I. Matematické základy informatiky

Update: 7. května 2018

1 Konečné automaty, regulární výrazy, uzávěrové vlastnosti třídy regulárních jazyků.

1.1 Konečné automaty

Konečný automat (KA) tvoří množina stavů, vstupní abceda, přechodová funkce, počáteční a koncové stavy. Můžeme jej znázornit jako **tabulku**, **graf** či **strom**.

Konečné automaty se dělí na **determistické** a **nedeterministické**. Deterministický konečný automat má pouze jeden počáteční stav a přechodová funkce vrací jeden stav. Zatímco nedeterministický KA může mít více počátečních stavů a přechodová funkce vrací množinu stavů.

- **Slovo** přijaté automatem je taková sekvence symbolů (ze vstupní abecedy), pro kterou automat skončí v koncovém stavu.
- **Regulární jazyk** je takový jazyk (množina slov) který lze popsat konečným automatem.

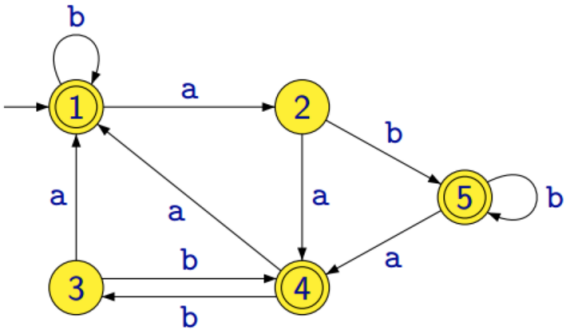
1.1.1 Deterministický konečný automat (DKA)

Skládá se ze **stavů** a **přechodů**. Jeden ze stavů je označen jako **počáteční stav** a některé jsou označeny jako **přijímací**. **Je definován jako uspořádaná pětice** $(Q, \Sigma, \delta, q_0, F)$, kde:

- Q je konečná neprázdná množina **stavů**.
- Σ (*sigma*) je konečná neprázdná množina vstupních symbolů, tzv. **vstupní abeceda**.
- δ (*delta*) je **přechodová funkce**, $\delta : Q \times \Sigma \to Q$.
- q_0 je **počáteční stav**, $q_0 \in Q$.
- F je neprázdná množina **koncových** neboli **přijímajících stavů**, $F \subseteq Q$.

Příklad

- $Q = \{1, 2, 3, 4, 5\}$, $\Sigma = \{a, b\}$, $F = \{1, 4, 5\}$
- $\delta(1, a) = 2$; $\delta(1, b) = 1$; $\delta(3, a) = 1$; $\delta(3, b) = 4$; $\delta(2, a) = 4$; $\delta(2, b) = 5$; $\delta(4, a) = 1$; $\delta(4, b) = 3$; $\delta(5, a) = 4$; $\delta(5, b) = 5$

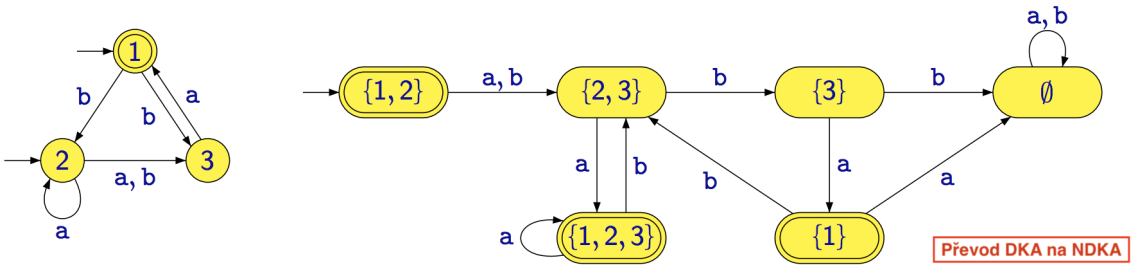


δ	a	b
↔ 1	2	1
2	4	5
3	1	4
← 4	1	3
← 5	4	5

1.1.2 Nedeterministický konečný automat (NDKA)

Formálně je NDKA definován jako pětice $A = (Q, \Sigma, \delta, I, F)$, s tím rozdílem, že oproti deterministickému KA má **více počátečních stavů** a **přechodová funkce vrací množinu** stavů:

- δ je přechodová funkce, vrací množinu stavů, $\delta : Q \times \Sigma \to P(Q)$.
- I je konečná množina počátečních stavů, $I \in Q$.



Na rozdíl od deterministického automatu:

- Může z jednoho stavu vést **libovolný počet přechodů** označených stejným symbolem (i **nulové** ϵ).
- Není zde nutné, aby z každého stavu vystupovaly všechny symboly, které do něj vstoupily → **nemusí ošetřovat všechny varianty**, pouze odhadne, kterou cestou půjde.
- Nedeterministický automat přijímá dané slovo, jestliže **existuje alespoň jeden jeho výpočet**, který vede k přijetí tohoto slova.
- V automatu může být **víc než jeden počáteční stav**.
- Lze ho **převést na deterministický**. Při převodu automatu, který má n stavů může mít výsledný nedeterministický až 2^n stavů.

1.1.3 Normovaný tvar

Začnu v počátečním stavu a procházím navštívené stavy a vytvářím tabulku. Každý KA má **právě 1** normovaný tvar. Také lze tímto způsobem zjistit, zda jsou automaty **ekvivalentní**.

1.2 Regulární výrazy

Regulární výraz je **řetězec popisující celou množinu řetězců**, konkrétně **regulární jazyk**. Regulární výrazy také můžeme chápat jako jednoduchý způsob, jak **popsat konečný automat** umožňující generovat všechna možná slova patřící do daného jazyka.

V regulárních výrazech využíváme znaky **abecedy** a symboly pro **sjednocení**, **zřetězení** a **iterace** regulárních výrazů. Za regulární výraz se považuje i samotný znak abecedy (např. a) stejně jako **prázdné slovo** ϵ a **prázdný jazyk** $\emptyset$.

1.2.1 Definice regulárních výrazů

Regulární výrazy popisují jazyky nad abecedou $A = \Sigma : \emptyset, \epsilon, a$ (kde $a \in \Sigma$) jsou regulární výrazy:

- $\emptyset$ označuje **prázdný jazyk**,
- ϵ označuje jazyk $\{\epsilon\}$,
- a označuje jazyk $\{a\}$.

Dále, jestliže α, β jsou regulární výrazy, pak i $(\alpha + \beta)$, $(\alpha \cdot \beta)$, $(\alpha*)$ jsou regulární výrazy, kde:

- $(\alpha + \beta)$ označuje **sjednocení** jazyků označených α a β,
- $(\alpha \cdot \beta)$ označuje **zřetězení** jazyků označených α a β,
- $(\alpha*)$ označuje **iteraci** jazyka označeného α.

Neexistují žádné další regulární výrazy než ty definované podle předchozích dvou bodů.

Příklady

Ve všech případech je $\Sigma = \{0, 1\}$:

- **01** (0 a 1) ... jazyk tvořený jedním slovem 01,
- **0+1** (0 nebo 1) ... jazyk tvořený dvěma slovy 0 a 1,
- **(01)*** ... jazyk tvořený slovy $\epsilon, 01, 0101, 010101, \ldots$,
- **(0+1)*** ... jazyk tvořený všemi slovy nad abecedou $\{0, 1\}$,
- **(01)*111(01)*** ... jazyk tvořený všemi slovy obsahující podslovo 111, předcházení i následované libovolným počtem slov 01,
- **(0+1)*00+(01)*111(01)*** ... jazyk tvořený všemi slovy, která buď končí 00 nebo obsahují podslovo 111 předcházené i následované libovolným počtem slov 01,
- **(0+1)*1(0+1)*** ... jazyk tvořený všemi slovy obsahujícími alespoň jeden symbol 1,
- **0*(10*10*)*** ... jazyk tvořený všemi slovy obsahujícími sudý počet symbolů 1.

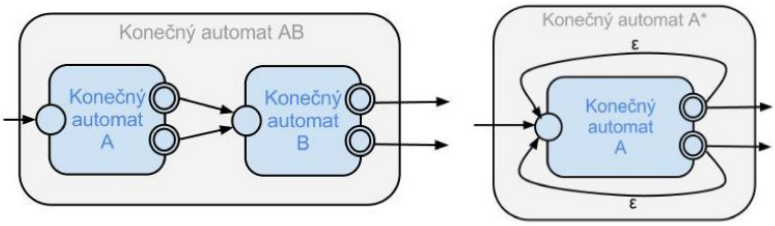
1.3 Uzávěrové vlastnosti třídy regulárních jazyků

Uzavřenost množiny nad operací znamená, že výsledek operace s libovolnými prvky z množiny bude opět spadat do dané množiny. Třídu regulárních jazyků značíme **REG**. Regulární výrazy (tedy i KA) jsou uzavřené vůči operacím:

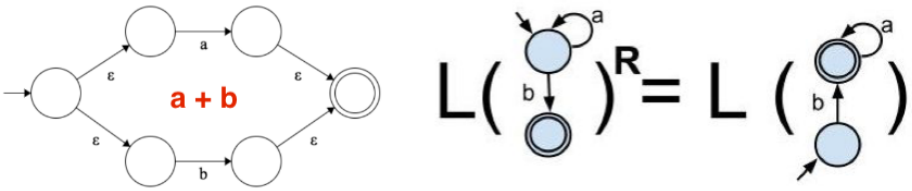
- **Sjednocení**, **průnik**, **doplněk** – je-li $L_1, L_2 \in \text{REG}$, pak také $L_1 \cup L_2, L_1 \cap L_2, L_1'$ jsou v REG.
- **Zřetězení**, **iterace** – je-li $L_1, L_2 \in \text{REG}$, pak také $L_1 \cdot L_2, L_1^*$ jsou v REG.
- **Zrcadlový obraz** – je-li $L \in \text{REG}$, pak také L^R jsou v REG.

1.3.1 Operace sjednocení, zřetězení, iterace a zrcadlový obraz u KA

- **Iterace** – **spojíme koncové stavy** jednoho KA s **počátečními** druhého KA ϵ přechodem. Na obrázku generuje automat A^* jazyk $L(A^*) = L(A)^*$, který je iterací jazyku generovaného modrého automatu A.
- **Zřetězení** – **spojíme koncové stavy jednoho** s **počátečními stavy druhého**. Na obrázku generuje konečný automat AB jazyk $L(AB) = L(A) \cdot L(B)$.



- **Sjednocení** – $L(A + B) = L(A) + L(B)$ získáme tak, že vytvoříme **nový počáteční stav**, ze kterého vedeme ϵ přechody do počátečních stavů obou automatů. Poté obdobě z koncových stavů obou automatů vedeme ϵ přechody do **nového koncového**.
- **Zrcadlový obraz** – pustíme automat pozpátku, celý jej převrátíme. **Přehodíme orientaci všech přechodů**, z počátečních stavů uděláme koncové a naopak.



- **Doplněk** – u DKA provedeme prohození označení příjmajících a ostatních stavů, u NDKA je nejprve nutné provézt převod na DKA.

2 Bezkontextové gramatiky a jazyky. Zásobníkové automaty, jejich vztah k bezkontextovým gramatikám.

2.1 Bezkontextové gramatiky (BG)

Bezkontextová gramatika definuje **bezkontextový jazyk**. Je tvořena **neterminály** (proměnné), **terminály** (konstanty) a **pravidly**, které každému neterminálu definují přepisovací pravidla. Jeden neterminál označíme jako **startovní**, kde začínáme a podle pravidel je dál přepisujeme na výrazy složené z terminálu a neterminálu. Jakmile už není co přepisovat, výraz obsahuje už jen neterminály, získali jsme **slovo**.

- Je **uzavřená** vůči operacím **sjednocení**, **zřetězení**, **iteraci** a **zrcadlový obraz**.
- Ke každé bezkontextové gramatice existuje **ekvivalentní zásobníkový automat**.

2.1.1 Formální definice BG

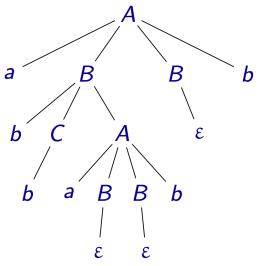
Bezkontextová gramatika je definována jako uspořádaná čtveřice $G = (\Pi, \Sigma, S, P)$, kde:

- Π (*velké pí*) je konečná množina **neterminálních** symbolů (neterminálů).
- Σ je konečná množina **terminálních** symbolů (terminálů), $\Pi \cap \Sigma = \emptyset$.
- S je **počáteční neterminál**, $S \in \Sigma$.
- P je konečná množina **přepisovacích pravidel**, $P \subseteq \Pi \times (\Pi \cup \Sigma)^*$.

2.1.2 Základní pojmy

- **Bezkontextový jazyk** – formální jazyk, který je akceptovaný nějakým zásobníkovým automatem.
- **Derivace slova** – jedno konkrétní odvození slova pomocí gramatiky, tedy záznam postupných přepisů od startovního neterminálu po konečné slovo. Derivace se podle postupu při přepisování dělí na:
  - **levou** – přepisujeme nejprve levé neterminály,
  - **pravou** – přepisujeme nejprve pravé neterminály.
- **Derivační strom** – grafické znázornění derivace slova stromem. Pro všechny možné derivace (levou, pravou, moji) by měl derivační strom být **stejný**. Není-li tomu tak jedná se o **nejednoznačnou gramatiku**, což je nežádoucí jev.
  - **Špatně** = $\mathrm{A} \to \mathrm{A} \mid \epsilon$ (lze generovat až N způsoby), **Správně** = $\mathrm{A} \to \epsilon$
- **Chomského normální forma** – gramatika může obsahovat pouze pravidla typu: **A** $\to$ **BC** nebo **A** $\to$ **a** nebo **S** $\to \epsilon$ (pokud gramatika generuje pouze prázdný řetězec).
- **Nevypouštějící gramatika** – neobsahuje ϵ (*epsilon*) přechody.

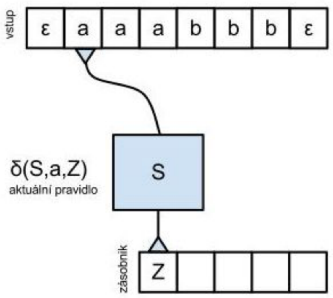
$A \to aBBb \mid AaA$
$B \to \varepsilon \mid bCA$
$C \to AB \mid a \mid b$



$A \Rightarrow aBBb \Rightarrow abCABb \Rightarrow abCaBBbBb \Rightarrow abCaBbBb \Rightarrow abbaBbBb \Rightarrow abbaBbb \Rightarrow abbabb$

2.2 Zásobníkové automaty (ZA)

Slouží k **rozpoznání bezkontextových jazyků**. S využitím zásobníků si může pamatovat kolik a jaké znaky přečetl, což je potřeba právě k rozpoznání bezkontextového jazyka. Zásobníkový automat je v podstatě konečný automat rozšířený o zásobník.



- ZA na základě **aktuálního znaku** na pásce, **prvního znaku v zásobníku** a **aktuálního stavu** změní svůj stav a **přepíše** znak v zásobníku podle daných pravidel.
- ZA **přijímá** dané slovo, jestliže skončí v konfiguraci (q, ϵ, ϵ), tedy když se přečte celé vstupní slovo a zásobník je **prázdný**.
- **Konfigurace** je dána: aktuálním stavem, obsahem pásky a obsahem zásobníku.
- **Deterministický** – nesmí se objevit **stejná konfigurace vícekrát**.

2.2.1 Formální definice zásobníkového automatu

Zásobníkový automat M je definován jako šestice $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$, kde:

- Q je konečná neprázdná množina **stavů**.
- Σ je konečná neprázdná množina **vstupních symbolů** (vstupní abeceda).
- Γ (*velká gamma*) je konečná neprázdná množina **zásobníkových symbolů**.
- δ je **přechodová funkce** (konečná množina instrukcí), $\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma ß P_{\text{fin}}(Q \times \Gamma^*)$.
- q_0 je **počáteční stav**, $q_0 \in Q$.
- Z_0 je **počáteční zásobníkový symbol**, $Z_0 \in \Gamma$.

2.2.2 Definice instrukcí (pravidel) v ZA

Instrukce (sady instrukcí reprezentují přechodovou funkci δ) definují **chování automatu**:

$$(q, a, X) \to (q', \alpha), \text{ kde } a \in \Sigma. \tag{1}$$

Tato instrukce je aplikovatelná jen v situaci (neboli konfiguraci), kdy **řídicí jednotka** je ve stavu q, **čtecí hlava** na vstupní pásce čte symbol a a na vrcholu zásobníku je symbol X. Pokud je **instrukce aplikována**, vykoná se následující:

1. řídicí jednotka **přejde do stavu** q,
2. čtecí hlava na vstupní pásce se **posune o jedno políčko doprava**,
3. vrchní symbol v zásobníku se **odebere** (vymaže),
4. **na vrchol zásobníku se přidá** řetězec α tak, že jeho nejlevější symbol je aktuálním vrcholem zásobníku.

Pravidlo	**Akce (Z = zásobník)**	**Význam**
$\delta(q_1, a, X) \to (q_1, YX)$	**přidání** prvku do Z	na začátek zásobníku se vloží Y
$\delta(q_1, a, X) \to (q_1, Y)$	**přepsání** prvku v Z	první prvek zásobníku se přepíše na Y
$\delta(q_1, a, X) \to (q_1, \epsilon)$	**smazání** prvku ze Z	první prvek zásobníku se smaže neboli nahradí prázdným slovem ϵ
$\delta(q_1, a, X) \to (q_2, X)$	**změna stavu**	stav q_1 se změní na stav q_2
$\delta(q_1, a, X) \to \emptyset$	**pád** automatu	ukončení výpočtu, slovo nebylo přijato

2.3 Převod BG na zásobníkový automat

Využívá se tzv. metody shora-dolů, která obsahuje pouze **1 stav**:

1. pro všechny **neterminály** vypíšu pravidla typu: $(q, \epsilon, A) \to \{(q, B), (q, C)\}$,
2. všechny **terminály** přepíšu na pravidla typu: $(q, a, a) \to (q, \epsilon)$.

Příklad

$S \to A|\ B$
$A \to a$
$B \to (c)$

$(Q, \epsilon, S) \to \{(q, A), (q, B)\}$
$(Q, \epsilon, A) \to (q, a)$
$(Q, \epsilon, B) \to (q, (c))$

$(Q, a, a) \to (q, \epsilon)$
$(Q, (, () \to (q, \epsilon)$
$(Q, c, c) \to (q, \epsilon)$
$(Q,),)) \to (q, \epsilon)$

$\Sigma = \{A, B, S\}$
$\Gamma = \{a, c, (,)\}$

3 Matematické modely algoritmů -Turingovy stroje a stroje RAM. Složitost algoritmu, asymptotické odhady. Algoritmicky nerozhodnutelné problémy.

4 Třídy složitosti problémů. Třída PTIME a NPTIME, NP-úplné problémy.

5 Jazyk predikátové logiky prvního řádu. Práce s kvantifikátory a ekvivalentní transformace formulí.

6 Pojem relace, operace s relacemi, vlastnosti relací. Typy binárních relací. Relace ekvivalence a relace uspořádání.

7 Pojem operace a obecný pojem algebra. Algebry s jednou a dvěma binárními operacemi.

8 FCA – formální kontext, formální koncept, konceptuální svazy. Asociační pravidla, hledání často se opakujících množin položek.

9 Metrické a topologické prostory – metriky a podobnosti.

10 Shlukování.

11 Náhodná veličina. Základní typy náhodných veličin. Funkce určující rozdělení náhodných veličin.

12 Vybraná rozdělení diskrétní a spojité náhodné veličiny - binomické, hypergeometrické, negativně binomické, Poissonovo, exponenciální, Weibullovo, normální rozdělení.

13 Popisná statistika. Číselné charakteristiky a vizualizace kategoriálních a kvantitativních proměnných.

14 Metody statistické indukce. Intervalové odhady. Princip testování hypotéz. Okruhy pokývají předměty Teoretická informatika, Pravděpodobnost a statistika, Matematika pro zpracování znalostí