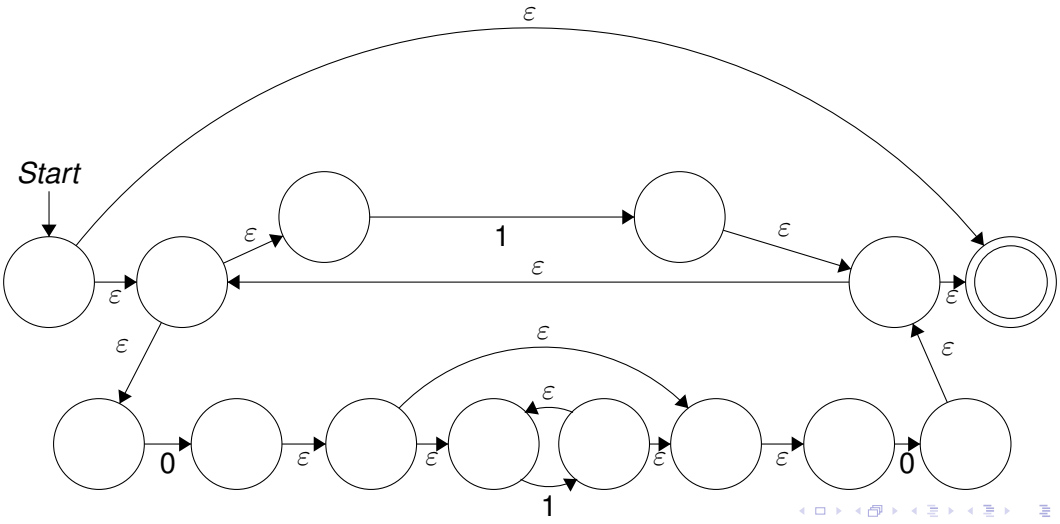
Keď už máme automaty pre výraz 0 a výraz 1^*0, môžeme zostrojiť automat pre ich zreťazenie 01^*0:



Keď už máme automaty pre výraz 1 a výraz 01^*0, môžeme zostrojiť automat pre ich zjednotenie $1 + 01^*0$:



Keď už máme automat pre výraz $1 + 01^*0$, môžeme zostrojiť automat pre iteráciu $(1 + 01^*0)^*$:



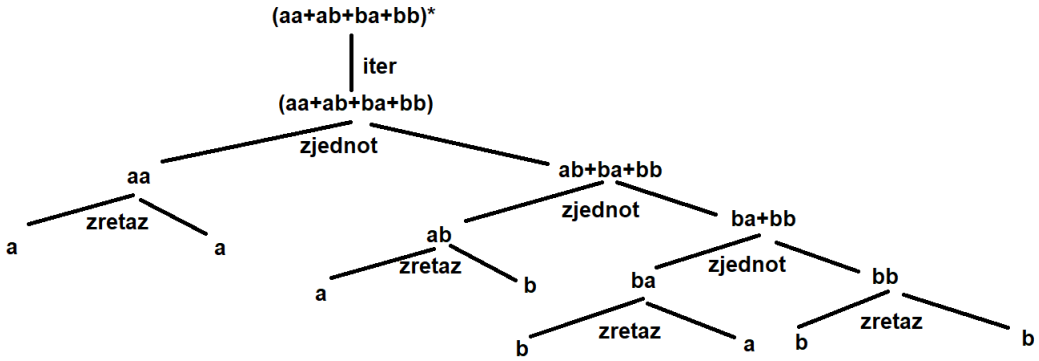
Výsledkom konverzie je teda ε-NKA, ktorý akceptuje práve jazyk popísaný regulárnym výrazom $(1 + 01^*0)^*$.

Pre praktické vyhľadávanie reťazcov spĺňajúcich tento regulárny výraz by sa ε-NKA previedol na ekvivalentný DKA pomocou algoritmu z prednášky č. 4 a následne by sa používal tento DKA.

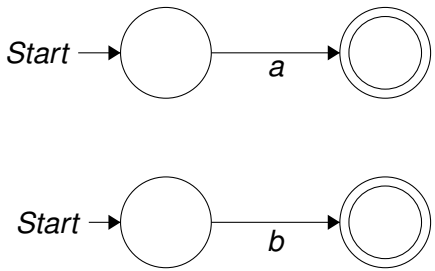
Príklad 2

Konvertujte regulárny výraz $(aa + ab + ba + bb)^*$ na ε-NKA, ktorý akceptuje práve jazyk popísaný regulárnym výrazom.

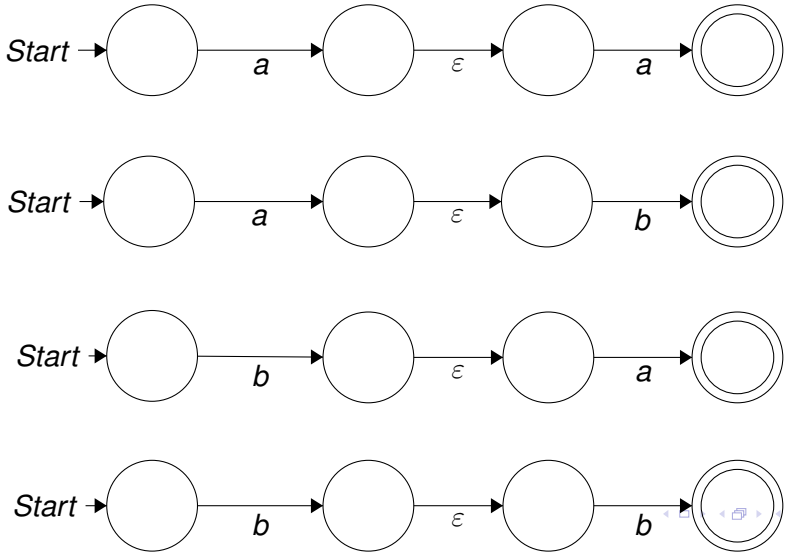
Regulárny výraz $(aa + ab + ba + bb)^*$ si rozpíšeme ako postupnosť aplikácií operátorov zjednotenie, zreťazenie, iterácia na menšie výrazy, aby sme videli, ako bude prebiehať konštrukcia výsledného ε-NKA. Tým dostaneme strom aplikácií operátorov:



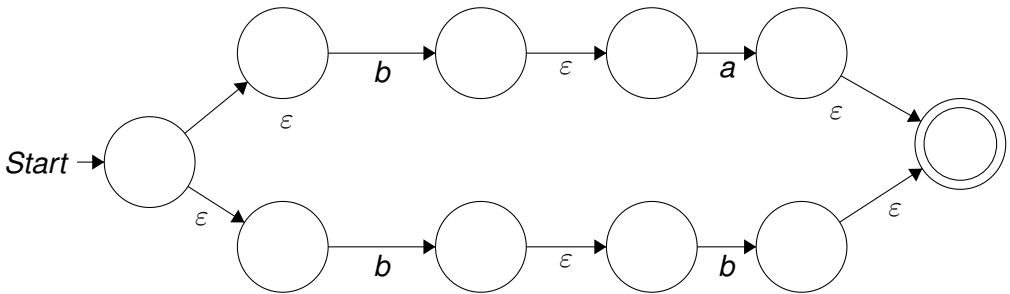
Vidíme, že základné potrebné elementy sú ε-NKA pre symboly a a b.



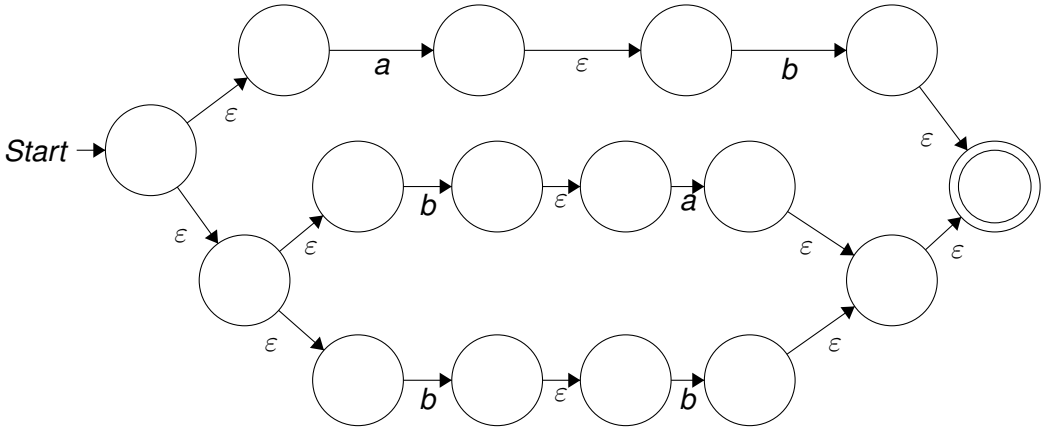
Ďalej zo stromu vidíme, že z týchto automatov potrebujeme vyrobiť automaty pre zreťazenia: *aa*, *ab*, *ba*, *bb* (v tomto poradí sú nakreslené pod sebou):



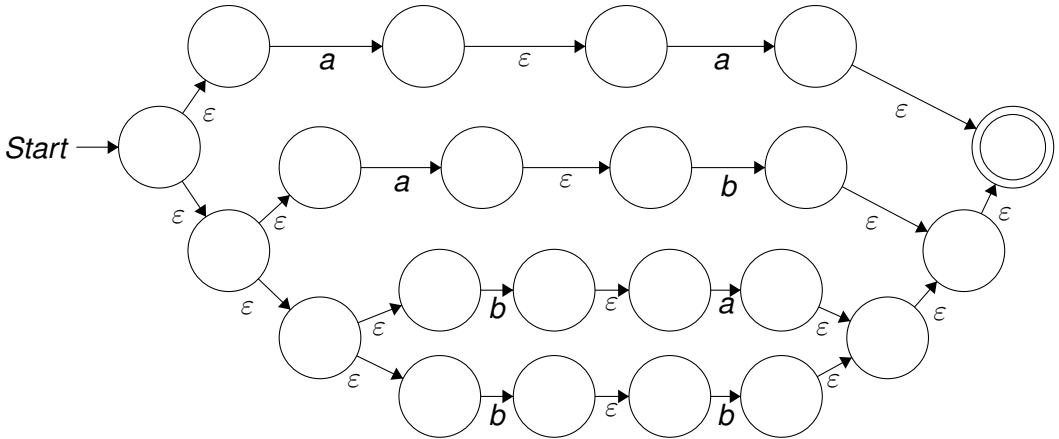
Najprv urobíme zjednotenie $ba + bb$:



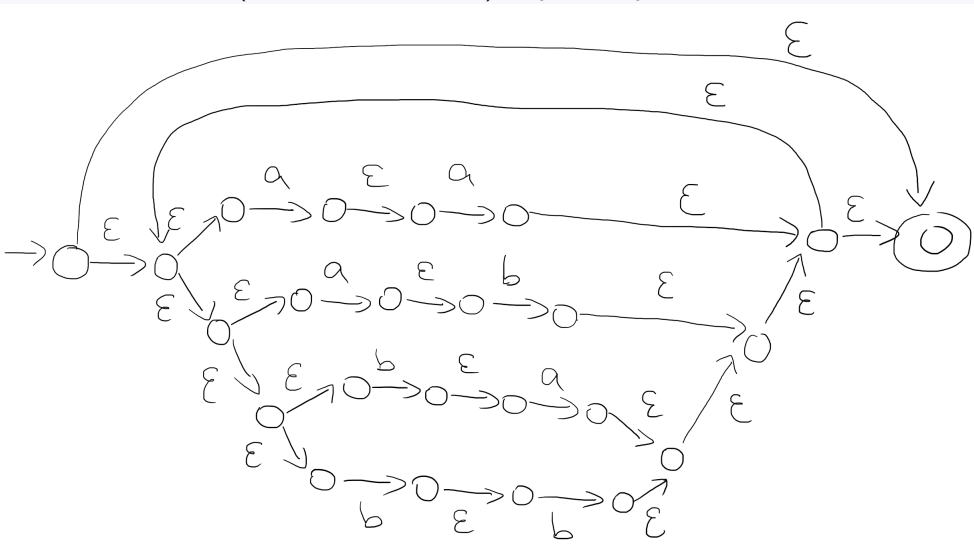
Teraz zjednotenie $ab + (ba + bb)$:



A teraz zjednotenie $aa + (ab + (ba + bb)) = aa + ab + ba + bb$:

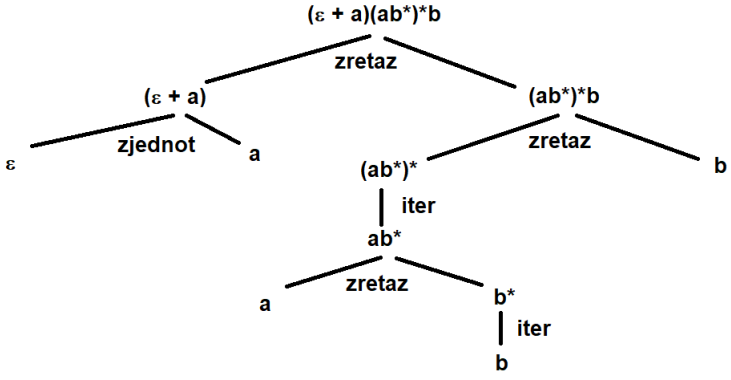


A na záver iterácia $(aa + ab + ba + bb)^*$, t.j. finálny ε-NKA:

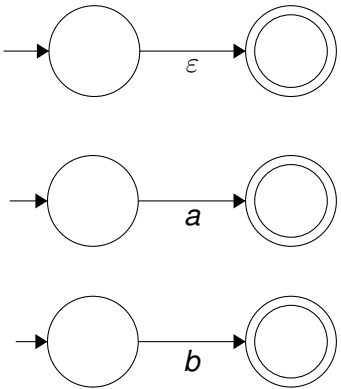


Konvertujte regulárny výraz $(\varepsilon + a)(ab^*)^*b$ na ε-NKA, ktorý akceptuje práve jazyk popísaný regulárnym výrazom.

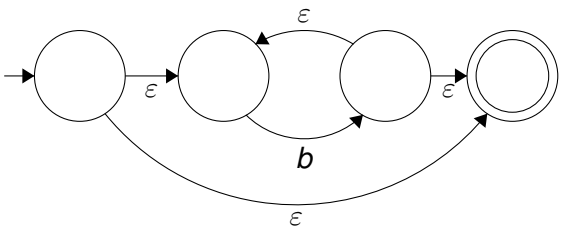
Regulárny výraz $(\varepsilon + a)(ab^*)^*b$ si rozpíšeme ako postupnosť aplikácií operátorov zjednotenie, zreťazenie, iterácia na menšie výrazy, aby sme videli, ako bude prebiehať konštrukcia výsledného ε-NKA. Tým dostaneme strom aplikácií operátorov:



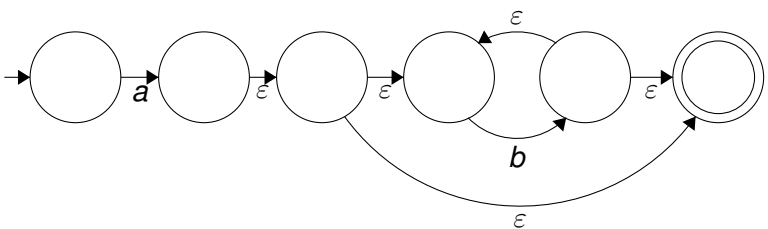
Základné výrazy v tomto regulárnom výraze sú teda (listy stromu z predchádzajúceho slajdu): ε, a, b. Príslušné ε-NKA sú:



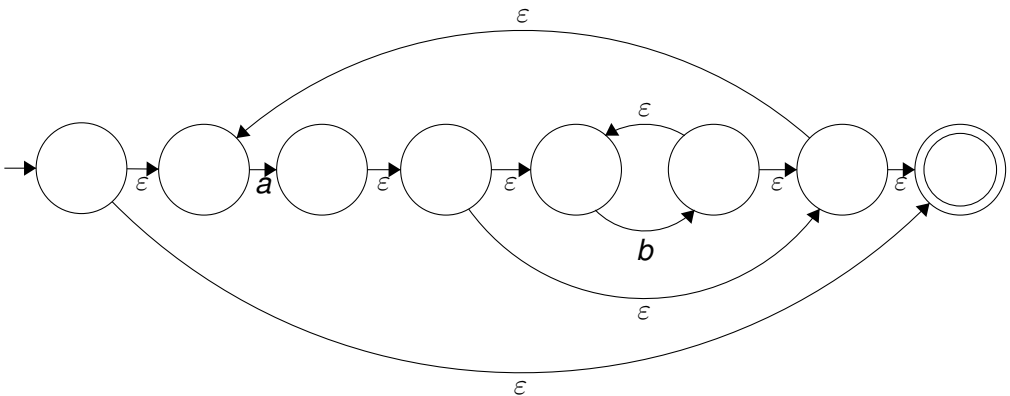
V tomto momente môžeme zostrojiť alebo ε-NKA pre $(\varepsilon + a)$, alebo pre b^*.
Zostrojme najprv ε-NKA pre b^*.



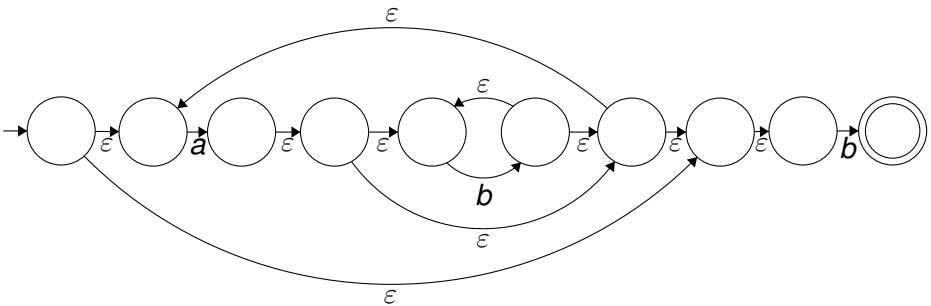
Keď máme KA pre a a b^*, môžeme zostrojiť KA pre ab^*:



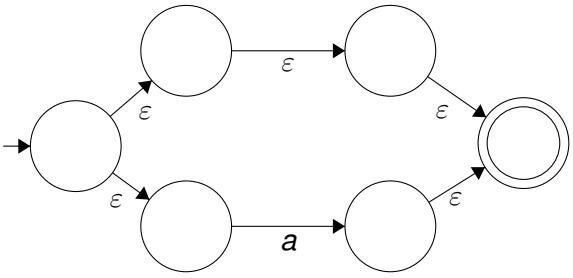
Z automatu pre ab^* zostrojíme automat pre $(ab^*)^*$:



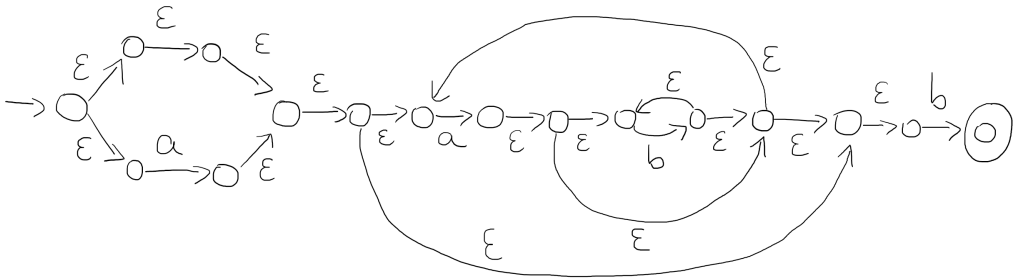
Keď máme automat pre $(ab^*)^*$ a pre b, tak zostrojíme ich zreťazenie $(ab^*)^*b$:



Z elementárnych automatov pre ε a a zostrojíme automat pre ich zjednotenie $(\varepsilon + a)$:



Z automatov pre $(\varepsilon + a)$ a $(ab^*)^*b$ zostrojíme výsledný automat pre $(\varepsilon + a)(ab^*)^*b$:



Použitá literatúra

1 Hopcroft, Motwani, Ullman - Introduction to Automata Theory, Languages and Computations, 3rd Ed.