

Jiří Balun

Formální jazyky a automaty

Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci

Obsah

- 1 Abeceda a řetězce
- 2 Jazyk
- 3 Regulární výrazy (RE) a regulární jazyky (RL)
- 4 Konečný deterministický automat (DFA)
- 5 Reprezentace DFA
- 6 Jazyk DFA
- 7 Konstrukce DFA
- 8 Součtinový DFA
- 9 Konečný nedeterministický automat (NFA)
- 10 Konstrukce NFA
- 11 Determinizace NFA
- 12 NFA s ε -přechody
- 13 Převod RE na ε -NFA
- 14 Převod DFA na RE
- 15 Minimalizace DFA
- 16 Pumping lemma
- 17 Homomorfismy jazyků
- 18 Gramatiky a derivace
- 19 Regulární gramatika (typ 3)
- 20 Bezkontextová gramatika (CFG, typ 2)
- 21 Derivační stromy a víceznačnost CFG
- 22 Chomského normální forma (CNF)
- 23 Algoritmus CYK
- 24 Zásobníkový automat (PDA)
- 25 Návrh PDA
- 26 Převod CFG na PDA
- 27 Gramatiky typu 0 a 1
- 28 Pumping lemma pro bezkontextové jazyky
- 29 Proč neplatí opačná implikace v pumping lemma?
- 30 Domácí úkoly

Seznam zkratek

- CFG – bezkontextová gramatika (*context-free grammar*)
- CFL – bezkontextový jazyk (*context-free language*)
- CNF – Chomského normální forma
- CSG – kontextová gramatika (*context-sensitive grammar*)
- CSL – kontextový jazyk (*context-sensitive language*)
- DFA – deterministický konečný automat (*deterministic finite automaton*)
- DPDA – deterministický zásobníkový automat (*deterministic pushdown automaton*)
- NFA – nedeterministický konečný automat (*nondeterministic finite automaton*)
- PDA – (nedeterministický) zásobníkový automat (*pushdown automaton*)
- PL – pumping lemma
- RE – regulární výraz (*regular expression*)
- RG – regulární gramatika
- RL – regulární jazyk (*regular language*)

1 Abeceda a řetězce

- **abeceda** je konečná (neprázdná) množina znaků, značí se Σ
- **řetězec** s nad abecedou Σ je libovolná konečná posloupnost znaků abecedy
- délka řetězce s se značí $|s|$; prázdný řetězec se značí ε a tedy $|\varepsilon| = 0$
- například pro řetězec $s = s_1 \dots s_n$ (kde $s_i \in \Sigma$ pro $i = 1, \dots, n$) je $|s| = n$
- Σ^* značí množinu všech řetězců nad abecedou Σ
- Σ^+ je množina všech neprázdných řetězců nad abecedou Σ
- například pro $\Sigma = \{0, 1\}$ je:

$$- \Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001 \dots\}$$

$$- \Sigma^+ = \{0, 1, 00, 01, 10, 11, 000, 001 \dots\}$$

poznámka: řetězce pro přehlednost píšeme v tzv. **shortlex** uspořádání

- **zřetězení** dvou řetězců $x = x_1 \dots x_n$ a $y = y_1 \dots y_m$ se značí $x \cdot y$ nebo zkráceně xy

$$xy = x_1 \dots x_n y_1 \dots y_m$$

- triviálně platí tyto vztahy:

$$|xy| =$$

$$x \cdot \varepsilon =$$

- pro řetězce platí $x = y$, právě když $|x| = |y|$ a zároveň $x_i = y_i$ pro všechna $i = 1, \dots, |x|$
- **mocnina** řetězce x^n je definována rekurzivně:

$$x^n = \left\{ \begin{array}{l} \end{array} \right.$$

- **prefix** a **sufix** řetězce:

$$- Pfx(x) =$$

$$- Sfx(x) =$$

- **reverz** řetězce $x = x_1 \dots x_n$ značíme $x^R = x_n \dots x_1$

- mějme řetězece $x = abb$, $y = ca$; doplňte:

1. $xy =$

2. $yx =$

3. $y^3x =$

4. $y(x^2y)^Rx =$

5. $Pfx(x) =$

6. $Sfx(x) =$

2 Jazyk

- **jazyk** je množina řetězců nad zvolenou abecedou Σ , neboli pro jazyk L platí $L \subseteq \Sigma^*$
- **zřetězení jazyků** $L_1 \cdot L_2 =$
- jaký je rozdíl mezi jazyky \emptyset a $\{\varepsilon\}$? Jak dopadnou výrazy:

1. $\emptyset \cdot \{a\} =$

2. $\{\varepsilon\} \cdot \{a\} =$

- mějme $L_1 = \{b, aa\}$, $L_2 = \{\varepsilon, aab\}$ a $L_3 = \{\varepsilon, b\}$ nad abecedou $\Sigma = \{a, b\}$; doplňte:

1. $L_1 \cdot L_2 =$

2. $L_2 \cdot L_1 =$

3. $\{\varepsilon\} \cdot L_1 =$

4. $L_1 \cup L_2 =$

5. $\overline{L_1} =$

6. $(L_1 \cap L_3) \cdot (L_2 \setminus L_3) =$

- **mocnina jazyka:**

$$L^n = \left\{ \right.$$

- **uzávěry jazyků:**

– $L^* =$

– $L^+ =$

- mějme jazyky $L_1 = \{a, b, aa\}$ a $L_2 = \{\varepsilon, bb\}$; doplňte:

1. $L_1^0 =$

2. $L_1^1 =$

3. $L_1^2 =$

4. $L_2^* =$

5. $\{\varepsilon\}^* =$

6. $\{\varepsilon\}^+ =$

7. $\emptyset^* =$

8. $\emptyset^+ =$

- rozhodněte zda platí vztah $(L^R)^* = (L^*)^R$, kde $L^R = \{x^R \mid x \in L\}$ je reverz jazyka

3 Regulární výrazy (RE) a regulární jazyky (RL)

- **regulární výraz** (zkráceně RE z *regular expression*) je formalismus pro čitelný a úsporný zápis regulárních jazyků (samotný výraz je vzor pro dané řetězce)
- jako **regulární jazyky** (zkráceně RL z *regular language*) označujeme třídu jazyků, které lze popsat pomocí regulárních výrazů
- pomocí $L(E)$ značíme jazyk reprezentovaný regulárním výrazem E
- **definice RE nad abecedou Σ :**
 1. pro každý symbol $a \in \Sigma$ je \mathbf{a} RE s jazykem $L(\mathbf{a}) = \{a\}$
 2. ϵ je RE s jazykem $L(\epsilon) = \{\epsilon\}$
 3. \emptyset je RE s jazykem $L(\emptyset) = \emptyset$
 4. nechť E_1 a E_2 jsou RE, pak $E_1 + E_2$ je RE s jazykem $L(E_1 + E_2) = L(E_1) \cup L(E_2)$
 5. nechť E_1 a E_2 jsou RE, pak $E_1 E_2$ je RE s jazykem $L(E_1 E_2) = L(E_1) \cdot L(E_2)$
 6. nechť E je RE, pak E^* je RE s jazykem $L(E^*) = (L(E))^*$
- priorita operátorů: nejvyšší má operace $*$, pak zřetězení \cdot , a nejnižší má operace $+$
- dále můžeme prioritu operací upravit dle libosti pomocí závorek
- pro přehlednější zápis dále zavedeme zkrácený zápis pro tyto RE:
 - Σ jako RE s jazykem $L(\Sigma) = \Sigma$
 - $\mathbf{E}^+ = \mathbf{E E}^*$
- navrhněte RE k následujícím jazykům:

$$L_1 = \{aaa, aab, aaac, aaaaa\}$$

$$L_2 = \{w \in \{a, b\}^* \mid w \text{ obsahuje podřetězce } aaa \text{ a } bbb\}$$

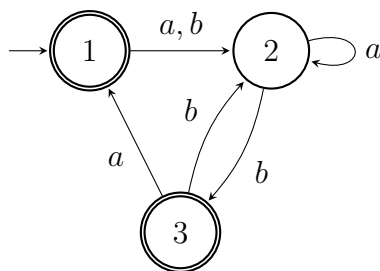
$$L_3 = \{w \in \{a, b\}^* \mid w \text{ neobsahuje podřetězce } ab \text{ a } ba\}$$

$$L_4 = \{w \in \{a, b\}^* \mid w \text{ začíná i končí stejným řetězcem, který je } aa, \text{ nebo } bb\}$$

$$L_5 = \{w \in \{a, b\}^* \mid w \text{ má sudý počet znaků } a \text{ nebo přesně dva znaky } b\}$$

4 Konečný deterministický automat (DFA)

- **konečný deterministický automat**, zkráceně **DFA** (*deterministic finite automaton*) je reprezentován uspořádanou pěticí $A = (Q, \Sigma, \delta, q_0, F)$, kde:
 1. Q je konečná množina stavů (proto název *konečný* automat)
 2. Σ je abeceda
 3. δ je přechodová funkce ve tvaru $\delta : Q \times \Sigma \rightarrow Q$, která stavu a symbolu přiřadí nějaký stav
 4. q_0 je počáteční stav (platí pro něj $q_0 \in Q$)
 5. F je množina koncových/akceptujících stavů (platí pro ně $F \subseteq Q$),
- takto definovaný automat je **deterministický**, protože se v každém kroku výpočtu nachází právě v jednom stavu (později si ukážeme i nedeterministické modely)
- přechodová funkce je **úplná**, pokud je $\delta(q, a)$ definovaná pro všechny $q \in Q$ a $a \in \Sigma$; budeme uvažovat pouze DFA s úplnou přechodovou funkcí
- grafická reprezentace konečných automatů pomocí grafu:
 - stavy jsou reprezentované jako uzly
 - počáteční stav je označený šipkou směřující k uzlu
 - koncové stavy jsou označené dvojitým uzlem
 - každý přechod je reprezentovaný orientovanou hranou, jejíž popisek (jeden nebo více symbolů abecedy) udává, který symbol realizuje přechod mezi počátečním a koncovým uzlem dané hrany
- podle grafu automatu A_1 doplňte:



1. $Q =$
2. $\Sigma =$
3. $q_0 =$
4. $F =$
5. $\delta(1, a) =$
6. $\delta(2, a) =$
7. $\delta(2, b) =$
8. $\delta(\delta(\delta(1, b), a), b) =$
9. $\delta(\delta(\delta(1, a), b), a) =$
10. $\delta(\delta(\delta(\delta(1, b), b), a), a) =$

5 Reprezentace DFA

- přechodovou funkci δ lze zapsat pomocí relace $\delta \subseteq Q \times \Sigma \times Q$, tedy jako množinu uspořádaných trojic, kde $\langle p, a, q \rangle \in \delta$ odpovídá přechodu $\delta(p, a) = q$
- mějme DFA $A_1 = (\{1, 2, 3\}, \{a, b\}, \delta, 1, \{3\})$, jehož přechodová funkce je zadána jako relace $\delta = \{\langle 1, a, 1 \rangle, \langle 1, b, 2 \rangle, \langle 2, a, 2 \rangle, \langle 2, b, 3 \rangle, \langle 3, a, 3 \rangle, \langle 3, b, 3 \rangle\}$; takto zadaný automat lze reprezentovat:

1. **grafem** – stavy jsou uzly, přičemž počáteční stav je označen šipkou a koncové stavy jsou označeny dvojitým uzlem; orientované hrany jsou přechody, kde hrana z p do q s popiskem a značí přechod $\delta(p, a) = q$

2. **tabulkou** – první sloupce udává stavy automatu, počáteční stav je opět označen šipkou a koncové stavy hvězdou; další sloupce pak udávají výsledky přechodu pro jednotlivé symboly abecedy

3. **v počítači** – jednoduchá implementace v LISPU může vypadat následovně:

```
1 (defun DFA (delta q0 F)
2   (let ((state q0))
3     (labels ((st-cmp (a transition)
4                 (and (equal (car transition) state)
5                        (equal (second transition) a))))
6       (lambda (&optional a)
7         (values ;; 1st value is resulting state, 2nd value is state type
8           (setf state (if a
9                           (third (find a delta :test #'st-cmp))
10                          state)) ;; no input, return actual state
11         (if (find state F) 'final 'non-final))))))
12
13 (defvar A (DFA '((1 a 1) (1 b 2) (2 a 2) (2 b 3) (3 a 3) (3 b 3)) 1 '(3)))
```

```
CL-USER> (funcall A 'b)
```

```
2
```

```
non-final      ⇐ stav 2 není koncový
```

```
CL-USER> (funcall A 'b)
```

```
3
```

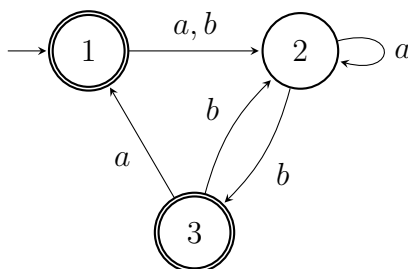
```
final          ⇐ stav 3 je koncový
```


6 Jazyk DFA

- definujeme si **rozšířenou přechodovou funkci** $\hat{\delta}$, kde $q \in Q$ a $s = s_1 \dots s_n \in \Sigma^*$:

$$\hat{\delta}(q, s) = \left\{ \begin{array}{l} \end{array} \right.$$

- jaký je výsledek přechodů v následujícím automatu, a které z nich přijme?



1. $\hat{\delta}(1, aabbaa) =$

2. $\hat{\delta}(1, abaa) =$

3. $\hat{\delta}(1, abbab) =$

- jazyk, který rozpoznává DFA A , budeme značit $L(A)$
- pokud se A po přečtení slova s přesune z počátečního stavu do koncového stavu, pak s patří do $L(A)$; **jazyk DFA** formálně definujeme jako:

$$L(A) =$$

- DFA rozpoznávají třídu **regulárních jazyků** (jsou tedy ekvivalentní s RE)
- jaký je jazyk automatu A_1 ze Sekce 5?

7 Konstrukce DFA

Navrhňte automaty rozponávající jazyky:

- $L_1 = \{\varepsilon\}$ nad abecedou $\Sigma = \{a, b\}$; jak by vypadal automat pro jazyky \emptyset a Σ^* ?
- $L_2 = \{a, b, ca, cb, cc, abc\}$ nad abecedou $\Sigma = \{a, b, c\}$
- $L_3 = \{s \mid s \text{ začíná řetězcem } abb\}$ nad abecedou $\Sigma = \{a, b\}$
- $L_4 = \{s \mid s \text{ obsahuje sudý počet znaků } a\}$ nad abecedou $\Sigma = \{a, b, c\}$
- $L_5 = \{s \mid s \text{ začíná a končí znakem } a\}$ nad abecedou $\Sigma = \{a, b\}$

- $L_6 = \{a\}^* \cup \{b\}^*$ nad abecedou $\Sigma = \{a, b\}$

- $L_7 = \{s \mid s \text{ obsahuje podřetězec } abab\}$ nad abecedou $\Sigma = \{a, b\}$

- $L_8 = \{s \mid s \text{ neobsahuje podřetězec } abab\}$ nad abecedou $\Sigma = \{a, b\}$

- $L_9 = \{s \mid \text{součet hodnot znaků } s \text{ je dělitelný číslem 4 beze zbytku}\}$ nad abecedou $\Sigma = \{0, 1, 2, 3\}$ (součet prázdné sekvence je 0, a tedy $\varepsilon \in L_9$)

8 Součinný DFA

- součinný/produktový DFA simuluje běh více DFA zároveň
- tuto konstrukci lze použít k vytvoření DFA, který rozpoznává průnik nebo sjednocení regulárních jazyků, nebo k ověření ekvivalence a inkluze regulárních jazyků
- **Konstrukce:** pro dva DFA $A_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1)$ a $A_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$ vytvoříme součinný DFA jako $A_1 \times A_2 = (Q_1 \times Q_2, \Sigma, \delta', \langle q_{0,1}, q_{0,2} \rangle, F')$, kde:
 - množina stavů obsahuje všechny dvojice stavů $\langle q_1, q_2 \rangle$, kde $q_1 \in Q_1$ a $q_2 \in Q_2$
 - abeceda zůstává stejná (oba automaty A_1 a A_2 musí být nad stejnou abecedou)
 - $\delta'(\langle q_1, q_2 \rangle, a) = \langle \delta_1(q_1, a), \delta_2(q_2, a) \rangle$ pro všechna $a \in \Sigma$
 - počáteční stav $\langle q_{0,1}, q_{0,2} \rangle$ je dvojice počátečních stavů z A_1 a A_2
 - množinu koncových stavů F' zvolíme podle účelu $A_1 \times A_2$
- **Volba koncových stavů:** nechť $L_1 = L(A_1)$ a $L_2 = L(A_2)$, pak:
 - pro ověření $L_1 = L_2$ (tj. platí $L_1 = L_2$ právě když $L(A_1 \times A_2) = \emptyset$) zvolíme:
$$F' = \{ \langle q_1, q_2 \rangle \mid (q_1 \in F_1 \wedge q_2 \notin F_2) \vee (q_1 \notin F_1 \wedge q_2 \in F_2) \}$$

poznámka: pro ověření $L_1 \subseteq L_2$ použijeme jen první část této podmínky
 - pro rozpoznání $L(A_1 \times A_2) = L_1 \cap L_2$ zvolíme $F' = \{ \langle q_1, q_2 \rangle \mid q_1 \in F_1 \wedge q_2 \in F_2 \}$
 - pro rozpoznání $L(A_1 \times A_2) = L_1 \cup L_2$ zvolíme $F' = \{ \langle q_1, q_2 \rangle \mid q_1 \in F_1 \vee q_2 \in F_2 \}$
 - pro rozpoznání $L(A_1 \times A_2) = L_1 \setminus L_2$ zvolíme $F' = \{ \langle q_1, q_2 \rangle \mid q_1 \in F_1 \wedge q_2 \notin F_2 \}$
- sestrojte součinný DFA pro jazyk nad abecedou $\Sigma = \{a, b\}$:
 - $\{s \mid s \text{ obsahuje podřetězec aba}\} \cap \{s \mid s \text{ neobsahuje podřetězec bb}\}$

9 Konečný nedeterministický automat (NFA)

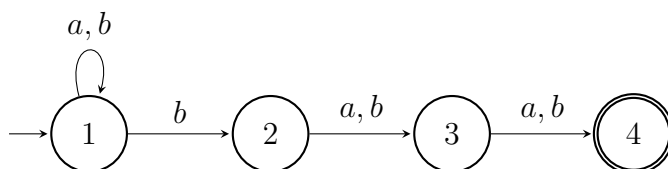
- konečný nedeterministický automat **NFA** (*nondeterministic finite automaton*) je reprezentován uspořádanou pěticí $A = (Q, \Sigma, \delta_N, q_0, F)$
- v definici NFA je oproti DFA jediná změna, a to v přechodové funkci, která je nyní ve tvaru $\delta_N: Q \times \Sigma \rightarrow 2^Q$ (pokud je z kontextu jasné, že se jedná o NFA, budeme psát jen δ místo δ_N)
- NFA se může nacházet ve více stavech najednou (alternativně lze nedeterminismus chápat jako schopnost „uhádnout“ ten správný přechod)
- výsledek přechodu z δ_N je tedy množina stavů – klidně i prázdná množina, proto v NFA už nemusí mít úplnou přechodovou funkci
- rozšířená přechodová funkce $\hat{\delta}: Q \times \Sigma^* \rightarrow 2^Q$, kde $q \in Q$ a $s = s_1 \dots s_n \in \Sigma^*$:

$$\hat{\delta}_N(q, s) = \begin{cases} \{q\} & \text{pro } s = \varepsilon \\ \bigcup_{p \in \hat{\delta}_N(q, s_1 \dots s_{n-1})} \delta_N(p, s_n) & \text{pro } |s| > 0 \end{cases}$$

- pokud se NFA A po přečtení slova s přesune z počátečního stavu aspoň do jednoho koncového stavu, pak slovo s patří do $L(A)$; **jazyk NFA** formálně zapíšeme jako:

$$L(A) =$$

- NFA rozpoznávají regulární jazyky, tedy stejné jazyky jako rozpoznávají DFA/RE
- mějme zadán následující automat A_1 :



1. $\hat{\delta}_N(1, abab) =$

2. $\hat{\delta}_N(1, bbbaab) =$

3. jaký je jazyk $L(A_1)$?

10 Konstrukce NFA

- navrhnete NFA pro následující jazyky:

- $L_1 = \{c, ab, ba, abc, bcb\}$ nad abecedou $\Sigma = \{a, b, c\}$

- $L_2 = \{s \in \{a, b, c\}^* \mid s \text{ obsahuje podslovo } abbc, acbc \text{ nebo } bcab\}$

- $L_3 = \{ab\}^* \cup \{abb\}^*$ nad abecedou $\Sigma = \{a, b\}$

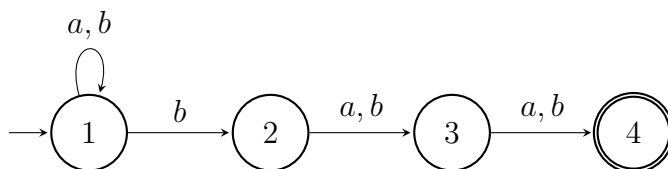
- $L_4 = \{s \in \{a, b\}^* \mid s \text{ má sudý počet znaků } a \text{ nebo přesně dva znaky } b\}$

11 Determinizace NFA

- existuje NFA rozponávající jazyk L právě tehdy, když existuje DFA rozponávající L :
(\Leftarrow) každý DFA je zároveň NFA (jen změním $\delta(q, a) = p$ na $\delta_N(q, a) = \{p\}$)
(\Rightarrow) *determinizace* – ke každému NFA lze sestavit DFA rozponávající stejný jazyk
- **Podmnožinová konstrukce:** k NFA $N = (Q, \Sigma, \delta_N, q_0, F)$ sestavíme DFA $D = (2^Q, \Sigma, \delta_D, \{q_0\}, F')$, tak aby platilo $L(N) = L(D)$:
 1. stavy D jsou podmnožiny Q
 2. počáteční stav je $\{q_0\}$, množina obsahující původní počáteční stav q_0
 3. koncové stavy jsou podmnožiny Q obsahující aspoň jeden původní koncový stav, tedy $F' = \{S \subseteq Q \mid S \cap F \neq \emptyset\}$
 4. δ_D je deterministická přechodová funkce definovaná jako:

$$\delta_D(\{q_1, \dots, q_k\}, a) = \bigcup_{i=1}^k \delta_N(q_i, a)$$

- D má $2^{|Q|}$ stavů, ale nám stačí jen dosažitelné stavy (ke každému automatu existuje automat, který rozponává stejný jazyk, ale má jen dosažitelné stavy)
- determinizujte automat A_1 :



12 NFA s ε -přechody

- rozšířením modelu NFA o ε -přechody získáme ε -NFA $A = (Q, \Sigma, \delta_E, q_0, F)$, který navíc umožňuje i přechody bez přechtení symbolu ze vstupu
- jediná změna je opět v přechodové funkci, která má tvar $\delta_E: Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$
- ε -NFA rozpoznávají regulární jazyky, stejně jako NFA a DFA
- NFA se často definují s množinou počátečních stavů $I \subseteq Q$; každý takový NFA lze upravit na ε -NFA, tak aby měl pouze jeden počáteční stav:
 - vytvoříme nový počáteční stav q_0
 - pro každý stav $p \in I$ přidáme přechod $\langle q_0, \varepsilon, p \rangle$
- ε -uzávěr $E(S)$ je množina dosažitelných stavů pomocí ε -přechodů ze stavů v $S \subseteq Q$
- převod ε -NFA na NFA:**
 - pro každý stav $q \in Q$ vypočítáme ε -uzávěr $E_q = E(\{q\})$
 - vypočítáme δ_N z δ_E pro každý stav $q \in Q$ a symbol $a \in \Sigma$:

$$\delta_N(q, a) = E\left(\bigcup_{p \in E_q} \delta_E(p, a)\right)$$

- označ jako koncový stav q_0 pokud platí $E(\{q_0\}) \cap F \neq \emptyset$
- převeďte následující ε -NFA A_1 na NFA bez ε -přechodů:

δ_E	ε	a	b
$\rightarrow 1$	$\{3\}$	$\{1\}$	$\{2\}$
2	\emptyset	$\{4\}$	$\{2\}$
$3\star$	$\{4\}$	\emptyset	$\{3\}$
4	\emptyset	$\{2, 3\}$	\emptyset

δ_N	E_q	a	b

13 Převod RE na ε -NFA

- ekvivalenci s ε -NFA ukážeme strukturální indukcí vzhledem ke složitosti RE (ke každému bodu definice RE najdeme ε -NFA se stejným jazykem):

1. $L(\mathbf{a}) = \{a\}$:

2. $L(\varepsilon) = \{\varepsilon\}$:

3. $L(\emptyset) = \emptyset$:

4. $L(E_1 + E_2) = L(E_1) \cup L(E_2)$:

5. $L(E_1 E_2) = L(E_1) \cdot L(E_2)$:

6. $L(E^*) = (L(E))^*$:

- převed'te RE na ε -NFA:

1. $(\mathbf{a+ab})^*\mathbf{b}$

2. $\varepsilon + (\mathbf{a+b})^*\mathbf{c(abc)^*}$

14 Převod DFA na RE

- **Konstrukce:** necht' $A = (\{1, \dots, n\}, \Sigma, \delta, 1, F)$ je DFA s n stavy (jsou označené čísly od 1 do n), pak RE R_A , pro který platí $L(R_A) = L(A)$, vypočítáme předpisem:

$$R_A = \sum_{f \in F} R_{1,f}^n$$

kde jednotlivé RE $R_{1,f}^n$ získáme jako:

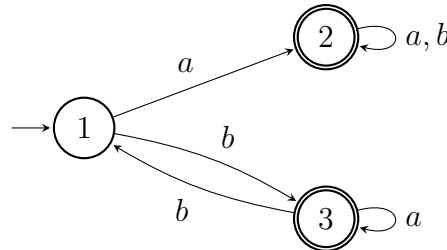
$$\begin{aligned} R_{ij}^k &= R_{ij}^{k-1} + R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1} \\ R_{ij}^0 &= \sum_{a \in \Sigma: \delta(i,a)=j} a \quad (\text{pro } i \neq j) \\ R_{ii}^0 &= \varepsilon + \sum_{a \in \Sigma: \delta(i,a)=i} a \end{aligned}$$

výraz $R_{i,j}^k$ je RE, který popisuje množinu všech cest (grafem automatu) ze stavu i do stavu j přes stavy, jež jsou označeny maximálně číslem k ; výraz vypočítáme z jednodušších podvýrazů:

1. R_{ij}^{k-1} jsou cesty z i do j , které nevedou přes k
2. $R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1}$ jsou cesty z i do j , které vedou přes k

výrazy, kde $k = 0$, slouží jako podmínka ukončující rekurzi (už jsme na úrovni jednotlivých přechodů mezi stavy)

- převed'te DFA na RE:



15 Minimalizace DFA

- chceme k zadanému DFA A najít ekvivalentní DFA A_m s nejmenším počtem stavů
- stavy $p, q \in Q$ v DFA A jsou nerozlišitelné, právě když pro $\forall w \in \Sigma^*$ platí:

$$\hat{\delta}(p, w) \in F \iff \hat{\delta}(q, w) \in F$$

- nerozlišitelnost stavů ' \equiv ' je binární relace na množině stavů Q , která je navíc ekvivalence (ověřte, že \equiv je reflexivní, symetrická a tranzitivní)
- pro stavy výsledného minimálního DFA A_m platí, že jsou to třídy \equiv a množina stavů A_m tvoří rozklad Q podle \equiv (předpokládáme, že A nemá nedosažitelné stavy)
- **Konstrukce:** A_m lze vypočítat pomocí tabulky dvojic stavů, která zachycuje \equiv :
 - nejprve označíme všechny dvojice, které obsahují pouze jeden koncový stav (ty jsme schopni rozlišit hned na začátku)
 - postupně označujeme dvojice $\langle p, q \rangle$, pro které existuje $a \in \Sigma$ takové, že dvojice $\langle \delta(p, a), \delta(q, a) \rangle$ už je označena; tento krok opakujeme dokud ještě lze označit nějakou další dvojici
 - neoznačené dvojice nejsme schopni rozlišit a pomocí tranzitivního uzávěru vypočítáme celou odpovídající třídu \equiv , kterou v A_m sloučíme do jednoho stavu

poznámka: není nutné počítat všechny políčka tabulky (reflexivita + symetrie)

- minimalizujte DFA zadaný tabulkou:

δ	a	b
$\rightarrow 1$	2	3
2	1	4
3*	5	6
4*	5	6
5*	6	6
6	6	6

\equiv	1	2	3	4	5	6
$\rightarrow 1$	✓					
2	*	✓				
3*	*	*	✓			
4*	*	*	*	✓		
5*	*	*	*	*	✓	
6	*	*	*	*	*	✓

16 Pumping lemma

- užitečný nástroj, který umožňuje dokázat, že daný jazyk **není** regulární (neumožňuje dokázat, že jazyk je regulární)
- **Pumping lemma:** nechť L je regulární jazyk, pak existuje číslo $n \in \mathbb{N}$, tak že každý řetězec $w \in L$ délky aspoň n lze rozdělit na tři části $w = xyz$, které splňují:

$$(1) |y| > 0$$

$$(2) |xy| \leq n$$

$$(3) xy^iz \in L \text{ pro všechna } i \geq 0$$

- pokud je jazyk L konečný (a tedy i regulární), pak lemma platí triviálně – n zvolíme větší než je délka nejdelšího řetězce v L

- **postup při použití lemma v důkazu sporem:**

1. předpokládáme, že daný jazyk je regulární a tedy musí existovat $n \in \mathbb{N}$ z lemma (nikdy nevolíme konkrétní n , protože pokud L není regulární, n neexistuje)
2. zvolíme vhodný řetězec w takový, že $|w| \geq n$ (w je tedy nějak vyjádřeno pomocí n ; opět nikdy nevolíme konkrétní řetězec!)
3. ukážeme, že pro každé rozdělení $xyz = w$ splňující podmínky (1) a (2) z lemma existuje i , tak že xy^iz nelze napumpovat dle bodu (3)

- dokažte, že jazyky nejsou regulární (popište celou úvahu!):

1. $L_1 = \{ww \mid w \in \{a, b\}^*\}$

2. $L_2 = \{a^p \mid p \text{ je prvočíslo}\}$

17 Homomorfismy jazyků

- **homomorfismus** na abecedě je funkce $h: \Sigma_1 \rightarrow \Sigma_2^*$, která přiřazuje každému symbolu $z \in \Sigma_1$ řetězec nad abecedou Σ_2
- h lze jednoduše definovat i pro řetězce, kde $h(w) = h(w_1) \dots h(w_n) \in \Sigma_2^*$ pro $w = w_1 \dots w_n \in \Sigma_1^*$ (zobrazíme zvlášť každý symbol v řetězci w)
- homomorfismus můžeme dále rozšířit jako operaci nad jazyky, kde pro jazyk L nad Σ_1^* je $h(L) = \{y \in \Sigma_2^* \mid x \in L \wedge h(x) = y\}$
- **inverzní homomorfismus** definujeme jako $h^{-1}(y) = \{x \in \Sigma_1^* \mid h(x) = y\}$ a analogicky pro jazyky $h^{-1}(L) = \{x \in \Sigma_1^* \mid h(x) \in L\}$
- pokud je jazyk L regulární, pak $h(L)$ je také regulární (to samé platí i pro $h^{-1}(L)$)
- mějme abecedy $\Sigma_1 = \{a, b, c, d\}$ a $\Sigma_2 = \{0, 1, 2\}$ a homomorfismus $h_1: \Sigma_1 \rightarrow \Sigma_2^*$, kde $h_1(a) = 02$, $h_1(b) = 21$, $h_1(c) = 10$, $h_1(d) = \varepsilon$; určete:

$$- h_1(adbccdda) =$$

$$- h_1^{-1}(0221) =$$

$$- h_1(L), \text{ kde } L = \{a^n b^* c^n \mid n \in \mathbb{N}\}$$

- mějme abecedu $\Sigma = \{a, b, c\}$ a homomorfismus $h_2: \Sigma \rightarrow \Sigma^*$, kde $h_2(a) = aa$, $h_2(b) = ba$, $h_2(c) = a$; určete:

$$- h_2^{-1}(aabaabaaba) =$$

$$- h_2(L), \text{ kde } L = \{a^p \mid p \text{ je prvočíslo}\}$$

$$- h_2^{-1}(L), \text{ kde } L = \{w \in \{a\}^* \mid |w| = 2n, \text{ pro } n \in \mathbb{N}\}$$

- pro $\Sigma_1 = \{0, 1\}$ a $\Sigma_2 = \{a\}$ zkuste najít homomorfismus $h: \Sigma_1 \rightarrow \Sigma_2^*$ takový, že pro všechna $w \in \Sigma_1^*$ platí $|h(w)| = 2^n$, kde $|w| = n$

18 Gramatiky a derivace

- gramatika je další způsob jak zapsat jazyk (**nejen regulární!** proto jsou gramatiky mnohem silnější nástroj než konečné automaty/RE)

- **Definice:** gramatika je reprezentována uspořádanou čtveřicí $\mathcal{G} = (N, \Sigma, P, S)$, kde:

1. N je množina neterminálů (značené jako velká písmena)
2. Σ je množina terminálů (symboly abecedy), platí $\Sigma \cap N = \emptyset$
3. P je množina pravidel ve tvaru $P \subseteq V^*NV^* \times V^*$, kde $V = N \cup \Sigma$
4. S je počáteční neterminál

- **pravidlo** $\alpha \rightarrow \beta$ chápeme tak, že při jeho aplikaci se α přepíše na β
- relaci přímého odvození $\gamma \Rightarrow_{\mathcal{G}} \delta$ chápeme jako aplikaci konkrétního pravidla na řetězec $\gamma = \eta\alpha\rho$, jehož výsledkem je řetězec $\eta\beta\rho$, neboli:

$$\eta\alpha\rho \Rightarrow_{\mathcal{G}} \eta\beta\rho$$

kde $\eta, \rho \in V^*$ a $\alpha \rightarrow \beta$ je právě aplikované pravidlo gramatiky \mathcal{G} (pokud je gramatika zřejmá z kontextu, pak můžeme označení \mathcal{G} u šipky vynechat)

- $\Rightarrow_{\mathcal{G}}^*$ umožňuje při odvození aplikovat libovolný počet libovolných pravidel (podobně $\Rightarrow_{\mathcal{G}}^k$ umožňuje přesně k aplikací, $\Rightarrow_{\mathcal{G}}^{\leq k}$ maximálně k aplikací atd.)
- **derivace** označuje proces, kdy na řetězec aplikujeme konečný počet kroků odvození
- \Rightarrow_{lm} značí nejlevější derivaci – přepisujeme vždy podle nejlevějšího neterminálu (analogicky \Rightarrow_{rm} je nejpravější derivace)
- **jazyk generovaný gramatikou** \mathcal{G} definujeme jako $L(\mathcal{G}) = \{w \in \Sigma^* \mid S \Rightarrow_{\mathcal{G}}^* w\}$ (řetězce v jazyku \mathcal{G} už neobsahují neterminály)
- mějme gramatiku \mathcal{G}_1 (ve zkrácené notaci – pravidla pro jeden neterminál jsou napsaná na jednom řádku oddělená pomocí '|'):

$$\begin{aligned} S &\rightarrow XSX \mid R \\ R &\rightarrow aTb \mid bTa \\ T &\rightarrow XTX \mid X \mid \varepsilon \\ X &\rightarrow a \mid b \end{aligned}$$

1. jaké jsou proměnné a terminály v \mathcal{G}_1 ?
2. napište tři řetězce z $L(\mathcal{G}_1)$
3. rozhodněte zda platí $S \Rightarrow^* aabba$
4. rozhodněte zda platí $S \Rightarrow^* babbab$
5. rozepište nejlevější derivaci $S \Rightarrow_{lm}^* ababab$

19 Regulární gramatika (typ 3)

- gramatika $\mathcal{G} = (N, \Sigma, P, S)$ je **regulární**, pokud jsou všechna pravidla ve tvaru:
 1. $A \rightarrow aB$ (neterminál generuje jeden terminál následovaný jedním neterminálem)
 2. $A \rightarrow a$ (neterminál generuje pouze jeden terminál)
 3. $S \rightarrow \varepsilon$, pokud se S nevyskytuje na pravé straně jakéhokoliv pravidla (pouze počáteční neterminál může generovat generovat prázdný řetězec)
- každá regulární gramatika \mathcal{G} popisuje nějaký regulární jazyk, a proto lze převést na NFA $A_{\mathcal{G}} = (N \cup \{q_f\}, \Sigma, \delta, S, F)$, kde:
 - $A_{\mathcal{G}}$ má jeden stav pro každý neterminál $B \in N$ a navíc jeden stav q_f
 - počáteční stav S odpovídá stavu počátečního neterminálu
 - F obsahuje stav q_f , a pokud \mathcal{G} obsahuje pravidlo $S \rightarrow \varepsilon$, pak i $S \in F$
 - pro každé pravidlo $A \rightarrow aB$ přidáme přechod $\delta(A, a) = B$ a pro pravidla ve tvaru $A \rightarrow a$ přidáme přechod $\delta(A, a) = q_f$
- navrhnete regulární gramatiky pro jazyky:

$$L_1 = \{w \in \{a, b\}^* \mid |w| \in \{0, 3, 5\}\}$$

$$L_2 = \{w \in \{a, b\}^* \mid w \text{ začíná a končí stejným řetězcem } aa \text{ nebo } bb\}$$

20 Bezkontextová gramatika (CFG, typ 2)

- gramatika $\mathcal{G} = (N, \Sigma, P, S)$ je bezkontextová (zkratka **CFG** z *context-free grammar*), pokud jsou všechna pravidla ve tvaru $A \rightarrow \alpha$, kde $\alpha \in (\Sigma \cup N)^*$, tedy neterminál generuje libovolný řetězec terminálů a neterminálů
- jazyk je **bezkontextový**, pokud jej generuje nějaká bezkontextová gramatika
- pokud je gramatika \mathcal{G} bezkontextová, tak to nemusí znamenat, že jazyk $L(\mathcal{G})$ není regulární – množina jazyků, které lze generovat pomocí CFG, je totiž nadmnožinou regulárních jazyků (a podmnožinou kontextově závislých jazyků)
- navrhnete bezkontextové gramatiky pro jazyky:

$$L_1 = \{w \in \{a, b, c\}^* \mid w = w^R\}$$

$$L_2 = \{w \in \{a, b, c\}^* \mid w \text{ obsahuje aspoň tři znaky } a\}$$

$$L_3 = \{a^{3n+2}b^{2n} \mid n \geq 1\}$$

$$L_4 = \{w \in \{p, \neg, \rightarrow, \vee, \wedge, (,), \text{true}, \text{false}\}^* \mid w \text{ je formule výrokové logiky}\}$$

příklad: " $\neg(p \wedge pp)$ " $\in L_4$ a " $(\vee p \wedge$ " $\notin L_4$ (p, pp, \dots jsou různé proměnné)

$$L_5 = \{a^n b^m \mid n \neq m\}$$

$L_6 = \{w \in \{0, \dots, 9, +, -, ., (,)\}^* \mid w \text{ je aritmetický výraz s desetinnými čísly}\}$
příklad: $"(-(-0.9))+(7.+45-3.415)" \in L_6$ a $"-00.9", "..", "5++5" \notin L_6$

- navrhnete bezkontextovou gramatiku pro minimalistický Lisp, který bude umět:
 - definovat proměnné pomocí **defvar**, kde název obsahuje pouze znaky **a-z**
 - definovat funkce s pevným počtem parametrů pomocí **defun** a s názvem opět jen ze znaků **a-z**
 - provádět aritmetické operace $+$, $-$, \cdot , $/$ na celých číslech
 - konstrukce **car**, **cdr**, **cons**, **list**, **if**, **let** a **lambda**
 - pro jednoduchost uvažujte, že každý výraz musí být oddělen whitespacem `␣`

21 Derivační stromy a víceznačnost CFG

- derivační strom je grafická reprezentace derivace (odvození) nějakého řetězce v CFG
- **Definice:** strom T nazveme derivačním stromem řetězce $\alpha \in (\Sigma \cup N)^*$ podle CFG $\mathcal{G} = (N, \Sigma, P, S)$, právě když platí:
 1. každý uzel je označen symbolem z $N \cup \Sigma \cup \{\varepsilon\}$, přičemž kořen je označen S
 2. nelistové (vniřní) uzly jsou označeny pouze neterminály
 3. pokud má nelistový uzel označení $A \in N$ a jeho potomci zleva doprava označení X_1, \dots, X_k , pak v \mathcal{G} existuje pravidlo $A \rightarrow X_1 \dots X_k$
 4. pokud je uzel označen ε , pak je list a nemá žádné sourozence
 5. zřetěžením listových uzlů dostaneme α , neboli α je *derivace* stromu T
- CFG \mathcal{G} je **víceznačná/nejednoznačná**, pokud existuje řetězec $S \Rightarrow^* \alpha$, který je derivací aspoň dvou různých derivačních stromů
- bezkontextový jazyk je **jednoznačný**, pokud existuje jednoznačná gramatika, která jej generuje (je rozdíl mezi jednoznačností gramatiky a jazyka, který generuje)
- pro některé bezkontextové jazyky neexistuje jednoznačná gramatika, takový jazyk je **dědičně víceznačný** (např. jazyk $\{a^i b^j c^k \mid i = j \vee j = k\}$)
- mějme víceznačnou gramatiku $\mathcal{G}_1 = (\{E\}, \{a, +, \times, (,)\}, P, E)$:

$$E \rightarrow E + E \mid E \times E \mid (E) \mid a$$

1. sestavte derivační strom k řetězci $((a))$
2. sestavte derivační strom k řetězci $(a \times (a + a))$
3. sestavte dva různé derivační stromy k jednomu řetězci

22 Chomského normální forma (CNF)

- **CNF** (zkratka z *Chomsky normal form*) je zjednodušený tvar CFG, který nám usnadní dokázat některá tvrzení (např. *pumping lemma* pro bezkontextové jazyky)
- do CNF lze převést každá CFG
- CFG $\mathcal{G} = (N, \Sigma, P, S)$ je v Chomského normální formě, pokud jsou všechna její pravidla ve tvaru:
 1. $A \rightarrow BC$ (neterminál generuje dva neterminály)
 2. $A \rightarrow a$ (neterminál generuje pouze jeden terminál)
 3. $S \rightarrow \varepsilon$, pokud se S nevyskytuje na pravé straně jakéhokoliv pravidla (pouze počáteční neterminál může generovat prázdný řetězec)
- mějme CFG $\mathcal{G}_1 = (N, \Sigma, P, S)$, kde $N = \{S, A, B, C, D, E, F\}$ a $\Sigma = \{a, b, c, d\}$:

$$S \rightarrow ASA \mid Ca \mid \varepsilon$$

$$A \rightarrow S \mid aB \mid Ca$$

$$B \rightarrow acB \mid bE$$

$$C \rightarrow acac \mid Da \mid \varepsilon$$

$$D \rightarrow dSS \mid E \mid aa$$

$$E \rightarrow cB \mid acE$$

$$F \rightarrow bD \mid ED$$

převod \mathcal{G}_1 do CNF provedeme v následujících krocích (záleží i na jejich pořadí):

1. odstraníme počáteční neterminál z pravé strany všech pravidel

- pokud se počáteční neterminál S vyskytuje na pravé straně nějakého pravidla, vytvoříme nový neterminál S' a přidáme nové pravidlo $S' \rightarrow S$
- zvolíme S' jako nový počáteční neterminál \mathcal{G}

2. odstraníme ε -pravidla

- odebereme libovolné pravidlo ve tvaru $A \rightarrow \varepsilon$, kde A není počáteční neterminál
- dále pro každé pravidlo $B \rightarrow \alpha$ obsahující A na pravé straně:
 - (a) přidáme do \mathcal{G} nová pravidla ve tvaru $B \rightarrow \beta$, kde β je řetězce α , ve kterém každý výskyt A může být nahrazen ε (např. pro pravidlo $B \rightarrow \alpha A \beta A \gamma \in P$ přidáme $B \rightarrow \alpha \beta A \gamma \mid \alpha A \beta \gamma \mid \alpha \beta \gamma$)
 - (b) pokud je pravidlo ve tvaru $B \rightarrow A$, pak přidáme nové pravidlo $B \rightarrow \varepsilon$, ale jen v případě, že jsme takové pravidlo již neodstranili (jinak by mohlo dojít k zacyklení)
- tyto kroky opakujeme, dokud neodstraníme všechna ε -pravidla (kromě $S \rightarrow \varepsilon$, kde S je startovní neterminál)

3. odstraníme jednoduchá pravidla

- pravidla ve tvaru $A \rightarrow B$, kde $B \in N$, nejsou v CNF povolena (můžeme vygenerovat vždy jen dva neterminály)
- každé takové pravidlo $A \rightarrow B$ odstraníme z \mathcal{G} , a poté přidáme nové pravidla $A \rightarrow \alpha_1 \mid \dots \mid \alpha_k$, kde $B \rightarrow \alpha_1 \mid \dots \mid \alpha_k$ jsou všechna zbývající pravidla s B na levé straně, které jsme při tomto procesu již neodstranili (jinak by mohlo dojít k zacyklení)

4. odstraníme neterminály, které nederivují terminální řetězec

- iterativně počítáme množinu neterminálů N_i , které generují terminální řetězec
- na začátku položíme $N_0 = \emptyset$ a postupně konstruujeme další $N_i = N_{i-1} \cup \{A \mid A \rightarrow \alpha, \alpha \in (N_{i-1} \cup \Sigma)^*\}$, výpočet končí pokud $N_i = N_{i-1}$
- nechť N_e je poslední vypočítaná množina N_i , pak z \mathcal{G} odstraníme všechny neterminály z $N \setminus N_e$ a pravidla, v nichž se tyto symboly vyskytují

5. odstraníme nedosažitelné symboly

- iterativně počítáme množinu symbolů $V_i \subset \Sigma \cup N$, které lze vygenerovat z počátečního neterminálu
- na začátku položíme $V_0 = \{S\}$ a postupně konstruujeme další $V_i = V_{i-1} \cup \{X \mid A \in V_{i-1} \wedge A \rightarrow \alpha X \beta \in P, \text{ pro nějaké } A \in N \text{ a } \alpha, \beta \in (N \cup \Sigma)^*\}$, výpočet končí pokud $V_i = V_{i-1}$
- nechť V_e je poslední vypočítaná množina V_i , pak z \mathcal{G} odstraníme všechny terminály i neterminály, které nejsou v V_e a pravidla, v nichž se tyto symboly vyskytují (formálně $N' = N \cap V_e, \Sigma' = \Sigma \cap V_e$ a $P' = P \cap (V_e \times V_e^*)$)

6. nakonec převedeme pravidla do správného tvaru

- pořád nám ještě zbývají pravidla, které generují kombinace terminálů a neterminálů, nebo více než dva symboly
- pro každý terminál $a \in \Sigma$ vytvoříme nový neterminál $\langle a \rangle$ a pravidlo $\langle a \rangle \rightarrow a$, které přidáme do \mathcal{G} jen v případě, že přidáme jiné pravidlo generující $\langle a \rangle$
- každé pravidlo ve tvaru $A \rightarrow x_1 \dots x_k$, kde $k \geq 3$ a x_1, \dots, x_k jsou buď terminály nebo neterminály, nahradíme novými pravidly:

$$A \rightarrow \hat{x}_1 \langle x_2 \dots x_k \rangle, \langle x_2 \dots x_k \rangle \rightarrow \hat{x}_2 \langle x_3 \dots x_k \rangle, \dots, \langle x_{k-1} x_k \rangle \rightarrow \hat{x}_{k-1} \hat{x}_k$$

kde $\langle x_2 \dots x_k \rangle, \dots, \langle x_{k-1} x_k \rangle$ jsou nové neterminály a \hat{x}_i je $\langle a \rangle$, pokud $x_i = a$ (tedy x_i je nějaký terminál), jinak \hat{x}_i je neterminál x_i

- každé pravidlo ve tvaru $A \rightarrow x_i x_j$, kde aspoň jeden z x_i, x_j je terminál, nahrad' novým pravidlem $A \rightarrow \hat{x}_i \hat{x}_j$ (\hat{x}_i je $\langle a \rangle$, pokud $x_i = a$, jinak x_i , analogicky \hat{x}_j)

23 Algoritmus CYK

- pro zadanou CFG $G = (N, \Sigma, P, S)$ v CNF a řetězec w testuje zda platí $S \Rightarrow^* w$
- časová složitost $O(n^3)$, kde n je délka slova $w = w_1 \dots w_n$; počítáme tabulku velikosti $O(n^2)$, přičemž výpočet jedné buňky tabulky trvá $O(n)$
- **Průběh algoritmu:**
 1. vytvoříme trojúhelníkovou tabulku (viz níže) o šířce a výšce n a pod každý sloupec i napíšeme napíšeme terminál w_i
poznámka: buňka v tabulce na indexu (i, j) odpovídá množině $X_{i,j}$, která obsahuje neterminály A , pro něž platí $A \Rightarrow^* w_i \dots w_j$
 2. inicializujeme spodní řádek tabulky: do každé množiny $X_{i,i}$ vložíme všechny neterminály A , pro které existuje v G pravidlo $A \rightarrow w_i$
 3. počítáme řadky od spodu tabulky: do $X_{i,j}$ vložíme neterminál A pokud existuje index k a pravidlo $A \rightarrow BC$ tak, že: $(i \leq k < j) \wedge B \in X_{i,k} \wedge C \in X_{k+1,j}$
poznámka: pro každé $A \rightarrow BC$ postupně testujeme náležení B a C do dvojic množin $(X_{i,i}, X_{i+1,j}), (X_{i,i+1}, X_{i+2,j}), \dots, (X_{i,j-1}, X_{j,j})$
 4. pokud na konci výpočtu $X_{1,n}$ obsahuje startovní neterminál, pak platí $S \Rightarrow^* w$
- pomocí algoritmu CYK ověřte, že $S \Rightarrow^*$ „time flies like an arrow“ v gramatice:

$S \rightarrow NP VP$
 $NP \rightarrow DET NP \mid NP NP \mid time \mid flies \mid arrow$
 $VP \rightarrow VP NP \mid VP PP \mid flies \mid like$
 $PP \rightarrow P NP$
 $DET \rightarrow an$
 $P \rightarrow like$

1, 5				
1, 4	2, 5			
1, 3	2, 4	3, 5		
1, 2	2, 3	3, 4	4, 5	
1, 1	2, 2	3, 3	4, 4	5, 5
<i>time</i>	<i>flies</i>	<i>like</i>	<i>an</i>	<i>arrow</i>

24 Zásobníkový automat (PDA)

- **PDA** (zkratka z *pushdown automaton*) je nedeterministický model podobný NFA, který má navíc přístup k potenciálně nekonečné paměti v podobě zásobníku
- motivace k PDA: chceme „zvýšit sílu“ NFA, aby rozponávaly bezkontextové jazyky
- **Definice:** PDA je reprezentován strukturou $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde:
 1. Q je konečná množina stavů
 2. Σ je vstupní abeceda
 3. Γ je zásobníková abeceda (platí $Z_0 \in \Gamma$)
 4. δ je přechodová funkce ve tvaru $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$, která stavu, vstupnímu symbolu a symbolu na vrcholu zásobníku přiřadí množinu dvojic skládajících se ze stavu a řetězce zásobníkových symbolů
 5. q_0 je počáteční stav (platí pro něj $q_0 \in Q$)
 6. Z_0 je symbol, který je jako jediný v zásobníku na začátku výpočtu
 7. F je množina koncových/akceptujících stavů (platí $F \subseteq Q$)
- pro jednoduchost povolíme i přechody, které nejsou podmíněné vrcholem zásobníku, tedy například přechod $\delta(q, \varepsilon, \varepsilon) = \{(p, \varepsilon)\}$ nečte nic ze vstupu ani zásobníku
- stav výpočtu PDA popisuje **konfigurace**, což je trojice $\langle q, w, \alpha \rangle \in Q \times \Sigma^* \times \Gamma^*$ obsahující aktuální stav, nezpracovanou část vstupního řetězce a stav zásobníku
- přechodová relace $X \vdash Y$ znamená, že z konfigurace X je dosažitelná konfigurace Y v jednom kroku (\vdash^* značí rozšíření na libovolný počet kroků)
- jazyk PDA lze definovat dvěma způsoby:

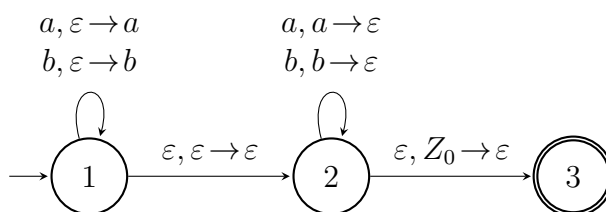
1. pomocí koncových stavů (podobně jako u NFA)

$$L(P) = \{w \in \Sigma^* \mid \langle q_0, w, Z_0 \rangle \vdash^* \langle q_f, \varepsilon, \alpha \rangle, \text{ kde } q_f \in F \text{ a } \alpha \in \Gamma^*\}$$

2. prázdným zásobníkem

$$N(P) = \{w \in \Sigma^* \mid \langle q_0, w, Z_0 \rangle \vdash^* \langle q, \varepsilon, \varepsilon \rangle, \text{ kde } q \in Q\}$$

- mějme PDA P_1 (popisky u přechodů jsou ve tvaru $\Sigma, \Gamma \rightarrow \Gamma^*$):



1. je P_1 deterministický?
2. jaké jsou jazyky $L(P_1)$ a $N(P_1)$?
3. jaký může být stav zásobníku po přečtení *baaaa* ze vstupu?
4. rozepište kroky $\langle 1, abba, Z_0 \rangle \vdash^* \langle 3, \varepsilon, \varepsilon \rangle$

25 Návrh PDA

$$L_1 = \{a^i b^j c^j d^i \mid i, j \geq 0\}$$

$$L_2 = \{a^i b^j \mid 1 \leq j \leq i \leq 2j\}$$

$$L_3 = \{a^i b^j c^k \mid i = j \text{ nebo } j = k\}$$

$$L_4 = \{w \in \{a, b\}^* \mid \#_a(w) \neq \#_b(w), \text{ tedy počet } a \text{ a } b \text{ není stejný}\}$$

26 Převod CFG na PDA

- PDA, stejně jako CFG, rozpoznávají třídu bezkontextových jazyků, a proto můžeme tyto modely mezi sebou převádět
- **Konstrukce:** mějme gramatiku $\mathcal{G} = (N, \Sigma, P, S)$, pak PDA $A = (\{q\}, \Sigma, N \cup \Sigma, \delta, q, S, \emptyset)$ takový, že $L(G) = N(A)$, sestrojíme v těchto krocích:
 1. A má pouze jeden stav, který je zároveň počáteční (nemá koncové stavy)
 2. zásobníková abeceda A obsahuje všechny terminály a neterminály z \mathcal{G}
 3. počáteční zásobníkový symbol je počáteční neterminál S
 4. pro každé pravidlo $A \rightarrow \alpha$ přidáme přechod $(q, \alpha) \in \delta(q, \varepsilon, A)$
 5. dále pro každý terminál $a \in \Sigma$ přidáme přechod $\delta(q, a, a) = \{(q, \varepsilon)\}$
- PDA A v této konstrukci simuluje nejlevější derivaci vstupního řetězce v \mathcal{G} na svém zásobníku, přičemž v každém kroku nedeterministicky vybere jeden z přechodů:
 - přechody v bodu 4. expandují pravidla \mathcal{G} na zásobník
 - přechody v bodu 5. srovnávají vygenerované terminály se vstupem
- mějme gramatiku $\mathcal{G}_1 = (\{E\}, \{a, +, \times, (,)\}, P, E)$:

$$E \rightarrow E + E \mid E \times E \mid (E) \mid a$$

1. převed'te \mathcal{G}_1 na PDA
2. rozepište kroky $\langle q, (a \times a), E \rangle \vdash^* \langle q, \varepsilon, \varepsilon \rangle$

27 Gramatiky typu 0 a 1

- jazyky, které generují gramatiky typu 0 a 1, jsou na rámec tohoto kurzu, více se o nich dozvíte v předmětu *Vyčíslitelnost a složitost*
- gramatika \mathcal{G} je **kontextová** (CSG nebo typ 1), pokud jsou všechna pravidla ve tvaru:
 1. $\alpha A \beta \rightarrow \alpha \gamma \beta$, kde $A \in N$, $\gamma \neq \varepsilon$ a $\alpha, \gamma, \beta \in (\Sigma \cup N)^*$ (levá strana pravidla může obsahovat společně s přepisovaným neterminálem A i další symboly, které podmiňují aplikování daného pravidla, ale samy nemohou být přepsány)
 2. $S \rightarrow \varepsilon$, pokud se S nevyskytuje na pravé straně jakéhokoliv pravidla
- mějme mějme gramatiku $\mathcal{G}_1 = (\{S, A, B, C, W, Z\}, \{a, b, c\}, P, S)$:

$$\begin{array}{ll}
 S \rightarrow aBC \mid aSBC & aB \rightarrow ab \\
 CB \rightarrow CZ & bB \rightarrow bb \\
 CZ \rightarrow WZ & bC \rightarrow bc \\
 WZ \rightarrow WC & cC \rightarrow cc \\
 WC \rightarrow BC &
 \end{array}$$

1. jaký jazyk generuje gramatika \mathcal{G}_1 ?
2. rozepište celou derivaci řetězce $aabbcc$

- pokud na pravidla neklademe žádné omezení, a tedy povolujeme libovolné pravidlo $\alpha \rightarrow \beta$ pro $\alpha, \beta \in (\Sigma \cup N)^*$, pak taková gramatika \mathcal{G} je **bez omezení** (typ 0)
- mějme mějme gramatiku $\mathcal{G}_2 = (\{S, A, B, C, D, E\}, \{a, b\}, P, S)$:

$$\begin{array}{ll}
 S \rightarrow ABC & Db \rightarrow bD \\
 AB \rightarrow aAD \mid bAE \mid \varepsilon & Ea \rightarrow aE \\
 DC \rightarrow BaC & Eb \rightarrow bE \\
 EC \rightarrow BbC & C \rightarrow \varepsilon \\
 Da \rightarrow aD & aB \rightarrow Ba \\
 & bB \rightarrow Bb
 \end{array}$$

1. jaký jazyk generuje gramatika \mathcal{G}_2 ?
2. rozepište celou derivaci řetězce $baabaa$

28 Pumping lemma pro bezkontextové jazyky

- **Pumping lemma pro CFL:** nechť L je bezkontextový jazyk, pak existuje číslo $n \in \mathbb{N}$ tak, že každý řetězec $s \in L$ délky aspoň n lze rozdělit na pět částí $s = uvxyz$, které splňují:

1. $|vy| > 0$
2. $|vxy| \leq n$
3. $uv^i xy^i z \in L$ pro všechna $i \geq 0$

- **intuice:** L je bezkontextový, proto existuje PDA P , který jej rozpoznává; pokud máme dostatečně dlouhý řetězec $s \in L$, pak se v něm musí nějaký podřetězec v opakovat, přičemž P si na zásobník uloží výskyty tohoto podřetězce, a později je může srovnat s výskyty jiného podřetězce y (proto mezi počtem opakování v a y může být závislost)
- zásobník umožňuje pouze LIFO přístup, a proto P nemůže zároveň srovnávat výskyty více jak jedné dvojice; navíc řetězce ze zásobníku čteme v opačném pořadí, než byly uloženy (proto L_1 není CFL, přitom $\{ww^R \mid w \in \{a, b\}^*\}$ je CFL)
- dokažte, že následující jazyky nejsou bezkontextové:

$$L_1 = \{ww \mid w \in \{a, b\}^*\}$$

$$L_2 = \{a^i \# a^j \# a^k \mid 0 < i < j < k\}$$

29 Proč neplatí opačná implikace v pumping lemma?

- pumping lemma je tvrzení ve tvaru implikace: „pokud je jazyk regulární, případně bezkontextový, pak splňuje nějaké podmínky“
- lze ukázat, že opačná implikace neplatí (tedy neplatí tvrzení: „pokud platí podmínky v lemmatu, pak je jazyk regulární/bezkontextový“)
- pro jednoduchost uvažujme pumping lemma pro regulární jazyky, ale protipříklad můžeme najít i pro bezkontextové jazyky
- chceme najít jazyk, který splňuje pumping lemma, ale přitom není regulární
- mějme abecedu $\Sigma = \{a, b\}$, nějaký jazyk $L \subseteq \{b\}^*$ a jazyk $L_p = (\{a\}^+ \cdot L) \cup \{b\}^*$; dokažte následující tvrzení:

1. L_p je regulární právě tehdy, když $(\{a\}^+ \cdot L)$ je regulární

2. $(\{a\}^+ \cdot L)$ je regulární právě tehdy, když L je regulární

3. pokud L není regulární, pak L_p také není regulární a zároveň splňuje podmínky pumping lemma

30 Domácí úkoly

1. mějme $L = \{ab, ba\}$, $M = \{aa, ab\}$ a $N = \{a, b\}$; vypište prvky jazyků:
- (a) $L \cdot (M \cup N)$
(b) $L \cdot (M \cdot N)$
2. mějme abecedu $\Sigma = \{a, b\}$ a jazyky $A = \{a\}$ a $B = \{b\}$; popište obsah nekonečných jazyků:

- (a) $B \cdot A^*$
(b) $A^* \cup B^+$
(c) $(A^* \cup B^*)^+$

poznámka: unární operace mají vyšší prioritu

3. zamyslete se, zda obecně platí vztahy a dokažte své tvrzení:

- (a) $L \cdot (M \cup N) = (L \cdot M) \cup (L \cdot N)$
(b) $L \cdot (M \cdot N) = (L \cdot M) \cdot N$

4. pro jazyky nad abecedou $\Sigma = \{0, 1, 2\}$ navrhňte RE:

- (a) $\{s \mid s \text{ neobsahuje podřetězec } 01\}$
(b) $\{s \mid s \text{ má lichý počet } 0\}$

5. s pomocí libovolného regex editoru* si vyzkoušejte napsat *rozšířené*** RE pro:

- (a) celá čísla a desetinná čísla:

✓ 10	✗ --10 (více než jedno znaménko)
✓ -0.5	✗ 0. (za tečkou musí následovat číslo)
✓ .5	✗ 10.2ab (nepovolené znaky)

- (b) emailovou adresu (zde není nutné se držet přesné syntaxe adres, stačí rozpoznat uvedené příklady):

✓ 1234567890@example.com	✗ email@ (chybí doména)
✓ ____@example.com	✗ emailexample.com (chybí @)
✓ email@example.museum	✗ em@ail@example.com (více jak jeden @)
✓ email@example.co.jp	✗ my email@example.com (znak mezery)

- (c) IPv4 adresu:

✓ 172.15.255.255	✗ 256.0.0.0 (256 je mimo rozsah)
✓ 172.32.0.0	✗ 1.0.1..255. (přebývajících tečky)

* například <https://regex101.com/> podporuje několik standardů pro zápis RE

** běžně používané regulární výrazy toho umožňují více, než uvádí formální definice

6. mějme abecedu $\Sigma = \{a, b\}$ a dva jazyky $A = \{a\}$, $B = \{b\}$; navrhňte DFA rozponávající jazyky:

- (a) $\Sigma^* \setminus (A^* \cdot B^*)$

(b) $A^+ \cdot B^+ \cdot A^*$

7. popište (slovně nebo pseudokódem) algoritmus, který o DFA A rozhodne, zda jeho jazyk $L(A)$ je konečná množina
8. popište průběh algoritmu, který z DFA odstraní nedosažitelné stavy
9. zkuste si implementovat konečný deterministický automat s rozšířenou přechodovou funkcí ve vašem oblíbeném programovacím jazyce
10. pomocí součinnové konstrukce sestrojte DFA pro jazyk:

$$\{s \in \{a, b\}^* \mid s \text{ má lichý počet } a\} \cup \{s \in \{a, b\}^* \mid \text{každé } a \text{ je následováno alespoň dvěma } b\}$$

11. determinizujte NFA A_1 (po odstranění ε -přechodů) ze Sekce 12
12. odstraňte ε -přechody z ε -NFA zadaného tabulkou a poté jej determinizujte:

δ_E	a	b	ε
$\rightarrow 1$	$\{2\}$	\emptyset	$\{2, 3\}$
2	$\{4\}$	$\{2, 3\}$	\emptyset
3	$\{5\}$	\emptyset	\emptyset
4	\emptyset	$\{5\}$	$\{5\}$
$5\star$	$\{5\}$	$\{3\}$	\emptyset

13. zamyslete se, jak by mohla vypadat obecná konstrukce DFA, který pro zadaný jazyk L rozpoznává jeho reverzní jazyk L^R
14. pokud je L regulární jazyk, pak jazyk sufixů všech slov daného jazyka $Sfx(L) = \{z \mid \exists x \in L: x = yz\}$ je také regulární (a proto existuje DFA, který jej rozpoznává); zamyslete se, jak by mohla vypadat obecná konstrukce DFA rozpoznávající $Sfx(L)$ pro zadaný jazyk L
15. minimalizujte DFA zadaný tabulkou a výsledný automat nakreslete:

δ	a	b
$\rightarrow 1$	2	4
2	2	2
$3\star$	3	2
4	7	9
5	2	2
$6\star$	3	5
7	5	8
8	7	9
$9\star$	6	8

16. pomocí *pumping lemma* dokažte, že jazyk není regulární:

$$L = \{s \in \{(\,,\,)\}^* \mid s \text{ je správně uzávorkovaný výraz}\}$$

– správně uzávorkovaný je například $()()$ nebo $((()))()$, přičemž $((()$ a $)()$ nejsou

17. pro $\Sigma_1 = \{a, b, c, d, e\}$ a $\Sigma_2 = \{0, 1\}$ najděte homomorfismus $h: \Sigma_1 \rightarrow \Sigma_2^*$ takový, že pro všechna $w \in \Sigma_1^*$ platí $h^{-1}(h(w)) = \{w\}$
18. jaký je jazyk generovaný gramatikou \mathcal{G}_1 ze Sekce 18?
19. navrhněte regulární gramatiku pro jazyk $\{w \in \{a, b, c\}^* \mid w \text{ neobsahuje podřetězec } abc\}$
20. najděte předpis, který pro zadané $n \in \mathbb{N}$ vytvoří NFA s n stavy, a jehož determinizace má 2^n dosažitelných stavů (takových předpisů automatů existuje více)
21. navrhněte gramatiku pro jazyk $L = \{w \in \{a, b\}^* \mid \#_a(w) \neq \#_b(w)\}$, tedy jazyk slov, kde počet a a b není stejný
22. dokažte, že třída bezkontextových jazyků je uzavřená na operace regulárních výrazů: uzávěr $*$, sjednocení a zřetězní
23. mějme zadanou gramatiku $\mathcal{G} = (\{E, T, F\}, \{a, +, \times, (,)\}, P, E)$,

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T \times F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}$$

sestavte derivační stromy k řetězcům:

- (a) $((a))$
- (b) $a + a \times a$
- (c) $(a + a) \times a$

24. nechť \mathcal{G} je libovolná gramatika v CNF a $w \in L(\mathcal{G}) \setminus \{\varepsilon\}$ je neprázdný řetězec; jaká je minimální a maximální délka odvození řetězce w v \mathcal{G} (počet provedených derivací)? pro své tvrzení uveďte důkaz
25. popište algoritmus, který pro CFG \mathcal{G} rozhodne zda $L(\mathcal{G})$ je konečná množina
26. DFA $A = (Q, \Sigma, \delta, q_0, F)$ je *synchronizující*, pokud existuje $w \in L(A)$ a $q \in Q$ takové, že pro všechny $p \in Q$ platí $\delta(p, w) = q$. Můžeme o libovolném DFA rozhodnout, zda je synchronizující?
27. dokažte, že jazyky jsou bezkontextové:

$$\begin{aligned} L_1 &= \{x_1 \# x_2 \# \dots \# x_k \mid k \geq 2, x_i \in \{a, b\}^*, \text{ a } x_i = x_j^R \text{ pro nějaké } i \neq j\} \\ L_2 &= \{x \# y \mid x, y \in \{a, b\}^+, |x| \leq |y| \wedge x \notin Pfx(y)\} \end{aligned}$$

poznámka: symbol $\#$ je v tomto případě další znak abecedy – oddělovač

28. zamyslete se jak se změní množina jazyků rozpoznávaných pomocí PDA, pokud model upravíme tak, že:
 - (a) nový model má pouze konečný zásobník, tj. mějme danou konstantu k , pak nemůžeme provést přechod, po němž by obsah zásobníku byl větší než k
 - (b) nový model má k dispozici dva zásobníky, přičemž každý přechod v δ může manipulovat (zapisovat i číst) vždy jen s jedním zásobníkem

29. mějme gramatiku $\mathcal{G} = (\{S, A, B, C, D, E\}, \{a\}, P, S)$:

$$S \rightarrow ACaB$$

$$Ca \rightarrow aaC$$

$$CB \rightarrow DB \mid E$$

$$aD \rightarrow Da$$

$$AD \rightarrow AC$$

$$aE \rightarrow Ea$$

$$AE \rightarrow \varepsilon$$

- (a) rozhodněte jakého je gramatika typu a odpověď zdůvodněte
- (b) rozhodněte zda platí $S \Rightarrow^* aaaa$ (pokud ano, rozepište celou derivaci)
- (c) jaký jazyk gramatika \mathcal{G} generuje?

30. mějme jazyk $L = \{a^i b^j c^j d^j \mid i, j \in \mathbb{N}, i \geq 1\} \cup \{b^j c^k d^\ell \mid j, k, \ell \in \mathbb{N}\}$; dokažte tvrzení:

- (a) L není bezkontextový
- (b) L splňuje podmínky pumping lemma pro bezkontextové jazyky