64. Regulární množiny, regulární výrazy a rovnice nad regulárními výrazy.

Odkaz na video: SZZ: Regulární množiny, regulární výrazy

Video pokrývá témata: *Regulární množiny, regulární výrazy. Regulární množiny. Regulární výrazy. Převod regulárních výrazů na konečné automaty. Rovnice nad regulárními výrazy. Převod konečných automatů na regulární výrazy pomocí jazyků stavů. Převod konečných automatů na regulární výrazy pomocí množin přístupových řetězců.*

Regulární množiny

Ať Σ je abeceda. **Regulární množina** nad abecedou Σ je libovolná množina definovaná rekurzivně následujícím způsobem:

- $\emptyset$ je regulární množina
- {ε} je regulární množina
- ∀a ∈ Σ: {a} je regulární množina
- ať P, Q jsou regulární množiny, pak
  - P ∪ Q je regulární množina
  - P · Q je regulární množina
  - P* je regulární množina
- žádné další regulární množiny nad abecedou Σ neexistují

Regulární množina je tedy taková množina, kterou je možné získat z prázdné množiny, z množiny obsahující epsilon a z jednoprvkových množin obsahujících symboly abecedy konečnou aplikací sjednocení, konkatenace a iterace množin. V tomto případě zavádíme precedenci operátorů * > · > ∪.

Každá regulární množina popisuje nějaký regulární jazyk. Naopak, každý regulární jazyk je možné popsat nějakou regulární množinou. Třída regulárních množin odpovídá třídě regulárních jazyků $\mathcal{L}_3$.

Příkladem regulárního jazyka je jazyk

$$L = \{w \in \{a, b\}^* \mid \#_a(w) \bmod 3 = 1 \vee w \in \{b, bb\}\},$$

který je možné zapsat pomocí regulárních množin jako

$$L = \{b\}^* \cdot \{a\} \cdot \{b\}^* \cdot (\{b\}^* \cdot \{a\} \cdot \{b\}^* \cdot \{a\} \cdot \{b\}^* \cdot \{a\} \cdot \{b\}^*)^* \cup \{b\} \cup \{b\} \cdot \{b\}.$$

Regulární výrazy

Regulární výrazy jsou definované pomocí regulárních množin. Jde o zjednodušený zápis regulárních množin. Ať Σ je abeceda. **Regulární výraz** nad abecedou Σ lze rekurzivně definovat následovně:

- ∅ je regulární výraz pro regulární množinu ∅
- ε je regulární výraz pro regulární množinu {ε}
- ∀a ∈ Σ: a je regulární výraz pro regulární množinu {a}
- ať p, q jsou po řadě regulární výrazy pro regulární množiny P, Q, pak
    - p + q je regulární výraz pro regulární množinu P ∪ Q
    - pq je regulární výraz pro regulární množinu P · Q
    - p* je regulární výraz pro regulární množinu P*
- žádné další regulární výrazy nad abecedou Σ neexistují

Regulární výraz je textový řetězec odpovídající nějakému regulárnímu jazyku. Ke každému regulárnímu jazyku (i ke každé regulární množině) existuje odpovídající regulární výraz popisující stejný jazyk. Analogicky, ke každému regulárnímu výrazu existuje ekvivalentní regulární jazyk (i regulární množina). Třída regulárních výrazů odpovídá třídě regulárních jazyků $\mathcal{L}_3$.

K regulární množině

$$L = \{b\}^* \cdot \{a\} \cdot \{b\}^* \cdot (\{b\}^* \cdot \{a\} \cdot \{b\}^* \cdot \{a\} \cdot \{b\}^* \cdot \{a\} \cdot \{b\}^*)^* \cup \{b\} \cup \{b\} \cdot \{b\}$$

lze najít regulární výraz

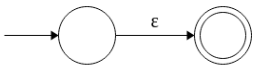
$$L = b^*ab^*(b^*ab^*ab^*ab^*)^* + b + bb.$$

Transformace regulárních výrazů na konečné automaty

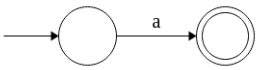
Každý regulární výraz jej možné transformovat na ekvivalentní konečný automat. Při převodu využijeme faktu, že regulární výrazy se definují rekurzivně, tedy že jsou definovány minimální regulární výrazy a způsoby, jak z nich tvořit složitější celky. Převod definujeme podobným způsobem, přičemž zajistíme, aby každý konečný automat, který bude výsledkem takového přechodu, měl právě jeden počáteční a právě jeden koncový stav. Uvažujme o abecedě Σ.

Automat pro regulární výraz ∅:

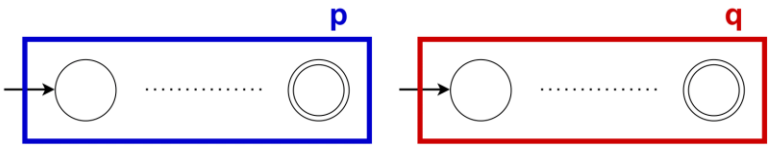
Automat pro regulární výraz ε:



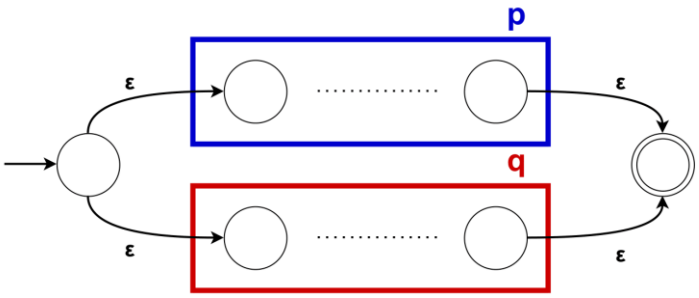
Automat pro regulární výraz a (a ∈ Σ)**:**



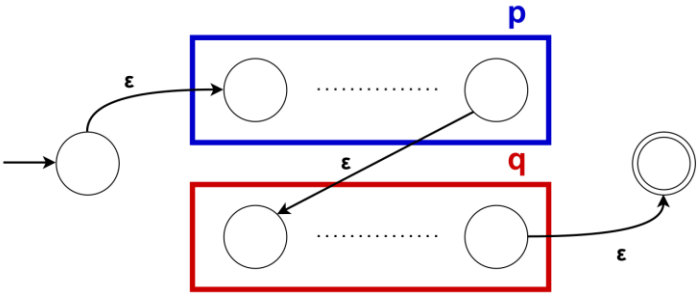
Ať p, q jsou dva regulární výrazy. Na obrázcích je naznačeno, že již máme k dispozici automaty ekvivalentní regulárním výrazům *p, q*. Jejich vnitřní struktura nás nyní nezajímá, podstatné je, že oba mají právě jeden počáteční a právě jeden koncový stav.



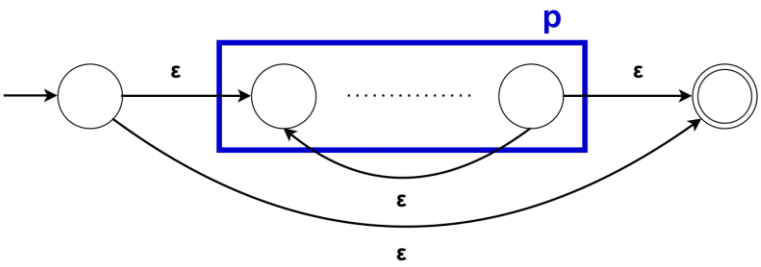
Automat pro regulární výraz p + q:
U automatů odpovídajících regulárním výrazům *p, q* zrušíme příznaky počátečních a koncových stavů. Vytvoříme nový počáteční a nový koncový stav. Z nového počátečního stavu povedeme ε-přechody do původních počátečních stavů, z původních koncových stavů povedeme ε-přechody do nového koncového stavu.



Automat pro regulární výraz pq:
U automatů odpovídajících regulárním výrazům zrušíme příznaky počátečních a koncových stavů. Vytvoříme nový počáteční a nový koncový stav. Z nového počátečního stavu povedeme ε-přechod do původního počátečního stavu pro automat odpovídající regulárnímu výrazu p. Z původního koncového stavu automatu odpovídajícího regulárnímu výrazu q povedeme ε-přechod do nového koncového stavu. Z původního koncového stavu automatu odpovídajícího regulárnímu výrazu p povedeme ε-přechod do původního počátečního stavu automatu odpovídajícího regulárnímu výrazu q.

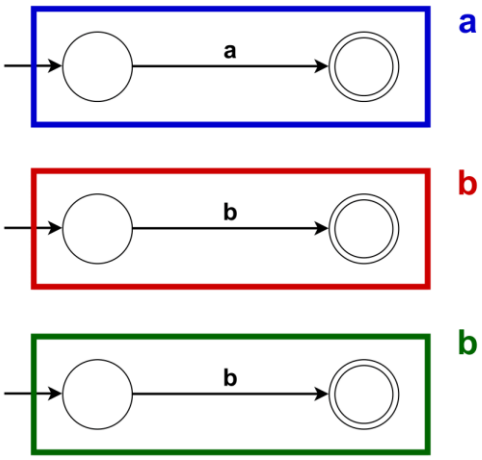


Automat pro regulární výraz p*
U automatu odpovídajícího regulárnímu výrazu p zrušíme příznak počátečního a koncového stavu. Vytvoříme nový počáteční stav, který povede přes ε-přechod do původního počátečního stavu. Rovněž vytvoříme nový koncový stav, do nějž povede ε-přechod z původního koncového stavu. Z původního koncového stavu povedeme ε-přechod do původního počátečního stavu, což odpovídá možnosti odstartovat další iteraci. Z nového počátečního stavu povedeme přechod do nového koncového stavu, což odpovídá možnosti neprovést jedinou iteraci a přijmout prázdný řetězec.

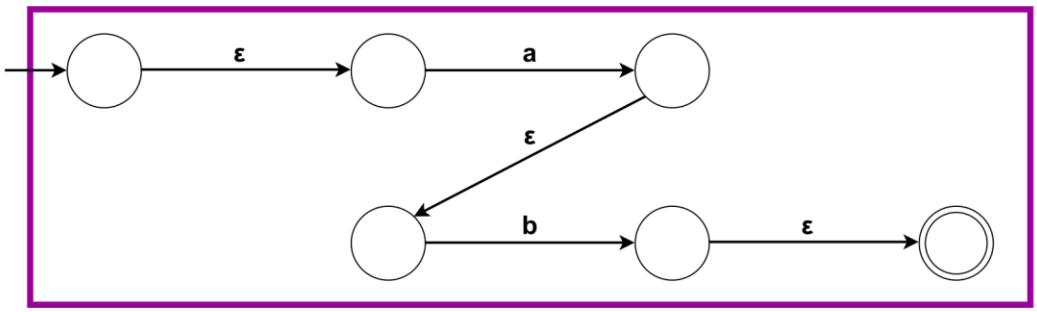


Konečnou aplikací těchto pravidel jsme schopni převést libovolný regulární výraz na konečný automat s ε-přechody. Tento automat je poté možné determinizovat a minimalizovat.

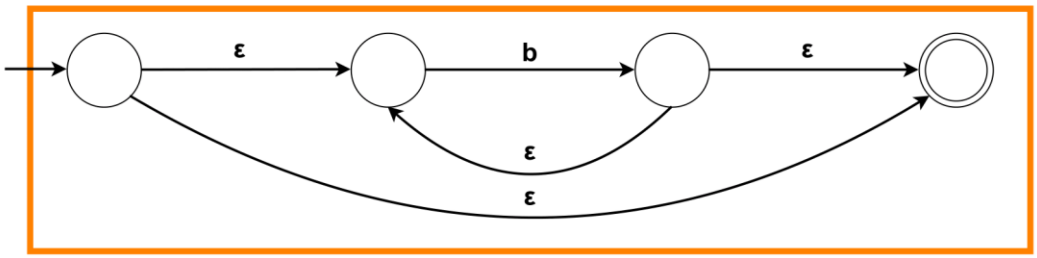
Příklad. Ukažme převod regulárního výrazu **(ab + b*)*** na konečný automat. Nejprve detekujme minimální elementy, pro něž vytvoříme odpovídající automaty. V případě regulárního výrazu **(ab + b*)*** půjde o automaty odpovídající regulárním výrazům **a**, **b**, **b**.



Uvědomme si, že precedence operátorů je následující: * > · > +. Respektujeme navíc i uzávorkování. Z automatů odpovídajících regulárním výrazům **a**, **b** vytvoříme automat odpovídající regulárnímu výrazu **ab**. Z automatu odpovídajícího regulárnímu výrazu **b** vytvoříme automat, jenž bude odpovídat regulárnímu výrazu **b***.

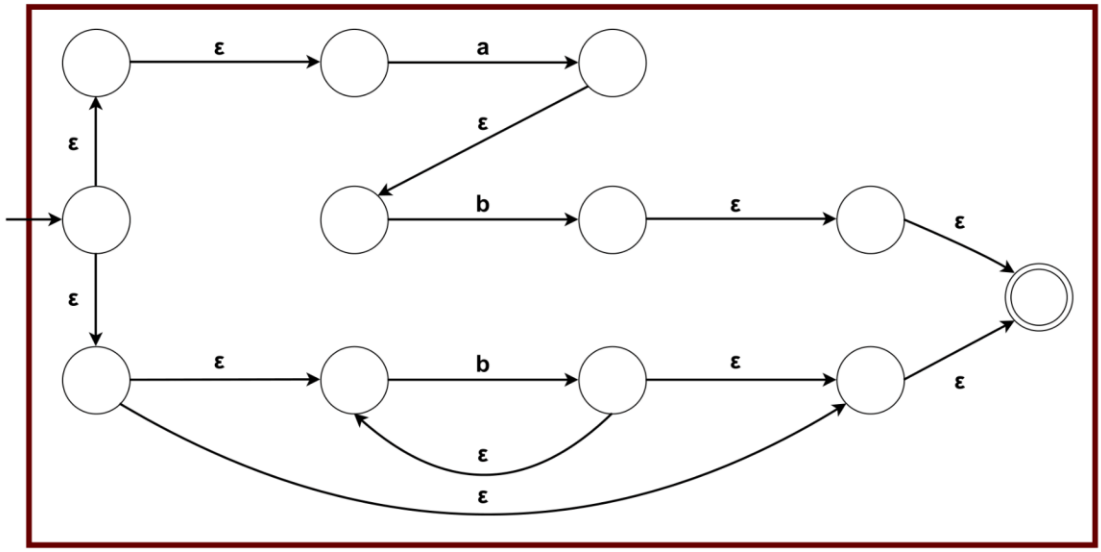


ab



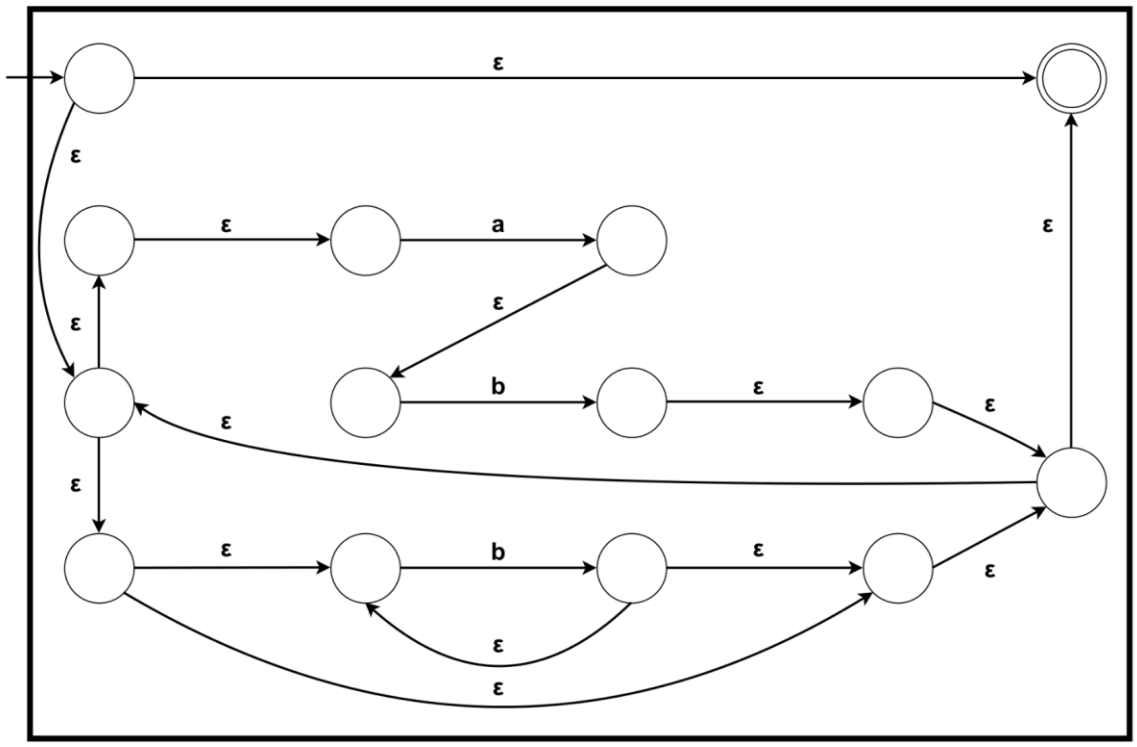
b*

Z automatů odpovídajících regulárním výrazům **ab**, **b*** nyní vytvoříme automat odpovídající regulárnímu výrazu **ab + b***.



ab + b*

Z automatu odpovídajícího regulárnímu výrazu **ab + b*** nyní vytvoříme automat odpovídající regulárnímu výrazu **(ab + b*)***.



(ab + b*)*

Tento automat samozřejmě nyní lze determinizovat a minimalizovat.

Rovnice nad regulárními výrazy

Ať Σ je abeceda a ať A = (Q, Σ, δ, s_0, F) je konečný automat, q ∈ Q.

Jazyk stavu q je množina řetězců

$$[q] = \{w \in \Sigma^* \mid \exists f \in F : (q, w) \vdash^* (f, \varepsilon)\}.$$

Jinak řečeno, jde o množinu takových řetězců, které by byly automatem přijaty, kdyby jejich čtení bylo zahájeno ve stavu q. Případně jde o jazyk, který by odpovídal automatu A, kdyby q byl počáteční stav. Zřejmě $[s_0] = L(A)$.

Množina přístupových řetězců stavu q je množina řetězců

$$L^{-1}(q) = \{w \in \Sigma^* \mid (s_0, w) \vdash^* (q, \varepsilon)\}.$$

Jinak řečeno, jde o množinu takových řetězců, které jsou automatem dočteny ve stavu q. Případně jde o jazyk, který by odpovídal automatu A, kdyby byl q jediný koncový stav. Zřejmě $\cup_{f \in F} L^{-1}(f) = L(A)$.

Rovnice nad regulárními výrazy jsou takové rovnice, kde jednotlivé koeficienty a neznámé odpovídají regulárním výrazům nad abecedou Σ. Pro spojování neznámých a koeficientů využíváme pouze operátory **+, ·, ***.

Ať Σ je abeceda. Rovnici nad regulárními výrazy nazveme **rovnicí nad regulárními výrazy ve standardním tvaru** vzhledem k neznámým {X_1, X_2, …, X_n}, pokud má následující tvar:

$$X_i = a_0 + a_1X_1 + a_2X_2 + \ldots + a_nX_n$$

kde $\{X_1, X_2, \ldots, X_n\} \cap \Sigma = \emptyset \wedge 1 <= i <= n \wedge i, n \in \mathbb{N}$ a $a_0, a_1, \ldots, a_n$ jsou regulární výrazy nad Σ. Systém takových rovnic označíme **soustavou rovnic nad regulárními výrazy ve standardním tvaru**.

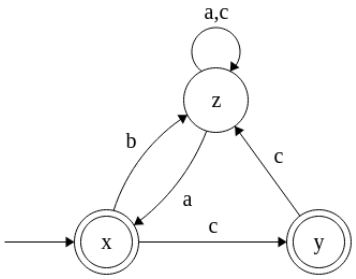
Manipulace s rovnicemi. Při řešení soustav rovnic nad regulárními výrazy je možné využívat techniky

- **dosazování**
  - dosazujeme pravé strany rovnic za odpovídající neznámé
  - máme-li například rovnice X = aY + bZ, Y = bX, lze psát Y = b(aY + bZ)
- **vytýkání a roznásobování**
  - jelikož **konkatenace je distributivní nad sjednocením**, je možné vytýkat a roznásobovat
  - je ovšem důležité si uvědomit, že **konkatenace není komutativní**
  - máme-li například rovnici X = aY + bY + cZ, lze psát X = (a + b)Y + cZ, **ale nelze napsat** X = Y(a + b) + cZ
  - máme-li například rovnici X = (a + c)Z + bZ, lze psát X = aZ + cZ + bZ, **ale nelze napsat** X = Za + Zc + bZ
- **řešení cyklické závislosti**
  - mějme rovnici X = pX + q, před dalšími manipulacemi je vhodné odstranit cyklickou závislost neznámé na sobě samé
  - jejím řešením je X = p*q
  - to se dá zcela neformálně odvodit tak, že v rovnici X = pX + q jde donekonečna dosazovat pX + q za X
  - X = pX + q = p(pX + q) + q = p(p(pX + q) + q) + q = p(p(p(pX + q) + q) + q) + q = … = p*q
- *další marginální úpravy*
  - přidávání a odebírání neutrálních prvků vzhledem ke sjednocení (∅) a konkatenaci (ε)
    - Y = (a + b)Y je totéž jako Y = (a + b)Y + ∅, zde vhodné použít před přípravou do tvaru X = pX + q kvůli absenci členu *q*
    - X = aX + bε je totéž jako X = aX + b, vhodné pro zjednodušení rovnice
  - komutativita sjednocení, pohlcení podmnožiny nadmnožinou u sjednocení
    - X = aX + bY je totéž jako X = bY + aX
    - X = c + c* je totéž jako X = c*
  - pohlcení iterované podmnožiny iterovanou nadmnožinou u konkatenace
    - X = b(a + b)*a* je totéž jako X = b(a + b)*
    - ovšem X = a*b(a + b)* není totéž jako X = b(a + b)*

Pomocí soustav rovnic nad regulárními výrazy je možné převádět konečné automaty na regulární výrazy. Neznámé budou v našem pojetí jazyky stavů jednotlivých automatů. Pro každý stav sestavíme rovnici a vhodnými manipulacemi vyjádříme neznámou odpovídající jazyku počátečního stavu.

Sestavení rovnic pro jazyky stavů automatu. Ať A = (Q, Σ, δ, s_0, F) je konečný automat, kde x, y ∈ Q jsou stavy, a ∈ Σ je symbol abecedy. Pokud y ∈ δ(x, a), pak při vytváření rovnice pro stav *x* sestrojíme neznámou *X* odpovídající jazyku daného stavu automatu, přičemž tato neznámá bude na jedné straně rovnice a na druhé straně rovnice se mimo jiné bude vyskytovat člen *aY*. Ten vyjadřuje, že do jazyku stavu *x* patří určitě každý řetězec, který začíná symbolem *a* a pokračuje libovolnou příponou odpovídající řetězci, který by patřil do jazyka stavu *y*. Díváme se tedy na příznak koncovosti a na všechny odchozí hrany.

Uvažujme o automatu vyobrazeném na schématu níže:



Automat má stavy *x, y, z*, tedy zavedeme tři neznámé *X, Y, Z*, které budou po řadě popisovat jazyky těchto stavů. Stavy *x, y* jsou koncové, tedy na pravých stranách rovnic pro tyto stavy bude muset být člen *ε*, neboť samotné *ε* patří do jazyků těchto stavů. Jelikož ze stavu *x* je možné provést přechod do stavu *y* přes symbol *c* a do stavu z přes symbol *b*, budou na pravé straně rovnice pro stav *x* i členy *cY*, *bZ*. Rovnice pro stav *x* tedy bude mít následující podobu:

$$X = cY + bZ + \varepsilon$$

Ostatní rovnice lze sestavit obdobně:

$$Y = cZ + \varepsilon$$
$$Z = aZ + cZ + aX$$

Tyto tři rovnice tvoří soustavu rovnic nad regulárními výrazy. Nyní ji vyřešme. Vytkneme ze třetí rovnice zprava Z a vyřešme cyklickou závislost neznámé Z:

$$Z = aZ + cZ + aX$$
$$Z = (a + c)Z + aX$$
$$Z = (a + c)^*aX$$

Dosaďme upravenou rovnici stavu *z* do rovnice stavu *y*:

$$Y = cZ + \varepsilon$$
$$Y = c(a + c)^*aX + \varepsilon$$

Dosaďme upravenou rovnici stavu *y* i upravenou rovnici stavu *z* do rovnice stavu *x*:

$$X = cY + bZ + \varepsilon$$
$$X = c(c(a + c)^*aX + \varepsilon) + b((a + c)^*aX) + \varepsilon$$

Rovnici roznásobme, vytkněme zprava *x* a odstraňme cyklickou závislost:

$$X = c(c(a + c)^*aX + \varepsilon) + b((a + c)^*aX) + \varepsilon$$
$$X = cc(a + c)^*aX + c + b(a + c)^*aX + \varepsilon$$

$$\mathbf{X = (cc(a + c)^*a + b(a + c)^*a)X + c + \varepsilon}$$
$$\mathbf{X = (cc(a + c)^*a + b(a + c)^*a)^*(c + \varepsilon)}$$

Výsledkem je rovnice pro jazyk počátečního stavu, tedy regulární výraz odpovídající jazyku automatu.

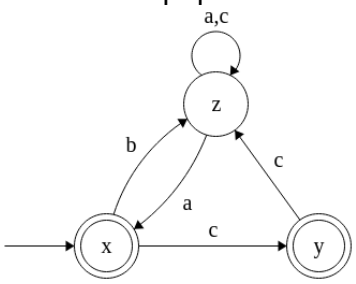
Alternativní metoda. Místo rovnic pro jazyky stavů je možné psát rovnice pro množiny přístupových řetězců stavů. Místo rovnic ve standardním tvaru

$$X_i = a_0 + a_1X_1 + a_2X_2 + \ldots + a_nX_n$$

vytváříme rovnice ve tvaru

$$X_i = a_0 + X_1a_1 + X_2a_2 + \ldots + X_na_n,$$

přičemž při sestavování rovnic dbáme na příchozí hrany a člen ε přidáme k rovnici počátečního stavu. Regulárním výrazem ekvivalentním danému automatu bude sjednocení regulárních výrazů odpovídajících koncovým stavům automatu. V případě automatu



sestavíme rovnice

$$\mathbf{X = Za + \varepsilon}$$
$$\mathbf{Y = Xc}$$
$$\mathbf{Z = Za + Zc + Yc + Xb}$$

a řešíme obdobnými manipulacemi, přičemž řešení rovnice X = Xp + q je nyní X = qp*

$$\mathbf{Z = Za + Zc + Yc + Xb}$$
$$\mathbf{Z = Z(a + c) + Yc + Xb}$$
$$\mathbf{Z = (Yc + Xb)(a + c)^*}$$
$$\mathbf{Z = (Xcc + Xb)(a + c)^*}$$
$$\mathbf{X = (Xcc + Xb)(a + c)^*a + \varepsilon}$$
$$\mathbf{X = Xcc(a + c)^*a + Xb(a + c)^*a + \varepsilon}$$
$$\mathbf{X = X(cc(a + c)^*a + b(a + c)^*a) + \varepsilon}$$
$$\mathbf{X = (cc(a + c)^*a + b(a + c)^*a)^*}$$
$$\mathbf{Y = (cc(a + c)^*a + b(a + c)^*a)^*c}$$
$$\mathbf{X + Y = (cc(a + c)^*a + b(a + c)^*a)^* + (cc(a + c)^*a + b(a + c)^*a)^*c}$$
$$\mathbf{X + Y = (cc(a + c)^*a + b(a + c)^*a)^*(\varepsilon + c)}$$

Řešením je zde sjednocení množin přístupových řetězců stavů *x, y*, neboť jde o koncové stavy.

Pokud v textu najdete chybu, nebudete něčemu rozumět nebo budete mít dojem, že by bylo vhodné něco doplnit, kontaktujte na discordu uživatele kocotom.