

# Teoretická informatika

TIN 2022/2023

**doc. RNDr. Milan Češka, Ph.D.**

`ceskam@fit.vutbr.cz`

**Upravené přednášky prof. Češky a prof. Vojnara**

# Forma kurzu a novinky

- garant (prof. Češka)
- přednášky a organizace (primárně doc. Češka)
- demo cvičení (primárně prof. Vojnar)
- numerická cvičení (prof. Vojnar, doc. Češka, doc. Holík, doc. Rogalewicz, dr. Lengál)
- demo cvičení (pátek) se nepravidelně střídají s numerickými cvičeními
- přednášky a dema se zaznamenávají a budou průběžně zveřejňovány
- hvězdičkové notace pro doplňující témata/důkazy, které se nebudou zkoušet
- 2 vnitrosemestrální testy (10+15 bodů)
- 3 domácí úkoly (3x 5 bodů, organizuje doc. Rogalewicz, dr. Lengál a doc. Holík)
- zápočet min 18 bodů z písemek a **všech** úkolů
- sbírka příkladů s ukázkou řešení + neoficiální řešení od studenta T. Kocourka
- větší důraz na praktickou složitost (amortizovaná složitost + analýza složitosti vybraných algoritmů)
- závěrečná zkouška vyžaduje minimální počty bodů z jednotlivých oblastí

# *Kde hledat informace o kurzu?*

- ❖ HLAVNÍ INFORMAČNÍ KANÁL: stránky předmětu

<https://www.fit.vutbr.cz/study/courses/TIN/public/.en>

- ❖ Domácí úlohy a diskuzní fóra jsou organizována na e-learningu VUT (moodle TIN)

- ❖ Registrace a hodnocení v předmětu ve STUDIS VUT

# Referenční literatura

❖ Opora – dostupná na stránkách předmětu.

❖ Předmět vychází zejména z následujících zdrojů:

- Češka, M.: [Teoretická informatika](#), učební text FIT VUT v Brně, 2002.  
<http://www.fit.vutbr.cz/study/courses/TIN/public/Texty/ti.pdf>
- Kozen, D.C.: [Automata and Computability](#), Springer-Verlag, New York, Inc, 1997.  
ISBN 0-387-94907-0
- Černá, I., Křetínský, M., Kučera, A.: [Automaty a formální jazyky I](#), učební text FI MU, Brno, 1999.
- Hopcroft, J.E., Motwani, R., Ullman, J.D.: [Introduction to Automata Theory, Languages, and Computation](#), Addison Wesley, 2. vydání, 2000. ISBN 0-201-44124-1
- Gruska, J.: [Foundations of Computing](#), International Thomson Computer Press, 1997. ISBN 1-85032-243-0
- Bovet, D.P., Crescenzi, P.: [Introduction to the Theory of Complexity](#), Prentice Hall Europe, Pearson Education Limited, 1994. ISBN 0-13-915380-2

# Motivace: Proč se učíme TIN?

- prohloubit znalosti základních infromatických konceptů (automaty, gramatiky, regulární výrazy) včetně algoritmů pro práci s nimi, které se přímo používají v mnohých reálných aplikacích (překladače, bezpečnost, analýza systémů atd.)
- získat základní orientaci v oblasti *vyčíslitelnosti a složitosti* (tj. jaké problémy umíme algoritmicky řešit a jaké výpočetní prostředky jsou nutné k jejich řešení)
- osvojit si schopnost *abstraktního a systematického* myšlení
- osvojit si schopnost *formálního* (tj. přesného) vyjadřování

# Reálný problém řešený v rámci jedné BP

- ❖ Vytvořte mobilní aplikaci pro následující problém plánování aktivit na dětském táboře. Je dáno  $m$  disciplín a  $n$  dětí – každé dítě má vybrané dvě oblíbené disciplíny. Vaším úkolem je najít (pokud existuje) bezkonfliktní rozdělení disciplín do 3 skupin (rund). Konflikt nastává pokud obě disciplíny vybrané jedním účastníkem přísluší stejné skupině.
- ❖ Je tento problém algoritmicky řešitelný?
- ❖ Je tento problém efektivně řešitelný?
- ❖ Jaká je časová složitost tohoto problému? Pro jak velké hodnoty  $m$  a  $n$  můžeme očekávat existenci algoritmu, který tento problém vyřeší v řádech sekund/minut.
- ❖ Dovednosti získané v TINu nám pomohou takové otázky zodpovědět a tudíž lépe pochopit daný problém

# Formulace problému

❖ Vytvořte mobilní aplikaci pro následující problém plánování aktivit na dětském táboře. Je dáno  $m$  disciplín a  $n$  dětí – každé dítě má vybrané dvě oblíbené disciplíny. Vaším úkolem je najít (pokud existuje) bezkonfliktní rozdělení disciplín do 3 skupin (rund). Konflikt nastává pokud obě disciplíny vybrané jedním účastníkem přísluší stejné skupině.

1. množina disciplín  $D = \{d_1, d_2, \dots, d_m\}$
2. výběry dětí:  $V = \{v_1, v_2, \dots, v_n\}$ , kde  $v_i = \{d_{i1}, d_{i2}\}$  pro  $d_{i1}, d_{i2} \in D$ .
3. rozdělení do skupin: funkce  $p : D \rightarrow \{1, 2, 3\}$ , kde  $\forall v_i \in V : p(d_{i1}) \neq p(d_{i2})$

❖ Je tento problém algoritmicky řešitelný? Ano, existuje pouze konečný počet funkcí  $p$  mezi dvěma konečnými množinami

❖ Je tento problém efektivně řešitelný? Naivní řešení musí vyzkoušet až  $3^m$  možných funkcí  $p$  a pro každou ověřit existenci konfliktu – exponenciální složitost.

❖ Umím to dělat lépe? Určitě ano, ale jak moc?

# Redukce na známý problém

1. množina disciplín  $D = \{d_1, d_2, \dots, d_m\}$
2. výběry dětí:  $V = \{v_1, v_2, \dots, v_n\}$ , kde  $v_i = \{d_{i1}, d_{i2}\}$  pro  $d_{i1}, d_{i2} \in D$ .
3. rozdělení do skupin: funkce  $p : D \rightarrow \{1, 2, 3\}$ , kde  $\forall v_i \in V : p(d_{i1}) \neq p(d_{i2})$

## ❖ Barvení grafů – 3 obarvitelnost

1.  $D$  je množina vrcholů
2.  $V = \{v_1, v_2, \dots, v_n\}$  je množina hran, kde  $v_i = \{d_{i1}, d_{i2}\}$  je hrana mezi vrcholy  $d_{i1}$  a  $d_{i2}$ .
3.  $p : D \rightarrow \{1, 2, 3\}$  je barvení grafu pomocí 3 barev.
4. existuje obarvení  $p$ , kde  $\forall v_i \in V : p(d_{i1}) \neq p(d_{i2})$



# Redukce na známý problém

- ❖ 3 obarvitelnost je **NP**-úplný problém: a) není znám žádný efektivní (tj. polynomiální algoritmus), b) existence takového algoritmu by znamenala zásadní průlom v computer science.
- ❖ Nemá (asi) cenu hledat obecné efektivní řešení našeho problému, ale spíše se soustředit na heuristiky, které mohou fungovat v prakticky zajímavých instancích: inspirace z řešení problému 3 obarvitelnost – branch-bound algoritmy
- ❖ Netriviální instance 3 obarvitelnosti pro  $m > 32$  už nejsou řešitelné v rozumném čase.

# Jazyky a jejich reprezentace

# Formální jazyky

❖ Prvotní pojmy: symbol, abeceda (neprázdna konečná množina symbolů).

**Definice 1.1** Necht'  $\Sigma$  je abeceda. Označme  $\Sigma^*$  množinu všech konečných posloupností  $w$  tvaru:

$$w = a_1 a_2 \dots a_n, a_i \in \Sigma \text{ pro } i = 1, \dots, n.$$

Posloupnosti  $w$  nazýváme **řetězce nad abecedou  $\Sigma$** . Dále definujeme délku  $|w|$  řetězce  $w$  jako  $|w| = n$ . Množina  $\Sigma^*$  obsahuje také speciální řetězec  $\varepsilon$ , pro který platí  $|\varepsilon| = 0$ .  $\varepsilon$  se nazývá **prázdný řetězec**.

**Definice 1.2** Množinu  $L$ , pro kterou platí  $L \subseteq \Sigma^*$  nazýváme **formálním jazykem nad abecedou  $\Sigma$** .

❖ Příklady jazyků nad abecedou  $\Sigma = \{0, 1\}$ :

- $L_1 = \{01, 0011\}$
- $L_2 = \{0^n 1^n \mid n \geq 0\}$
- $L_3 = \{ww^R \mid w \in \{0, 1\}^+\}$

❖ Na množině  $\Sigma^*$  zavedeme operaci  $\cdot$  takto:

Jsou-li dány dva řetězce  $w, w' \in \Sigma^*$ :

$$\begin{aligned}w &= a_1 a_2 \dots a_n, \\w' &= a'_1 a'_2 \dots a'_m \quad n, m \geq 0,\end{aligned}$$

pak  $w \cdot w' = a_1 a_2 \dots a_n a'_1 a'_2 \dots a'_m (= ww')$ .

Operace  $\cdot$  se nazývá **zřetězení** nebo **konkatenace**.

Pro  $w, w', w''$  platí:

1.  $|ww'| = |w| + |w'|$ ,
2.  $w(w'w'') = (ww'')w''$  tj. **asociativnost konkatenace**,
3.  $w\varepsilon = \varepsilon w = w$  tj.  $\varepsilon$  je jednotkový prvek vzhledem k operaci  $\cdot$ .

❖ Dále zavádíme pojmy:

- prefix, sufix, podřetězec,
- $a^i = a_1 a_2 \dots a_i$  kde  $\forall 1 \leq j \leq i : a_j = a$
- $\#_a(w)$  je počet znaků  $a \in \Sigma$  v řetězci  $w \in \Sigma^*$
- $w^R$  (tj. revers řetězce): pro  $w = a_1 a_2 \dots a_n$  je  $w^R = a_n a_{n-1} \dots a_1$

# Operace nad jazyky

Jazyk je množina  $\rightarrow$  jsou definovány všechny množinové operace nad jazyky.  
Připomeňme operaci komplement jazyka  $L$  nad abecedou  $\Sigma$  (často značený jako  $co-L$ ), který je definován jako  $co-L = \Sigma^* \setminus L$ .

**Definice 1.3** Nechť  $L_1$  je jazyk nad abecedou  $\Sigma_1$ ,  $L_2$  je jazyk nad abecedou  $\Sigma_2$ .  
**Součinem (konkatenací)** jazyků  $L_1$  a  $L_2$  nad abecedou  $\Sigma_1 \cup \Sigma_2$  rozumíme jazyk

$$L_1 \cdot L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$$

**Příklad 1.1** Nechť  $P = \{A, B, \dots, Z, a, b, \dots, z\}$ ,  $C = \{0, 1, \dots, 9\}$  jsou abecedy,  
 $L_1 = P$  a  $L_2 = (P \cup C)^*$  jazyky nad  $P$  resp.  $P \cup C$ .  
Jaký jazyk určuje součin  $L_1 L_2$ ?

# Iterace a pozitivní iterace

**Definice 1.4** Necht'  $L$  je jazyk. **Iterací**  $L^*$  jazyka  $L$  a **pozitivní iterací**  $L^+$  jazyka  $L$  definujeme takto:

1.  $L^0 = \{\varepsilon\}$
2.  $L^n = L \cdot L^{n-1}$  pro  $n \geq 1$
3.  $L^* = \bigcup_{n \geq 0} L^n$
4.  $L^+ = \bigcup_{n \geq 1} L^n$

Je-li  $L$  jazyk, pak platí:

1.  $L^* = L^+ \cup \{\varepsilon\}$
2.  $L^+ = L \cdot L^* = L^* \cdot L$

**Příklad 1.2**  $L_1 = \{(p)\}$ ,  $L_2 = \{, p\}$ ,  $L_3 = \{\}$   
 $L_1 L_2^* L_3 = \{(p), (p, p), (p, p, p), \dots\}$

# \*Algebra jazyků\*

**Definice 1.5** Algebraická struktura  $\langle A, +, \cdot, 0, 1 \rangle$  se nazývá **polookruh**, jestliže:

1.  $\langle A, +, 0 \rangle$  je komutativní monoid (tj.  $+$  je asociativní a  $0$  je neutrální prvek) ,
2.  $\langle A, \cdot, 1 \rangle$  je monoid,
3. pro operaci  $\cdot$  platí distributivní zákon vzhledem k  $+$ :  
 $\forall a, b, c \in A : a(b + c) = ab + ac, (a + b)c = ac + bc.$

**Věta 1.1** Algebra jazyků  $\langle 2^{\Sigma^*}, \cup, \cdot, \emptyset, \{\varepsilon\} \rangle$ , kde  $\cup$  je sjednocení a  $\cdot$  konkatenace jazyků tvoří polookruh.

*Důkaz.*

1.  $\langle 2^{\Sigma^*}, \cup, \emptyset \rangle$  je komutativní monoid ( $\cup$  je komutativní a asociativní operace a  $L \cup \emptyset = \emptyset \cup L = L$  pro všechna  $L \in 2^{\Sigma^*}$ ).
2.  $\langle 2^{\Sigma^*}, \cdot, \{\varepsilon\} \rangle$  je monoid:  
 $L \cdot \{\varepsilon\} = \{\varepsilon\} \cdot L = L$  pro všechna  $L \in 2^{\Sigma^*}$ .
3. Pro všechny  $L_1, L_2, L_3 \in 2^{\Sigma^*}$  :  
 $L_1(L_2 \cup L_3) = \{xy \mid (x \in L_1) \wedge (y \in L_2 \vee y \in L_3)\} = \dots = L_1L_2 \cup L_1L_3$

# Důkazy identity jazyků

## Příklad:

Rozhodněte a dokažte, zda platí následující tvrzení:

1.  $\forall L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$
2.  $\exists L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$
3.  $\forall L \subseteq \Sigma^* : L^* = (L^*)^*$



# Důkazy identity jazyků

## Příklad:

Rozhodněte a dokažte, zda platí následující tvrzení:

1.  $\forall L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$
2.  $\exists L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$
3.  $\forall L \subseteq \Sigma^* : L^* = (L^*)^*$

Řešení 1: Tvrzení neplatí to jest platí jeho negace  $\exists L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* \neq (L_1 \cup L_2)^*$   
Zvolme například  $L_1 = \{a\}$  a  $L_2 = \{b\}$ . Pak  $L_1^* \cup L_2^* = \{a\}^* \cup \{b\}^*$  ale  
 $(L_1 \cup L_2)^* = \{a, b\}^*$ . Zřejmě  $\{a\}^* \cup \{b\}^* \neq \{a, b\}^*$ .

# Důkazy identity jazyků

## Příklad:

Rozhodněte a dokažte, zda platí následující tvrzení:

1.  $\forall L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$
2.  $\exists L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$
3.  $\forall L \subseteq \Sigma^* : L^* = (L^*)^*$

Řešení 1: Tvrzení neplatí to jest platí jeho negace  $\exists L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* \neq (L_1 \cup L_2)^*$   
Zvolme například  $L_1 = \{a\}$  a  $L_2 = \{b\}$ . Pak  $L_1^* \cup L_2^* = \{a\}^* \cup \{b\}^*$  ale  $(L_1 \cup L_2)^* = \{a, b\}^*$ . Zřejmě  $\{a\}^* \cup \{b\}^* \neq \{a, b\}^*$ .

Řešení 2: Tvrzení platí. Zvolme například  $L_1 = L_2 = \{a\}$ . Pak  $L_1^* \cup L_2^* = \{a\}^* = (L_1 \cup L_2)^*$ .

# Důkazy identity jazyků

## Příklad:

Rozhodněte a dokažte, zda platí následující tvrzení:

1.  $\forall L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$
2.  $\exists L_1, L_2 \subseteq \Sigma^* : L_1^* \cup L_2^* = (L_1 \cup L_2)^*$
3.  $\forall L \subseteq \Sigma^* : L^* = (L^*)^*$

Řešení 3: Tvrzení platí. Notace:  $w_0$  i  $w_i^0$  (pro  $i \geq 1$ ) označuje  $\epsilon$

$$L^* = \bigcup_{n \geq 0} L^n = \{w \mid w \in L^i \wedge i \geq 0\} = \{w_1 w_2 \dots w_n \mid n \geq 0 \wedge \forall 1 \leq i \leq n : w_i \in L\}$$

$$\begin{aligned} (L^*)^* &= \{w_1 w_2 \dots w_n \mid n \geq 0 \wedge \forall 1 \leq i \leq n : w_i \in L^{m_i} \wedge m_i \geq 0\} = \\ &= \{w_1^1 w_1^2 \dots w_1^{m_1} w_2^1 w_2^2 \dots w_2^{m_2} \dots w_n^1 w_n^2 \dots w_n^{m_n} \mid n \geq 0 \wedge \\ &\quad \forall 1 \leq i \leq n : m_i \geq 0 \wedge \forall 1 \leq j \leq \max\{m_i \mid 1 \leq i \leq n\} : w_i^j \in L\} = \\ &= \{w_1 w_2 \dots w_k \mid k = m_1 + m_2 + \dots + m_n \wedge \forall 1 \leq i \leq k : w_i \in L\} = \\ &= \{w_1 w_2 \dots w_k \mid k \geq 0 \wedge \forall 1 \leq i \leq k : w_i \in L\} = L^* \end{aligned}$$

# Gramatiky

❖ Pozn. Reprezentace jazyků – problém reprezentace, způsoby reprezentace.

**Definice 1.6** Gramatika  $G$  je čtveřice  $G = (N, \Sigma, P, S)$ , kde

1.  $N$  je konečná množina **nonterminálních symbolů**.
2.  $\Sigma$  je abeceda (tj. konečná množina **terminálních symbolů**), kde  $N \cap \Sigma = \emptyset$ .
3.  $P$  je konečná podmnožina kartézského součinu

$$(N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$$

nazývaná **množina přepisovacích pravidel**

4.  $S \in N$  je **výchozí** (startovací) symbol gramatiky  $G$ .

❖ Prvek  $(\alpha, \beta) \in P$  je **přepisovací pravidlo** a zapisuje se ve tvaru

$$\alpha \rightarrow \beta,$$

kde  $\alpha$  je **levá strana**,  $\beta$  je **pravá strana** pravidla  $\alpha \rightarrow \beta$ .



❖ **Konvence 1:** Obsahuje-li množina  $P$  přepisovací pravidla tvaru

$$\alpha \rightarrow \beta_1, \alpha \rightarrow \beta_2, \dots, \alpha \rightarrow \beta_n$$

pak pro zkrácení budeme používat zápisu

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \dots \mid \beta_n$$

❖ **Konvence 2:** Pro zápis symbolů a řetězců budeme užívat této úmluvy:

1.  $a, b, c, d$  reprezentují terminální symboly.
2.  $A, B, C, D, S$  reprezentují nonterminální symboly,  $S$  výchozí symbol.
3.  $U, V, \dots, Z$  reprezentují terminální nebo nonterminální symboly.
4.  $\alpha, \beta, \dots, \omega$  reprezentují řetězce z množiny  $(N \cup \Sigma)^*$ .
5.  $u, v, w, \dots, z$  reprezentují řetězce z  $\Sigma^*$ .

**Definice 1.7** Necht'  $G = (N, \Sigma, P, S)$  je gramatika a necht'  $\lambda, \mu$  jsou řetězce z  $(N \cup \Sigma)^*$ . Mezi  $\lambda$  a  $\mu$  platí binární relace  $\xRightarrow{G}$ , zvaná **přímá derivace**, můžeme-li řetězce  $\lambda$  a  $\mu$  vyjádřit ve tvaru

$$\begin{aligned}\lambda &= \gamma\alpha\delta \\ \mu &= \gamma\beta\delta\end{aligned}$$

$\gamma, \delta \in (N \cup \Sigma)^*$  a  $\alpha \rightarrow \beta$  je nějaké přepisovací pravidlo z  $P$ . Pak píšeme

$$\begin{aligned}\lambda &\xRightarrow{G} \mu \text{ nebo} \\ \lambda &\Rightarrow \mu.\end{aligned}$$

**Definice 1.8** Necht'  $G = (N, \Sigma, P, S)$  je gramatika a  $\Rightarrow$  relace přímé derivace na  $(N \cup \Sigma)^*$ . Relace  $\xRightarrow{+}$  označuje tranzitivní uzávěr relace  $\Rightarrow$  a nazývá se **relací derivace**. Platí-li  $\lambda \xRightarrow{+} \mu$ , pak existuje posloupnost

$$\lambda = \nu_0 \Rightarrow \nu_1 \Rightarrow \dots \Rightarrow \nu_n = \mu, \quad n \geq 1,$$

která se nazývá **derivací délky  $n$** .

❖ Relace  $\xRightarrow{*}$  označuje reflexivní a tranzitivní uzávěr relace  $\Rightarrow$ :

$$\lambda \xRightarrow{*} \mu \quad \Rightarrow \quad \lambda \xRightarrow{+} \mu \text{ nebo } \lambda = \mu$$

**Příklad 1.3** Derivace v gramatice  $G_1$ , resp.  $G_2$ , ze strany 11:

❖ V gramatice  $G_1$ :

- Pravidlo  $0A \rightarrow 00A1$  implikuje  $0^n A 1^n \Rightarrow 0^{n+1} A 1^{n+1}$ ,
- tedy  $0A1 \xRightarrow{*} 0^n A 1^n$  pro libovolné  $n > 0$ .

❖ V gramatice  $G_2$ :

- $I \Rightarrow IP \Rightarrow IPP \Rightarrow ICPP \Rightarrow PCPP \Rightarrow aCPP \Rightarrow a1PP \Rightarrow a1xP \Rightarrow a1xy$ ,
- tj.  $I \xRightarrow{+} a1xy$ .



**Definice 1.9** Necht'  $G = (N, \Sigma, P, S)$  je gramatika. Řetězec  $\alpha \in (N \cup \Sigma)^*$  nazýváme **větnou formou**, platí-li  $S \xRightarrow{*} \alpha$ . Větná forma, která obsahuje pouze terminální symboly se nazývá **věta**.

Jazyk  $L(G)$  generovaný gramatikou  $G$  je množina:

$$L(G) = \{w \in \Sigma^* \mid S \xRightarrow{*} w\}$$

### Příklad 1.4

$$L(G_1) = \{0^n 1^n \mid n > 0\}$$

protože

$$S \Rightarrow 0A1$$

$$S \xRightarrow{*} 0^n A 1^n \quad (\text{viz předchozí příklad})$$

$$S \xRightarrow{*} 0^n 1^n \quad (\text{pravidlo } A \rightarrow \varepsilon)$$

# Chomského hierarchie

Je definována na základě tvaru přepisovacích pravidel:

- Typ 0 – obecné (neomezené) gramatiky:

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

- Typ 1 – kontextové gramatiky:

$$\alpha A \beta \rightarrow \alpha \gamma \beta \quad A \in N, \alpha, \beta \in (N \cup \Sigma)^*, \gamma \in (N \cup \Sigma)^+$$

nebo  $S \rightarrow \varepsilon$ , pakliže se  $S$  neobjevuje na pravé straně žádného pravidla

(Alternativní definice definující stejnou třídu jazyků:  $\alpha \rightarrow \beta$ ,  $|\alpha| \leq |\beta|$  nebo  $S \rightarrow \varepsilon$  omezené jako výše.)

- Typ 2 – bezkontextové gramatiky:

$$A \rightarrow \alpha \quad A \in N, \alpha \in (N \cup \Sigma)^*$$

- Typ 3 – regulární gramatiky:

$$A \rightarrow xB \quad \text{nebo}$$
$$A \rightarrow x \mid \varepsilon \quad A, B \in N, x \in \Sigma$$

(Alternativní tvary gramatik, které mají stejnou vyjadřovací sílu, jsou uvedeny v opoře.) Regulární jazyky 1 – p.26/69

**Definice 1.10** Jazyk generovaný gram. typu  $i \in \{0, 1, 2, 3\}$ , se nazývá **jazykem typu  $i$** .

Existuje synonymní označení jazyků:

- Jazyk typu 0 — **rekurzivně vyčíslitelný jazyk**.
- Jazyk typu 1 — **kontextový jazyk**.
- Jazyk typu 2 — **bezkontextový jazyk**.
- Jazyk typu 3 — **regulární jazyk**.

**Věta 1.2** Nechť  $\mathcal{L}_i$  značí třídu všech jazyků typu  $i$  nad abecedou  $\Sigma$ .

Pak platí:

$$\mathcal{L}_3 \subseteq \mathcal{L}_2 \subseteq \mathcal{L}_1 \subseteq \mathcal{L}_0$$

*Důkaz.*

Důkaz plyne z definice tříd Chomského hierarchie jazyků.

□

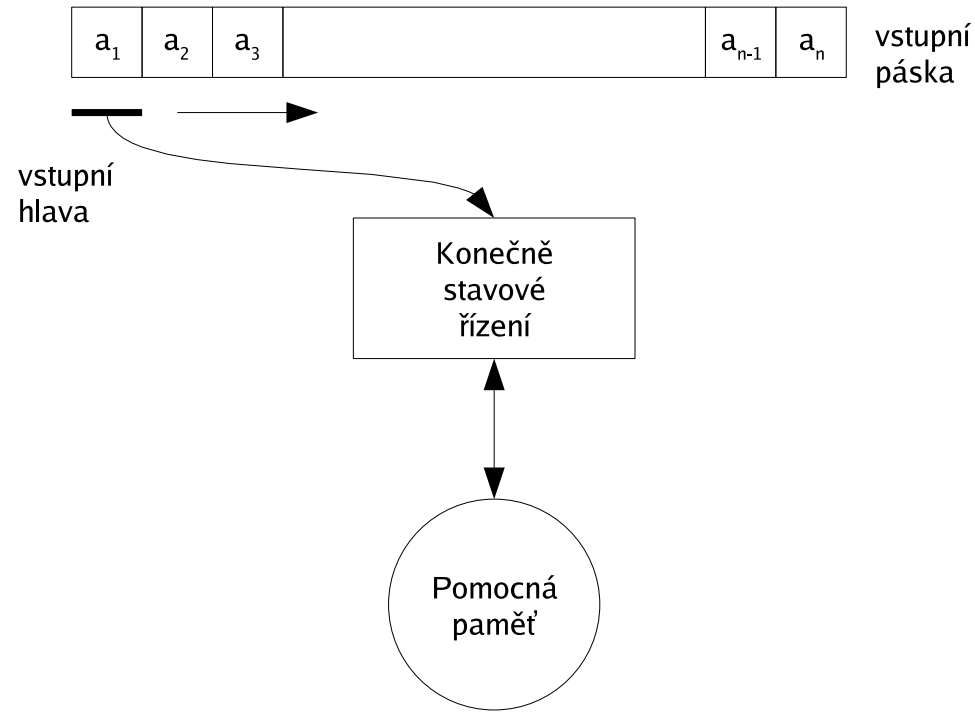
**Věta 1.3** Platí:

$$\mathcal{L}_3 \subset \mathcal{L}_2 \subset \mathcal{L}_1 \subset \mathcal{L}_0$$

*Důkaz jednotlivých inkluzí v průběhu semestru.*

# Regulární jazyky a Konečné automaty

# Automaty



## ❖ Klasifikace:

- podle mechanismu a funkce čtecí hlavy,
- pomocné paměti,

# Konečný automat

**Definice 1.11** Konečným nedeterministický automatem (NKA) rozumíme automat  $M$  specifikovaný 5-ticí

$$M = (Q, \Sigma, \delta, q_0, F), \quad \text{kde:}$$

1.  $Q$  je konečná množina stavů,
2.  $\Sigma$  je konečná vstupní abeceda,
3.  $\delta$  je funkce přechodů (přechodová funkce) tvaru  $\delta : Q \times \Sigma \rightarrow 2^Q$ ,
4.  $q_0 \in Q$  je počáteční stav,
5.  $F \subseteq Q$  je množina koncových stavů.

Je-li  $\forall q \in Q \forall a \in \Sigma : |\delta(q, a)| \leq 1$ , pak  $M$  nazýváme deterministickým konečným automatem (zkráceně DKA).

**Deterministický konečný automat** je také často specifikován 5-ticí

$$M = (Q, \Sigma, \delta, q_0, F), \quad \text{kde:}$$

- $\delta$  je parciální funkce tvaru  $\delta : Q \times \Sigma \rightarrow Q$ ,
- a význam ostatních složek zůstává zachován.

Je-li přechodová funkce  $\delta$  totální, pak  $M$  nazýváme **úplně definovaným deterministickým konečným automatem**.

Dále budeme obvykle pracovat s touto specifikací DKA.

**Lemma 1.1** Ke každému DKA  $M$  existuje „ekvivalentní“ úplně definovaný DKA  $M'$ .

*Důkaz.* (idea) Množinu stavů automatu  $M'$  rozšíříme o nový, nekonečný stav (anglicky označovaný jako *SINK* stav) a s využitím tohoto stavu doplníme prvky přechodové funkce  $\delta'$  automatu  $M'$  tak, aby byla totální. □



## Příklad 1.5 Zápís NKA.

$$\text{NKA } M_1 = (\{q_0, q_1, q_2, q_F\}, \{0, 1\}, \delta, q_0, \{q_F\})$$

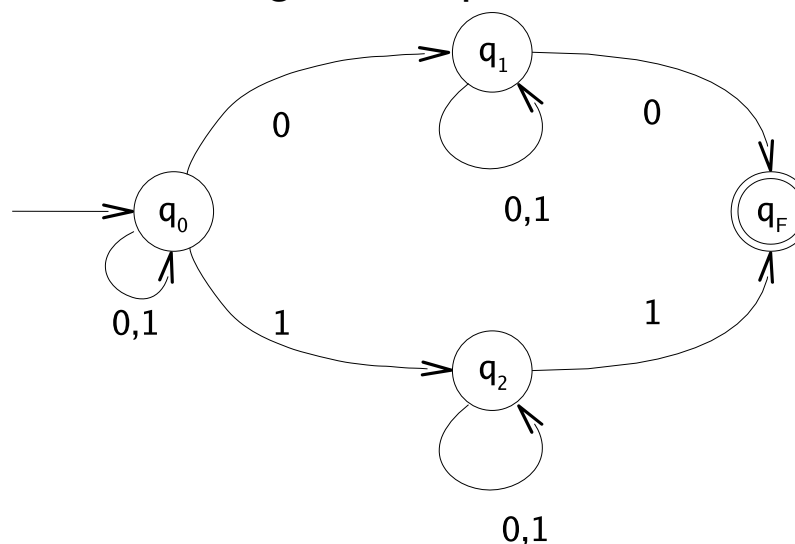
$$\begin{aligned} \delta : \quad & \delta(q_0, 0) = \{q_0, q_1\} & \delta(q_0, 1) &= \{q_0, q_2\} \\ & \delta(q_1, 0) = \{q_1, q_F\} & \delta(q_1, 1) &= \{q_1\} \\ & \delta(q_2, 0) = \{q_2\} & \delta(q_2, 1) &= \{q_2, q_F\} \\ & \delta(q_F, 0) = \emptyset & \delta(q_F, 1) &= \emptyset \end{aligned}$$

❖ Alternativní způsoby reprezentace funkce  $\delta$ :

1. maticí (přechodů)

	0	1
$q_0$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$q_1$	$\{q_1, q_F\}$	$\{q_1\}$
$q_2$	$\{q_2\}$	$\{q_2, q_F\}$
$q_F$	$\emptyset$	$\emptyset$

2. diagramem přechodů



**Definice 1.12** Necht'  $M = (Q, \Sigma, \delta, q_0, F)$  je konečný automat (tj. NKA).

❖ **Konfigurace**  $C$  konečného automatu  $M$  je uspořádaná dvojice

$$C = (q, w), \quad (q, w) \in Q \times \Sigma^*$$

kde  $q$  je aktuální stav,  $w$  je dosud nezpracovaná část vstupního řetězce.

❖ **Počáteční konfigurace** je konfigurace  $(q_0, a_1 a_2 \dots a_n)$ .

❖ **Koncová konfigurace** je konfigurace  $(q_F, \varepsilon)$ ,  $q_F \in F$ .

❖ **Přechodem automatu**  $M$  rozumíme binární relaci  $\vdash_M$  v množině konfigurací  $C$

$$\vdash_M \subseteq (Q \times \Sigma^*) \times (Q \times \Sigma^*)$$

která je definována takto:

$$(q, w) \vdash_M (q', w') \iff^{def.} w = aw' \wedge q' \in \delta(q, a) \text{ pro } q, q' \in Q, a \in \Sigma, w, w' \in \Sigma^*$$

Relace  $\vdash_M^k$ ,  $\vdash_M^+$ ,  $\vdash_M^*$  mají obvyklý význam, tj.  $k$ -tá mocnina relace  $\vdash$ , tranzitivní a tranzitivní a reflexivní uzávěr relace  $\vdash$ .

❖ Řetězec  $w$  přijímaný NKA  $M$  je definován takto:  $(q_0, w) \stackrel{*}{\vdash}_M (q, \varepsilon), \quad q \in F$ .

❖ Jazyk  $L(M)$  přijímaný NKA  $M$  je definován takto:

$$L(M) = \{w \mid (q_0, w) \stackrel{*}{\vdash}_M (q, \varepsilon) \wedge q \in F\}.$$

**Příklad 1.6** Uvažujme NKA  $M_1$  z příkladu 1.5. Platí:

$$(q_0, 1010) \vdash (q_0, 010) \vdash (q_1, 10) \vdash (q_1, 0) \vdash (q_f, \varepsilon)$$

a tedy:  $(q_0, 1010) \stackrel{*}{\vdash} (q_f, \varepsilon)$

Neplatí například  $(q_0, \varepsilon) \stackrel{*}{\vdash} (q_f, \varepsilon)$

Vyjádření jazyka  $L(M_1)$ :

$$L(M_1) = \{w \mid w \in \{0, 1\}^* \wedge w \text{ končí symbolem, který je již v řetězci } w \text{ obsažen}\}$$

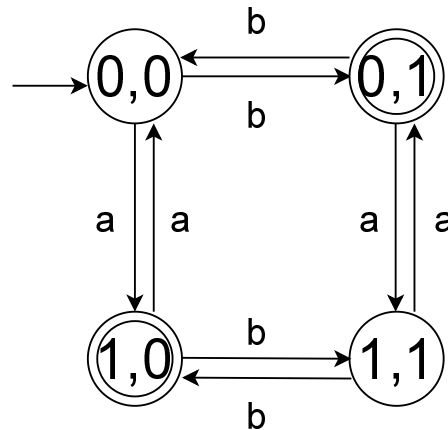
□

# Konstrukce konečných automatů

❖ **Interpretace stavů:** Stav reprezentuje informaci o průběhu výpočtu

**Příklad:**  $L = \{w \in \{a, b\}^* \mid \#_a(w) \bmod 2 \neq \#_b(w) \bmod 2\}$

Stav kóduje dvojici  $(p, q)$  kde  $p = \#_a(u) \bmod 2$  a  $q = \#_b(u) \bmod 2$  a  $u$  je zatím přečtený vstup.



# Konstrukce konečných automatů

❖ **Interpretace stavů:** Stav reprezentuje informaci o průběhu výpočtu

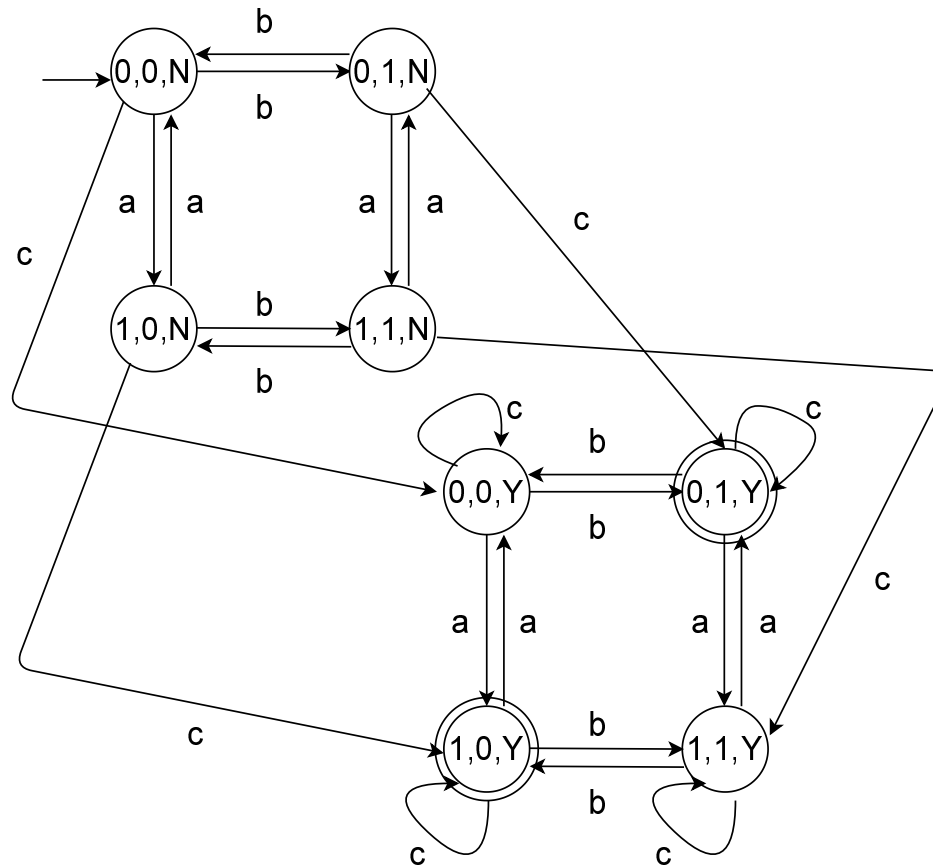
$$L = \{w \in \{a, b, c\}^* \mid \#_a(w) \bmod 2 \neq \#_b(w) \bmod 2 \wedge \#_c(w) > 0\}$$

# Konstrukce konečných automatů

❖ Interpretace stavů: Stav reprezentuje informaci o průběhu výpočtu

$$L = \{w \in \{a, b, c\}^* \mid \#_a(w) \bmod 2 \neq \#_b(w) \bmod 2 \wedge \#_c(w) > 0\}$$

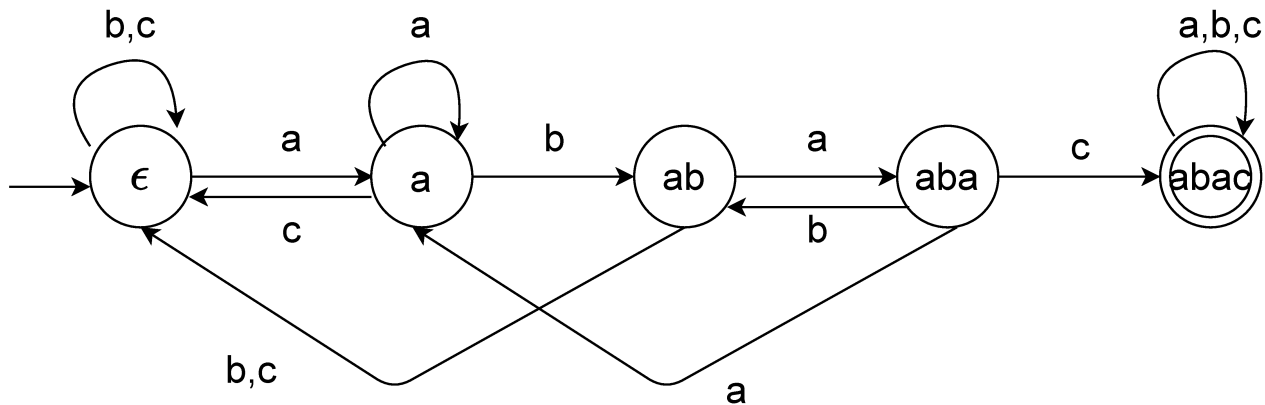
Stav kóduje trojici  $(p, q, r)$  kde  $p = \#_a(u) \bmod 2$  a  $q = \#_b(u) \bmod 2$  a  $r = N$  pokud  $\#_c(u) = 0$  a  $r = Y$  pokud  $\#_c(u) > 0$



# Konstrukce konečných automatů

❖ **Využití nedeterminismu:** Stroj "uhádne" jistý aspekt výpočtu vedoucí k přijetí slova z jazyka.

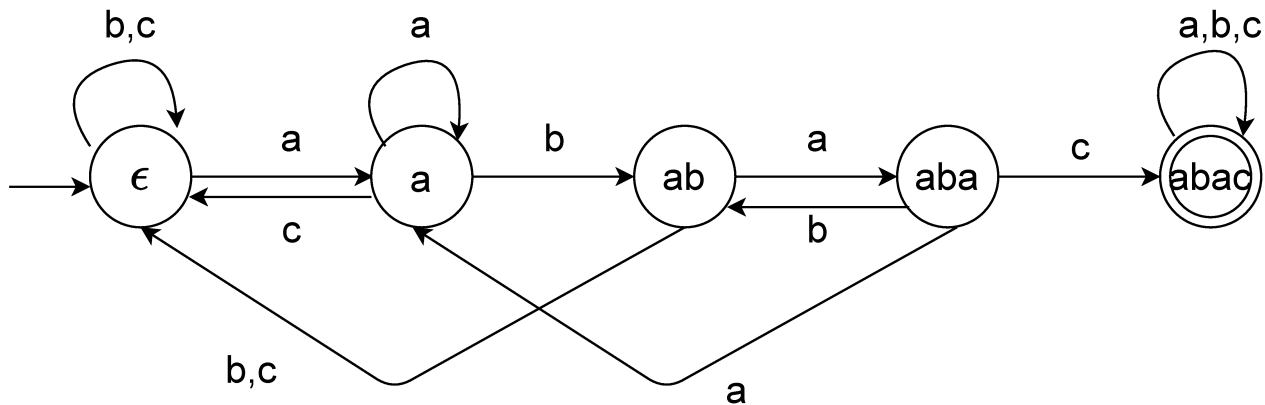
$$L = \{w \in \{a, b, c\}^* \mid w \text{ obsahuje podslovo } abac\}$$



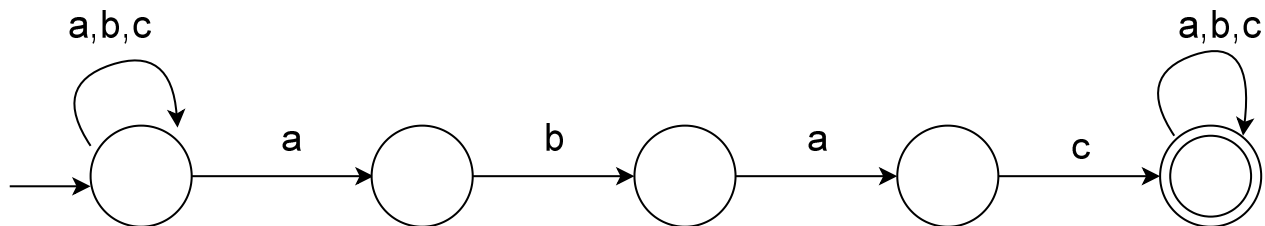
# Konstrukce konečných automatů

❖ **Využití nedeterminismu:** Stroj "uhádne" jistý aspekt výpočtu vedoucí k přijetí slova z jazyka.

$$L = \{w \in \{a, b, c\}^* \mid w \text{ obsahuje podslovo } abac\}$$



❖ **NKA "uhádne"** kde začíná hledané podslovo a pouze ověří jeho tvar.





# Ekvivalence NKA a DKA

**Věta 1.4** Každý NKA  $M$  lze převést na DKA  $M'$  tak, že  
$$L(M) = L(M').$$

*Důkaz.*

1. Nalezneme algoritmus převodu  $M \rightarrow M'$  (níže).
2. Ukážeme, že  $L(M) = L(M')$  tj. ukážeme, že platí:
  - (a)  $L(M) \subseteq L(M')$  a současně,
  - (b)  $L(M') \subseteq L(M)$ .

□

# Převod NKA na ekvivalentní DKA

## Algoritmus 1.1

- ❖ Vstup: NKA  $M = (Q, \Sigma, \delta, q_0, F)$
- ❖ Výstup: DKA  $M' = (Q', \Sigma, \delta', q'_0, F')$ ,  $L(M) = L(M')$
- ❖ Metoda:
  1. Polož  $Q' = 2^Q$ .
  2. Polož  $q'_0 = \{q_0\}$ .
  3. Polož  $F' = \{S \mid S \in 2^Q \wedge S \cap F \neq \emptyset\}$ .
  4. Pro všechna  $S \in 2^Q$  a pro všechna  $a \in \Sigma$  polož:
    - $\delta'(S, a) = \bigcup_{q \in S} \delta(q, a)$ .

*Důkaz.*  $L(M) = L(M')$

Na deterministické automaty lze pohlížet jako na speciální případ nedeterministických automatů (tj.  $\delta : Q \times \Sigma \rightarrow 2^Q$ ), kdy pro každý  $q \in Q$  a  $a \in \Sigma$  je množina  $\delta(q, a)$  nanejvýš jednoprvková.

Zavedme tedy rozšířenou přechodovou funkci  $\hat{\delta} : Q \times \Sigma^* \rightarrow 2^Q$ , kde

- $\hat{\delta}(q, \varepsilon) = \{q\}$
- $\hat{\delta}(q, wa) = \bigcup_{p \in \hat{\delta}(q, w)} \delta(p, a)$

Nyní ukážeme, že pro každé  $w \in \Sigma^*$  platí  $\hat{\delta}(q_0, w) = \hat{\delta}'(\{q_0\}, w)$ . Indukcí k délce  $w$  dostáváme.

- Pro  $|w| = 0$  platí  $\hat{\delta}(q_0, \varepsilon) = \{q_0\} = \hat{\delta}'(\{q_0\}, \varepsilon)$ .
- Indukční krok: Necht'  $w = va$ , kde  $v \in \Sigma^*$  a  $a \in \Sigma$ . Pak platí  
 $\hat{\delta}(q_0, va) = \bigcup_{p \in \hat{\delta}(q_0, v)} \delta(p, a) = \delta'(\hat{\delta}(q_0, v), a)$  (dle definice  $\delta'$ ) =  
 $\delta'(\hat{\delta}'(\{q_0\}, v), a)$  (dle indukčního předpokladu) =  $\hat{\delta}'(\{q_0\}, va)$ .

Pak platí:  $w \in L(M) \Leftrightarrow \hat{\delta}(q_0, w) \cap F \neq \emptyset \Leftrightarrow \hat{\delta}'(\{q_0\}, w) \cap F \neq \emptyset \Leftrightarrow w \in L(M')$ .

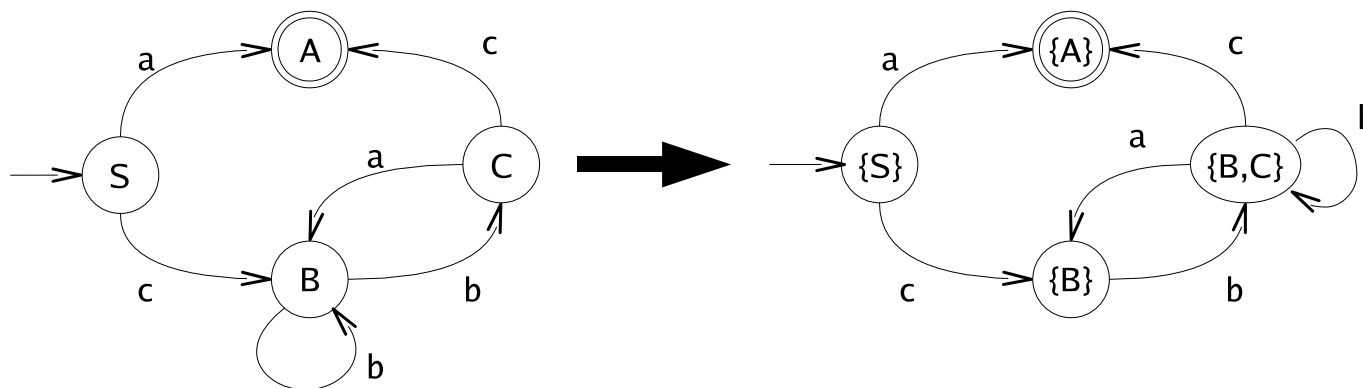
□

**Příklad 1.7** Uvažujme NKA  $M_2 = (\{S, A, B, C\}, \{a, b, c\}, \delta, S, \{A\})$

$$\delta : \quad \delta(S, a) = \{A\} \quad \delta(S, c) = \{B\} \quad \delta(B, b) = \{B, C\} \quad \delta(C, a) = \{B\} \quad \delta(C, c) = \{A\}$$

K nalezení funkce  $\delta'$  příslušného DKA aplikujeme zkrácený postup, využívající skutečnosti, že řada stavů z  $2^Q$  může být nedostupných:

1. Počáteční stav:  $\{S\}$
2.  $\delta'(\{S\}, a) = \{A\}$  — koncový stav  
 $\delta'(\{S\}, c) = \{B\}$
3.  $\delta'(\{B\}, b) = \{B, C\}$
4.  $\delta'(\{B, C\}, a) = \delta(B, a) \cup \delta(C, a) = \{B\}$   
 $\delta'(\{B, C\}, b) = \{B, C\}$   $\delta'(\{B, C\}, c) = \{A\}$



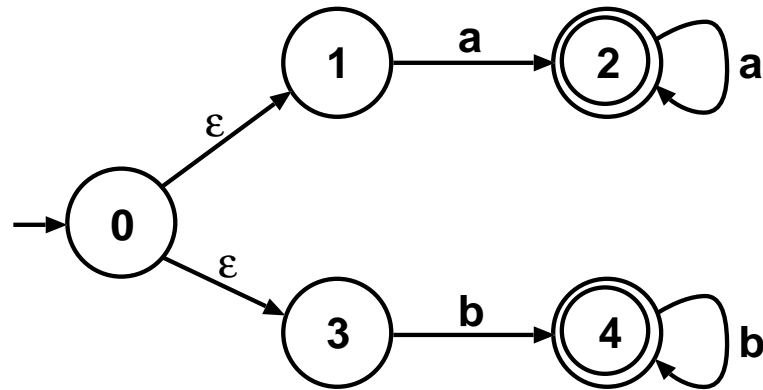
# Rozšířené konečné automaty

❖ Dovolují jednodušší návrh a konstrukci automatů.

**Definice 1.13** Rozšířený konečný automat (RKA) je pětice  $M = (Q, \Sigma, \delta, q_0, F)$ , kde

- $Q$  je konečná množina stavů,
- $\Sigma$  je konečná vstupní abeceda,
- $\delta$  je zobrazení  $Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$ ,
- $q_0 \in Q$  je počáteční stav,
- $F \subseteq Q$  je množina koncových stavů.

**Příklad 1.8**  $M = (\{0, 1, 2, 3, 4\}, \{a, b\}, \delta, 0, \{2, 4\})$



$$L(M) = aa^* + bb^* = a^+ + b^+$$

# $\varepsilon$ -uzávěr

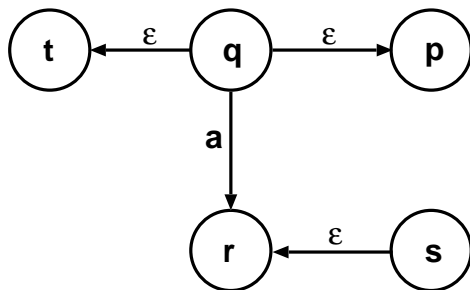
❖ Klíčovou funkcí v algoritmu převodu RKA na DKA má výpočet funkce, která k danému stavu určí množinu všech stavů, jež jsou dostupné po  $\varepsilon$  hranách diagramu přechodů funkce  $\delta$ . Označme tuto funkci jako  $\varepsilon$ -uzávěr:

$$\varepsilon\text{-uzávěr}(q) = \{p \mid \exists w \in \Sigma^* : (q, w) \vdash^* (p, w)\}$$

❖ Funkci  $\varepsilon$ -uzávěr zobecníme tak, aby argumentem mohla být množina  $T \subseteq Q$ :

$$\varepsilon\text{-uzávěr}(T) = \bigcup_{s \in T} \varepsilon\text{-uzávěr}(s)$$

## Příklad 1.9



$$\varepsilon\text{-uzávěr}(\{q, r, s\}) = \{p, q, r, s, t\}$$

# Výpočet $\varepsilon$ -uzávěru

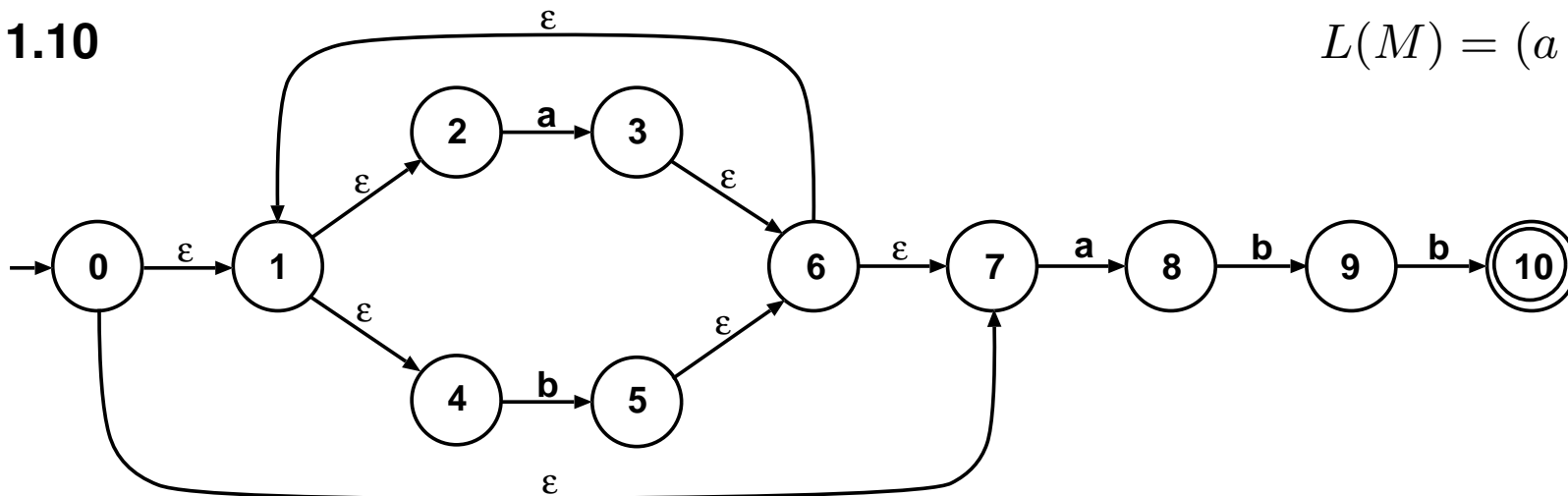
❖ Zavedeme relaci  $\xrightarrow{\varepsilon}$  na množině  $Q$  takto:

$$\forall q_1, q_2 \in Q : q_1 \xrightarrow{\varepsilon} q_2 \stackrel{def}{\iff} q_2 \in \delta(q_1, \varepsilon)$$

Pak  $\varepsilon$ -uzávěr( $p$ ) =  $\{q \in Q \mid p \xrightarrow{\varepsilon^*} q\}$  je reflexivní a tranzitivní uzávěr relace  $\xrightarrow{\varepsilon}$ .

❖ K výpočtu tranzitivního uzávěru použijeme Warshallův algoritmus.

## Příklad 1.10



$$L(M) = (a + b)^* abb$$

$$\varepsilon\text{-uzávěr}(3) = \{3, 6, 7, 1, 2, 4\}$$

$$\varepsilon\text{-uzávěr}(\{1, 0\}) = \{0, 1, 2, 4, 7\}$$

# Převod RKA na ekvivalentní DKA

## Algoritmus 1.2 Převod RKA na DKA

Vstup: RKA  $M = (Q, \Sigma, \delta, q_0, F)$ .

Výstup: DKA  $M' = (Q', \Sigma, \delta', q'_0, F')$ ,  $L(M) = L(M')$ .

Metoda:

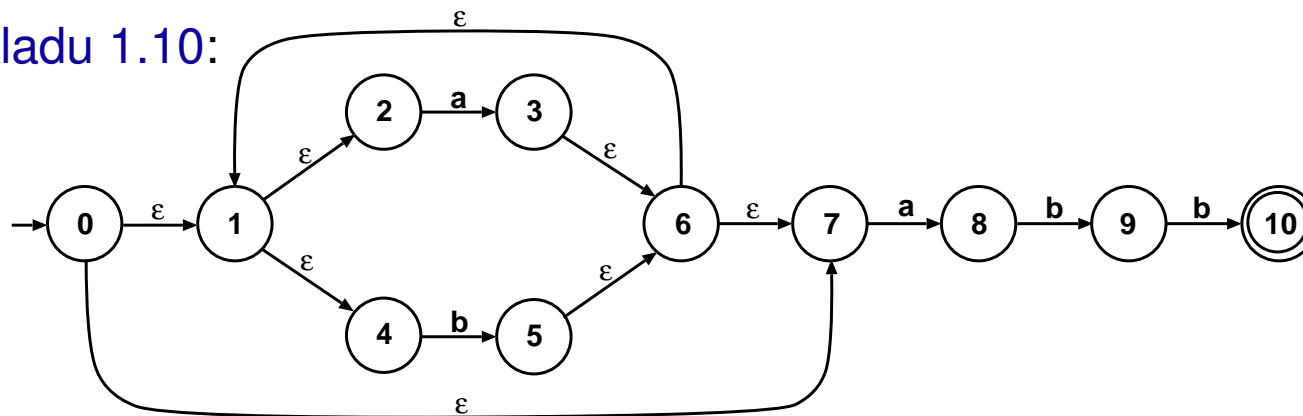
1.  $Q' := 2^Q$ .
2.  $q'_0 := \varepsilon$ -uzávěr( $q_0$ ).
3.  $\delta' : Q' \times \Sigma \rightarrow Q'$  je vypočtena takto:
  - Necht'  $\forall T \in Q', a \in \Sigma : \bar{\delta}(T, a) = \bigcup_{q \in T} \delta(q, a)$ .
  - Pak pro každé  $T \in Q', a \in \Sigma : \delta'(T, a) = \varepsilon$ -uzávěr( $\bar{\delta}(T, a)$ ),
4.  $F' := \{S \mid S \in Q' \wedge S \cap F \neq \emptyset\}$ .



### Příklad 1.11 Aplikujeme algoritmus na automat z příkladu 1.10:

1. Počáteční stav, označíme ho  $A$ , je  $A = \varepsilon\text{-uzávěr}(0) = \{0, 1, 2, 4, 7\}$ .
2.  $\delta'(A, a) = \varepsilon\text{-uzávěr}(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\} = B$ .
3.  $\delta'(A, b) = \varepsilon\text{-uzávěr}(\{5\}) = \{1, 2, 4, 5, 6, 7\} = C$ .
4.  $\delta'(B, a) = \varepsilon\text{-uzávěr}(\{3, 8\}) = B$ .
5.  $\delta'(B, b) = \varepsilon\text{-uzávěr}(\{5, 9\}) = \{1, 2, 4, 5, 6, 7, 9\} = D$ .
6.  $\delta'(C, a) = \varepsilon\text{-uzávěr}(\{3, 8\}) = B$ .
7.  $\delta'(C, b) = \varepsilon\text{-uzávěr}(\{5\}) = C$ .
8.  $\delta'(D, a) = \varepsilon\text{-uzávěr}(\{3, 8\}) = B$ .
9.  $\delta'(D, b) = \varepsilon\text{-uzávěr}(\{5, 10\}) = \{1, 2, 4, 5, 6, 7, 10\} = E$ .
10.  $\delta'(E, a) = \varepsilon\text{-uzávěr}(\{3, 8\}) = B$ .
11.  $\delta'(E, b) = \varepsilon\text{-uzávěr}(\{5\}) = C$ .
12. Množina koncových stavů  $F = \{E\}$ .

Automat z příkladu 1.10:



# Převod gramatiky typu 3 na NKA

**Věta 1.5** Nechť  $\mathcal{L}_M$  je množina (třída) všech jazyků přijímaných konečnými automaty a nechť  $L$  je libovolný jazyk typu 3 ( $L \in \mathcal{L}_3$ ). Pak existuje konečný automat  $M$  takový, že:

$$L = L(M), \text{ tj. } \mathcal{L}_3 \subseteq \mathcal{L}_M.$$

*Důkaz.*

1. Ke gramatice  $G = (N, \Sigma, P, S)$  sestrojíme NKA  $M = (Q, \Sigma, \delta, q_0, F)$  takto:

(a)  $Q = N \cup \{q_F\}$

(b)  $\Sigma = \Sigma$

(c)  $\delta : \delta(A, a)$  obsahuje  $B$ , právě když  $A \rightarrow aB$  je v  $P$

(d)  $\delta : \delta(A, a)$  obsahuje  $q_F$ , právě když  $A \rightarrow a$  je v  $P$

(e)  $q_0 = S$

(f)  $F = \{A \mid A \rightarrow \varepsilon \text{ je v } P\} \cup \{q_F\}$

*Důkaz pokračuje dále.*

*Pokračování důkazu.*

2. Matematickou indukcí ukážeme, že  $L(G) = L(M)$ . Indukční hypotézu formulujeme obecněji ve tvaru:

$$\forall A \in N : A \xrightarrow[G]{i+1} w \iff (A, w) \vdash_M^i (C, \varepsilon) \text{ pro } C \in F, w \in \Sigma^*$$

Pro  $i = 0$  dostáváme

$$A \Rightarrow \varepsilon \iff (A, \varepsilon) \vdash^0 (A, \varepsilon) \text{ pro } A \in F$$

a tvrzení tedy platí.

Pro  $i = 1$  dostáváme

$$A \Rightarrow a \iff (A, a) \vdash^1 (q_F, \varepsilon) \text{ a } q_F \in F$$

a tvrzení tedy platí.

Nyní předpokládejme, že dokazovaná hypotéza platí pro  $i > 0$  a položme  $w = ax$ , kde  $a \in \Sigma$  a  $|x| = i - 1$ .

*Důkaz pokračuje dále.*

*Pokračování důkazu.*

3. pokračování.

Dále předpokládejme  $A \Rightarrow aB \xRightarrow{i} ax$ ,

z indukční hypotézy plyne  $B \xRightarrow{i} x \iff (B, x) \vdash^{i-1} (C, \varepsilon), C \in F$

a z definice funkce  $\delta$ :  $A \Rightarrow aB \iff B \in \delta(A, a)$

Dohromady tedy

$$A \Rightarrow aB \xRightarrow{i} ax = w' \iff (A, ax) \vdash^{i-1} (B, x) \vdash^{i-1} (C, \varepsilon), C \in F$$

tedy  $A \xRightarrow{i+1} w' \iff (A, w') \vdash^i (C, \varepsilon), C \in F$

tj. tvrzení platí i pro  $i + 1$ .

Pro případ  $A = S$  je dokázaná hypotéza tvrzením věty, tj.

$$\forall w' \in \Sigma^* : S \xRightarrow{*} w' \iff (S, w') \vdash^* (C, \varepsilon), C \in F, \text{ tj. } L(G) = L(M)$$

□

## Příklad 1.12

Uvažme gramatiku  $G = (\{S, A, B, U, V, X, Y, Z\}, \{a, b, c\}, P, S)$  s pravidly  $P$ :

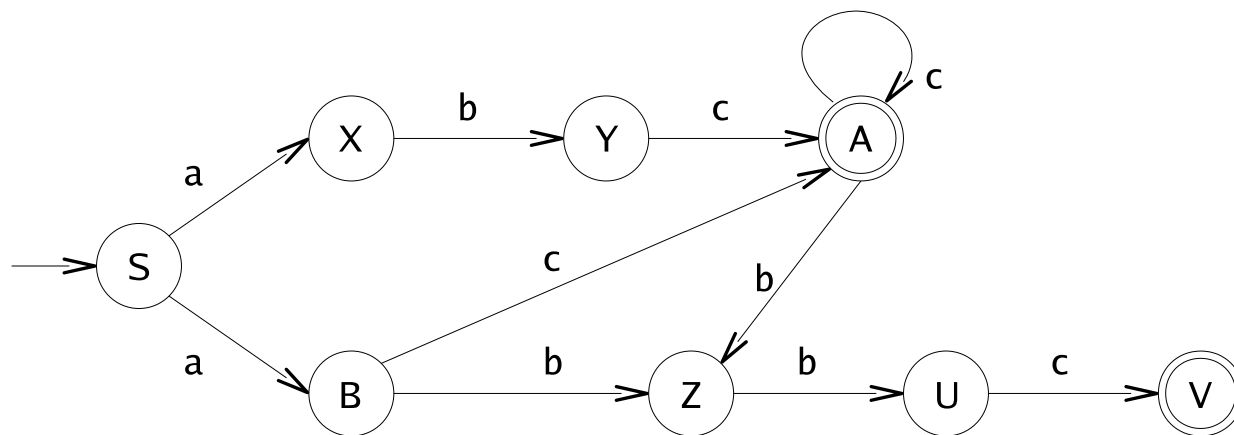
$$S \rightarrow aX \mid aB \qquad A \rightarrow cA \mid bZ \mid \varepsilon$$

$$X \rightarrow bY \qquad B \rightarrow cA \mid bZ$$

$$Y \rightarrow cA \qquad U \rightarrow cV$$

$$Z \rightarrow bU \qquad V \rightarrow \varepsilon$$

Takové gramatice odpovídá konečný automat:



# Převod NKA na gramatiku typu 3

**Věta 1.6** Nechť  $M$  je NKA. Pak existuje gramatika  $G$  typu 3 taková, že:

$$L(M) = L(G), \text{ tj. } \mathcal{L}_M \subseteq \mathcal{L}_3.$$

*Důkaz.* Nechť  $M = (Q, \Sigma, \delta, q_0, F)$ . Předpokládejme, že  $M$  je NKA. Nechť  $G = (Q, \Sigma, P, q_0)$  je gramatika, jejíž pravidla jsou definována takto:

1. pokud  $\delta(q, a)$  obsahuje  $r$ , pak  $P$  obsahuje pravidlo  $q \rightarrow ar$
2. je-li  $p \in F$ , pak  $P$  obsahuje pravidlo  $p \rightarrow \varepsilon$
3. jiná pravidla množina  $P$  neobsahuje.

$G$  je zřejmě typu 3 a indukcí lze dokázat, že platí  $L(G) = L(M)$ .

□

**Příklad 1.13** Uvažujme KA  $M_3 = (\{A, B, C, D\}, \{a, b, c\}, \delta, A, \{C, D\})$ , kde

$$\begin{aligned}\delta : \quad & \delta(A, a) = B & \delta(C, c) = D \\ & \delta(B, b) = A & \delta(D, a) = A \\ & \delta(B, c) = B & \delta(D, b) = D \\ & \delta(B, a) = C\end{aligned}$$

Gramatika  $G$  typu 3, která generuje jazyk  $L(M_3)$ , má tvar:

$$\begin{aligned}G = & (\{A, B, C, D\}, \{a, b, c\}, P, A) \\ P : \quad & A \rightarrow aB & C \rightarrow cD \mid \varepsilon \\ & B \rightarrow bA \mid cB \mid aC & D \rightarrow aA \mid bD \mid \varepsilon\end{aligned}$$

# Regulární množiny a výrazy



# Regulární množiny

**Definice 1.14** Necht'  $\Sigma$  je konečná abeceda. Regulární množinu nad  $\Sigma$  definujeme rekurzívně takto:

1.  $\emptyset$  (tj. prázdná množina) je regulární množina nad  $\Sigma$ ,
2.  $\{\varepsilon\}$  je regulární množina nad  $\Sigma$ ,
3.  $\{a\}$  je regulární množina nad  $\Sigma$  pro všechny  $a \in \Sigma$ ,
4. jsou-li  $P$  a  $Q$  regulární množiny nad  $\Sigma$ , pak také
  - (a)  $P \cup Q$ ,
  - (b)  $P.Q$ ,
  - (c)  $P^*$jsou regulární množiny nad  $\Sigma$ .

5. Žádné jiné množiny, než ty, které lze získat pomocí výše uvedených pravidel, nejsou regulárními množinami.

**Příklad 1.14**  $L = (\{a\} \cup \{d\}).(\{b\}^*).\{c\}$  je regulární množina nad  $\Sigma = \{a, b, c, d\}$ .

# Regulární výrazy

**Definice 1.15** Regulární výrazy nad  $\Sigma$  a regulární množiny, které označují, jsou rekurzívně definovány takto:

1.  $\emptyset$  je regulární výraz označující regulární množinu  $\emptyset$ ,
2.  $\varepsilon$  je regulární výraz označující regulární množinu  $\{\varepsilon\}$ ,
3.  $a$  je regulární výraz označující regulární množinu  $\{a\}$  pro všechny  $a \in \Sigma$ ,
4. jsou-li  $p, q$  regulární výrazy označující regulární množiny  $P$  a  $Q$ , pak
  - (a)  $(p + q)$  je regulární výraz označující regulární množinu  $P \cup Q$ ,
  - (b)  $(pq)$  je regulární výraz označující regulární množinu  $P.Q$ ,
  - (c)  $(p^*)$  je regulární výraz označující regulární množinu  $P^*$ .
5. Žádné jiné regulární výrazy nad  $\Sigma$  neexistují.

## ❖ Konvence:

1. Regulární výraz  $p^+$  značí regulární výraz  $pp^*$ .
2. Abychom minimalizovali počet používaných závorek, stanovujeme **priority operátorů**:
  1.  $*$ ,  $+$  (iterace – nejvyšší priorita),
  2.  $.$  (konkatenace),
  3.  $+$  (alternativa).

## Příklad 1.15

1.  $01$  odpovídá  $\{01\}$ .
2.  $0^*$  odpovídá  $\{0\}^*$ .
3.  $(0 + 1)^*$  odpovídá  $\{0, 1\}^*$ .
4.  $(0 + 1)^*011$  značí množinu řetězců nad  $\{0, 1\}$  končících  $011$ .
5.  $(a + b)(a + b + 0 + 1)^*(0 + 1)$  značí množinu řetězců nad  $\{a, b, 0, 1\}$ , které začínají symbolem  $a$  nebo  $b$  a končí symbolem  $0$  nebo  $1$ .

# \*Kleeneho algebra\*

**Definice 1.16** Kleeneho algebra sestává z neprázdné množiny se dvěma význačnými konstantami 0 a 1, dvěma binárními operacemi + a . a unární operací \*, které splňují následující axiomy:

$$a + (b + c) = (a + b) + c \quad \text{asociativita } + \quad \text{[A.1]}$$

$$a + b = b + a \quad \text{komutativita } + \quad \text{[A.2]}$$

$$a + a = a \quad \text{idempotence } + \quad \text{[A.3]}$$

$$a + 0 = a \quad 0 \text{ je identitou pro } + \quad \text{[A.4]}$$

$$a(bc) = (ab)c \quad \text{asociativita } . \quad \text{[A.5]}$$

$$a1 = 1a = a \quad 1 \text{ je identitou pro } . \quad \text{[A.6]}$$

$$a0 = 0a = 0 \quad 0 \text{ je anihilátorem pro } . \quad \text{[A.7]}$$

$$a(b + c) = ab + ac \quad \text{distributivita zleva} \quad \text{[A.8]}$$

$$(a + b)c = ac + bc \quad \text{distributivita zprava} \quad \text{[A.9]}$$

$$1 + aa^* = a^* \quad \text{[A.10]}$$

$$1 + a^*a = a^* \quad \text{[A.11]}$$

$$b + ac \leq c \Rightarrow a^*b \leq c \quad \text{[A.12]}$$

$$b + ca \leq c \Rightarrow ba^* \leq c \quad \text{[A.13]}$$

V A.12 a A13 reprezentuje  $\leq$  uspořádání definované takto:  $a \leq b \stackrel{def}{\iff} a + b = b$ .

## ❖ Příklady Kleeneho algeber:

- Třída  $2^{\Sigma^*}$  všech podmnožin  $\Sigma^*$  s konstantami  $\emptyset$  a  $\{\varepsilon\}$  a operacemi  $\cup$ ,  $\cdot$  a  $*$ .
- Třída všech regulárních podmnožin  $\Sigma^*$  s konstantami  $\emptyset$  a  $\{\varepsilon\}$  a operacemi  $\cup$ ,  $\cdot$  a  $*$ .
- Třída všech binárních relací nad množinou  $X$  s konstantami v podobě prázdné relace a identity a  $\cup$ , kompozicí (součinem) binárních relací a reflexivním tranzitivním uzávěrem binární relace jako operacemi.
- Matice nad Kleeneho algebrami.

❖ Vlastnosti Kleeneho algeber umožňují snadno řešit **systemy lineárních rovnic** nad těmito algebrami.

❖ Kleeneho algebra nad regulárními výrazy je klíčová pro úpravy a zjednodušování RV.

# Rovnice nad regulárními výrazy

**Definice 1.17** Rovnice, jejímiž složkami jsou koeficienty a neznámé, které reprezentují (dané a hledané) regulární výrazy, nazýváme **rovnici nad regulárními výrazy**.

**Příklad 1.16** Uvažujme rovnici nad regulárními výrazy nad abecedou  $\{a, b\}$

$$X = aX + b$$

Její řešením je regulární výraz  $X = a^*b$ .

*Důkaz.*

- $LS = a^*b$
- $PS = a(a^*b) + b = a^+b + b = (a^+ + \varepsilon)b = a^*b$ .

□

❖ Ne vždy existuje **jediné** řešení rovnice nad reg. výrazy.

**Věta 1.7** Nejmenším pevným bodem („nejmenším řešením“) rovnice  $X = pX + q$  je:

$$X = p^*q$$

*Důkaz.*

- $PS = p^*q$
- $LS = pp^*q + q = (pp^* + \varepsilon)q = p^*q$
- Minimalita plyne přímo z A.12.

□

# Soustavy rovnic nad regulárními výrazy

**Definice 1.18** Soustava rovnic nad reg. výrazy je ve **standardním tvaru** vzhledem k neznámým  $\Delta = \{X_1, X_2, \dots, X_n\}$ , má-li soustava tvar

$$\bigwedge_{i \in \{1, \dots, n\}} X_i = \alpha_{i0} + \alpha_{i1}X_1 + \alpha_{i2}X_2 + \dots + \alpha_{in}X_n$$

kde  $\alpha_{ij}$  jsou reg. výrazy nad nějakou abecedou  $\Sigma$ ,  $\Sigma \cap \Delta = \emptyset$ .

**Věta 1.8** Je-li soustava rovnic nad reg. výrazy ve std. tvaru, pak **existuje její minimální pevný bod a algoritmus jeho nalezení**.

*Důkaz.* Vyjadřujeme hodnotu jednotlivých proměnných pomocí řešení rovnice  $X = pX + q$  jako regulární výraz s proměnnými, jejichž počet se postupně snižuje: Z rovnice pro  $X_n$  vyjádříme např.  $X_n$  jako regulární výraz nad  $\Sigma$  a  $X_1, \dots, X_{n-1}$ . Dosadíme za  $X_n$  do rovnice pro  $X_{n-1}$  a postup opakujeme. Jsou přitom možné (ale ne nutné) různé optimalizace tohoto pořadí. □



**Příklad 1.17** Řešme soustavu rovnic nad reg. výrazy:

$$(1) X_1 = (01^* + 1)X_1 + X_2$$

$$(2) X_2 = 11 + 1X_1 + 00X_3$$

$$(3) X_3 = \varepsilon + X_1 + X_2$$

- Výraz pro  $X_3$  dosadíme z (3) do (2). Dostaneme soustavu:

$$(4) X_1 = (01^* + 1)X_1 + X_2$$

$$(5) X_2 = 11 + 1X_1 + 00(\varepsilon + X_1 + X_2) = 00 + 11 + (1 + 00)X_1 + 00X_2$$

- Ze (4) vyjádříme  $X_1$  s využitím řešení rovnice  $X = pX + q$ :

$$(6) X_1 = (01^* + 1)^* X_2 = (0 + 1)^* X_2$$

- Dosazením do (5):

$$(7) X_2 = 00 + 11 + (1 + 00)(0 + 1)^* X_2 + 00X_2 = 00 + 11 + (1 + 00)(0 + 1)^* X_2$$

- Vypočtením  $X_2$  jako řešení rovnice  $X = pX + q$  dostaneme:

$$(8) X_2 = ((1 + 00)(0 + 1)^*)^*(00 + 11)$$

- Dosazením do (6) dostaneme:

$$(9) X_1 = (0 + 1)^*((1 + 00)(0 + 1)^*)^*(00 + 11) = (0 + 1)^*(00 + 11)$$

- Dosazením do (3) dostaneme:

$$\begin{aligned}(10) X_3 &= \varepsilon + (0 + 1)^*(00 + 11) + ((1 + 00)(0 + 1)^*)^*(00 + 11) = \\ &= \varepsilon + ((0 + 1)^* + ((1 + 00)(0 + 1)^*)^*)(00 + 11) = \\ &= \varepsilon + (0 + 1)^*(00 + 11)\end{aligned}$$

# Regulární množiny a jazyky typu 3

**Věta 1.9** Jazyk  $L$  je regulární množinou právě tehdy, je-li  $L$  jazykem typu 3. Označíme-li  $\mathcal{L}_R$  třídu všech regulárních množin, pak:

$$\mathcal{L}_R = \mathcal{L}_3$$

*Důkaz.* I.  $\mathcal{L}_R \subseteq \mathcal{L}_3$ , tj. každou regulární množinu lze generovat gramatikou typu 3.

<i>regulární množina</i>	<i>gramatika typu 3</i>
(1) $\emptyset$	$G_\emptyset = (\{S\}, \Sigma, \emptyset, S)$
(2) $\{\varepsilon\}$	$G_\varepsilon = (\{S\}, \Sigma, \{S \rightarrow \varepsilon\}, S)$
(3) $\{a\}$ pro každé $a \in \Sigma$	$G_a = (\{S\}, \Sigma, \{S \rightarrow a\}, S)$

Nyní ukážeme, že sjednocení, konkatenaci a iteraci reg. množin lze generovat rovněž gramatikou typu 3. Nechť tedy

- $L_1 = L(G_1)$ , kde  $G_1 = (N_1, \Sigma_1, P_1, S_1)$ ,
- $L_2 = L(G_2)$ , kde  $G_2 = (N_2, \Sigma_2, P_2, S_2)$

a  $G_1, G_2$  jsou gramatiky typu 3,  $N_1 \cap N_2 = \emptyset$  (nonterminály je vždy možno takto odlišit).

*Důkaz pokračuje dále.*

## Pokračování důkazu.

$G_4 = (N_4, \Sigma_1 \cup \Sigma_2, P_4, S_4)$ , kde

(4)  $L_1 \cup L_2$

- $N_4 = N_1 \cup N_2 \cup \{S_4\}$ ,  $S_4 \notin N_1 \cup N_2$ ,
- $P_4 = \{S_4 \rightarrow \alpha \mid S_1 \rightarrow \alpha \in P_1 \vee S_2 \rightarrow \alpha \in P_2\} \cup P_1 \cup P_2$

$G_5 = (N_1 \cup N_2, \Sigma_1 \cup \Sigma_2, P_5, S_1)$  a  $P_5$  je nejmenší množina splňující:

(5)  $L_1.L_2$

- je-li  $(A \rightarrow xB) \in P_1$ , pak  $(A \rightarrow xB) \in P_5$ ,
- je-li  $(A \rightarrow x) \in P_1$ , pak  $(A \rightarrow xS_2) \in P_5$ ,
- je-li  $(A \rightarrow \varepsilon) \in P_1$ , pak  $(A \rightarrow \alpha) \in P_5$  pro všechna pravidla  $(S_2 \rightarrow \alpha) \in P_2$ ,
- $\forall (A \rightarrow \alpha) \in P_2 : (A \rightarrow \alpha) \in P_5$ .

$G_6 = (N_1 \cup \{S_6\}, \Sigma_1, P_6, S_6)$ ,  $S_6 \notin N_1$  a  $P_6$  je nejmenší množina splňující:

(6)  $L_1^*$

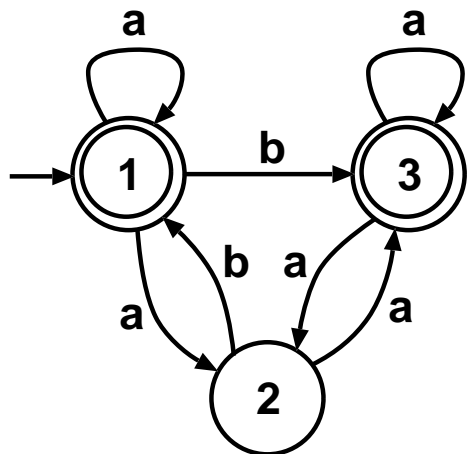
- je-li  $(A \rightarrow xB) \in P_1$ , pak  $(A \rightarrow xB) \in P_6$ ,
- je-li  $(A \rightarrow x) \in P_1$ , pak  $(A \rightarrow xS_6) \in P_6$ ,
- je-li  $(A \rightarrow \varepsilon) \in P_1$ , pak  $(A \rightarrow \alpha) \in P_6$  pro všechna pravidla  $(S_1 \rightarrow \alpha) \in P_1$ ,
- je-li  $(S_1 \rightarrow xB) \in P_1$ , pak  $(S_6 \rightarrow xB) \in P_6$
- je-li  $(S_1 \rightarrow x) \in P_1$ , pak  $(S_6 \rightarrow xS_6) \in P_6$
- $(S_6 \rightarrow \varepsilon) \in P_6$ .

*Pokračování důkazu.* II.  $\mathcal{L}_3 \subseteq \mathcal{L}_R$ , tj. každý jazyk generovaný gramatikou typu 3 je regulární množinou.

- Necht'  $L \in \mathcal{L}_3$  je libovolný jazyk typu 3. Již víme, že ho můžeme popsat KA  $M = (Q, \Sigma, \delta, q_0, F)$ . Necht'  $Q = \{q_0, q_1, \dots, q_n\}$ .
- Vytvoříme **soustavu rovnic** na reg. výrazy s proměnnými  $X_0, X_1, \dots, X_n$  ve standardním tvaru. Rovnice pro  $X_i$  popisuje množinu řetězců přijímaných ze stavu  $Q_i$ .
- Řešením této soustavy získáme reg. výraz pro proměnnou  $X_0$ , který reprezentuje jazyk  $L$ .

□

### Příklad 1.18



$$\begin{aligned}
 X_1 &= \varepsilon + aX_1 + aX_2 + bX_3 \\
 X_2 &= bX_1 + aX_3 \\
 X_3 &= \varepsilon + aX_2 + aX_3
 \end{aligned}$$

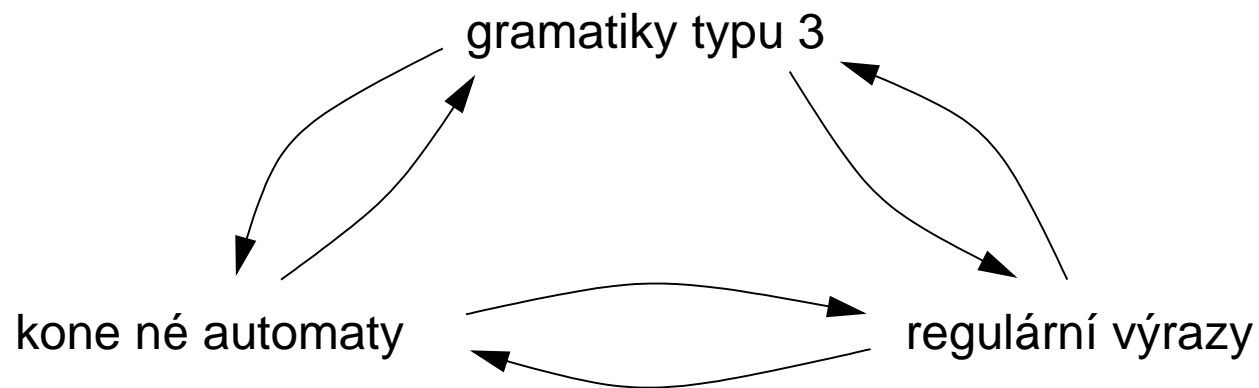
Jazyk  $L$  popisuje reg. výraz, který je řešením této soustavy pro proměnnou  $X_1$ .

# Vztahy regulárních gramatik, KA a RV

❖ Můžeme tedy shrnout, že

- gramatiky typu 3
- (rozšířené/nedeterministické/deterministické) **konečné automaty** a
- **regulární výrazy**

mají ekvivalentní vyjadřovací sílu.



❖ \*Alternativní algoritmy pro převod viz opora\*