1. Úvod do formálních jazyků

Formální jazyky a automaty

Jiří Balun

Obsah

1 Úvod
- Motivace
- Základní pojmy: Abeceda, řetězec a jazyk
- Operace s řetězci a jazyky

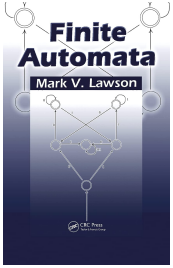
2 Deterministický konečný automat (DFA)
- Popis DFA
- Výpočet DFA
- Jazyk DFA

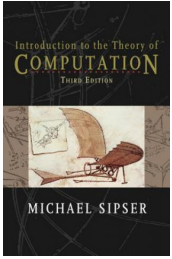
Organizační záležitosti

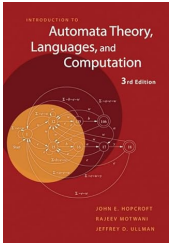
- konzultační hodiny (kancelář 5.044):
  - Úterý 13:30-14:30
  - Pátek 13:00-14:00
  - případně v jiný čas po domluvě emailem
- cvičení:
  - budeme řešit úkoly z materiálů dostupných na mém webu
- zápočet:
  - zápočet se skládá ze dvou částí, tj. dvou zápočtových testů
  - z každé části/testu potřebujete získat alespoň 50% bodů
  - každý test si budete moci jednou opravit
  - testy se budou psát v rámci přednášek
- zkouška:
  - bude upřesněna později, pravděpodobně ústní

Literatura

- Formální jazyky a automaty I (Černá, Křetínský, Kučera, 2006)
  - odkaz: https://is.muni.cz/elportal/?id=703389
- Finite Automata (Lawson, 2003)
- Introduction to the Theory of Computation (Sipser, 2012)
- Introduction to Automata Theory, Languages, and Computation (Hopcroft, Motwani, Ullman, 2006)







Motivace

Předmět Formální jazyky a automaty

- jeden ze základních oborů teoretické informatiky
- součástí většiny informaticky zaměřených oborů (MFF UK, ČVUT, VUT,...)
- slouží jako úvod pro další předměty jako Vyčíslitelnost a Složitost
- možné jako téma bakalářské práce

Proč se zabývat jazyky?

- na data a programy můžeme pohlížet jako množiny řetězců
  - data a programy v počítači jsou reprezentovány pomocí posloupností symbolů 0 a 1
  - přirozená motivace studovat jejich vlastnosti a jak s nimi pracovat
- studium výpočetních modelů
  - jak silný model potřebujeme k řešení různých problémů
  - začneme u těch jednoduchých a postupně budeme přecházet k silnějším
- modely, které budeme studovat mají spoustu reálných aplikací
  - regulární výrazy, překladače, modelování diskrétních systémů,...

Abeceda

Intuice: abecedu budeme chápat v jejím přirozeném významu, tedy jako soubor znaků, které mohou být použity k vytvoření řetězců

Definice

Abeceda Σ je libovolná konečná neprázdná množina.

- prvky abecedy se nazývají **znaky**, případně **symboly**
- spousta příkladů ze života:
  - latinské písmo $\{a, b, c, \ldots, z\}$, japonské písmo $\{$ア, キ, ラ$, \ldots\}$, atd.
  - binární abecedy $\{0, 1\}$ nebo $\{\texttt{true}, \texttt{false}\}$
  - znaková sada ASCII $\{0, \ldots, 9, a, \ldots, z, A, \ldots, Z, \texttt{NULL}, \texttt{LF}, \ldots\}$
  - množina klíčových slov jazyka C: $\{\texttt{if}, \texttt{for}, \texttt{while}, \ldots\}$
  - symboly výrokové logiky $\{\neg, \vee, \wedge, (,), \varphi, \psi \ldots\}$
  - a mnoho dalších příkladů...
- každou abecedu lze zakódovat pomocí binarní abecedy

Řetězec

Definice

Řetězec $w = w_1 \dots w_n$ nad abecedou Σ je libovolná konečná posloupnost znaků této abecedy, tj. $w \in \Sigma^*$ (platí $w_i \in \Sigma$ pro všechna $i = 1, \dots, n$).

- délka řetězce $w = w_1 \dots w_n$ se značí $|w| = n$
- **prázdný řetězec** se značí ε a platí pro něj $|\varepsilon| = 0$
- Σ^* označuje množinu všech řetězců nad abecedou Σ
- $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$ označuje množinu všech neprázdných řetězců nad Σ

Příklad

Mějme binární abecedu $\Sigma = \{0, 1\}$, pak:

1. pro řetězec $w = 101010 \in \Sigma^*$ je $|w| = 6$
2. $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \dots\}$
3. $\Sigma^+ = \{0, 1, 00, 01, 10, 11, 000, \dots\}$

Poznámka: množiny řetězců budeme psát v tzv. **lexikografickém uspořádání**

Jazyk

Definice

Jazyk nad abecedou Σ je libovolná podmnožina Σ^*.

- **základní problém:** umět rozhodnout zda řetězec patří do daného jazyka
- jazyky budeme dělit do tříd podle toho jak silný model potřebujeme pro jejich rozhodování/generování

Základní dělení

1. konečné jazyky:
   - například množina slov českého jazyka (přibližně 250 000 slov)
   - je jednoduché ověřit zda nějaký řetězec do takového jazyka patří
   - reprezentace pomocí výčtu prvků
   - z našeho pohledu triviální a nezajímavé
2. nekonečné jazyky:
   - například množina všech syntakticky korektních programů jazyka C
   - ověřit náležení do takového jazyka může být velice těžké nebo dokonce nemožné
   - reprezentace pomocí nějakého verifikátoru/generátoru (pokud je to možné)

Příklady jazyků

Příklad

Trivialní jazyky nad libovolnou abecedou Σ jsou $\emptyset$ (nejmenší) a Σ^* (největší).

Příklad

Jazyk L_1 korektně zapsaných emailových adres nad abecedou ASCII, kde například

$$jiri.balun@upol.cz \in L_1 \qquad jiri@balun@upol@cz \notin L_1$$

lze jednoduše verifikovat pomocí tzv. konečných automatů (viz později).

Příklad

Jazyk L_2 druhých mocnim nad abecedou $\Sigma = \{0, 1, \ldots, 9\}$ definován jako

$$L_2 = \{w \in \Sigma^* \mid \text{číselná hodnota } w \text{ je druhá mocnina pro nějaké } n \in \mathbb{N}\},$$

kde například $25 \in L_2$ a $42 \notin L_2$, nelze rozhodovat bez výpočtu druhé mocniny/odmocniny – už potřebujeme mnohem silnější model.

Operace s řetězci

Definice

Pro libovolné řetězce $x = x_1 \dots x_n$ a $y = y_1 \dots y_m$ definujeme

- binární operaci **zřetězení** $x \cdot y$ (nebo zkráceně xy):

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

- unární operaci n-**té mocniny** x^n:

$$x^n = \begin{cases} \varepsilon & \text{pro } n = 0 \\ x \cdot x^{n-1} & \text{pro } n > 0 \end{cases}$$

Příklad

Pro řetězce $x = lorem$, $y = ipsum$ a $z = dolor$

- $x \cdot y \cdot z = loremipsumdolor$
- $x^3 = loremloremlorem$

Operace s jazyky

Definice

Pro libovolné jazyky L, L_1 a L_2 definujeme

- klasické množinové operace průnik, sjednocení, doplněk (vzhledem k Σ^*), ...
- binární operaci **zřetězení jazyků** $L_1 \cdot L_2$:

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

- unární operaci n-**té mocniny** L^n:

$$L^n = \begin{cases} \{\varepsilon\} & \text{pro } n = 0 \\ L \cdot L^{n-1} & \text{pro } n > 0 \end{cases}$$

- unární operaci **Kleeneho uzávěr** L^* a jeho variantu bez nulté mocniny L^+:

$$L^* = \bigcup_{n=0}^{\infty} L^n \qquad\qquad L^+ = \bigcup_{n=1}^{\infty} L^n$$

Operace s jazyky – příklady

Příklad

Pro jazyky $L_1 = \{0, 00\}$, $L_2 = \{1, 11\}$ a $L_3 = \{10\}$ máme

$$\begin{aligned} L_1 \cdot L_2 &= \{01, 001, 011, 0011\} \\ L_2 \cdot L_1 &= \{10, 100, 110, 1100\} \\ L_1^3 &= \{0^3, 0^4, 0^5, 0^6\} \\ L_1^* &= \{\varepsilon, 0, 00, 000, \ldots\} = \{0\}^* \\ L_3^+ &= \{10, 1010, 101010, \ldots\} \end{aligned}$$

Příklad

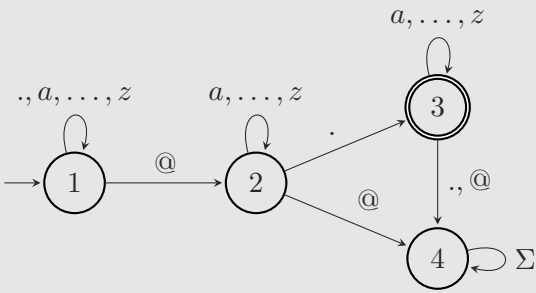
Mějme jazyk L_4 nad abecedou $\Sigma = \{a\}$ definován jako $L_4 = \{a^p \mid p \text{ je prvočíslo}\}$, pak

$$\begin{aligned} L_4 \cdot \{aa\} &= \{a^{p+2} \mid p \text{ je prvočíslo}\} \\ L_4^2 &= \{a^\ell \mid \ell \text{ je násobkem dvou prvočísel}\} \end{aligned}$$

Motivační příklad – první návrh

Příklad

Chceme ověřovat zda řetězce nad $\Sigma = \{., @, a, \ldots, z\}$ obsahují emailovou adresu



Grafická reprezentace

- uzly reprezentují stavy, ve kterých se můžeme během výpočtu nacházet
- šipka u stavu 1 označuje počáteční stav a zdvojené stavy jsou akceptující
- hrany a jejich popisy reprezentují přechody mezi stavy – definují průběh výpočtu

Deterministický konečný automat

Intuice: automat je jednoduchý abstraktní výpočetní model (algoritmus), který realizuje rozhodovací problém náležení do jazyka

Definice

Deterministický konečný automat (zkráceně DFA z *deterministic finite automaton*) je reprezentován uspořádanou pěticí $A = (Q, \Sigma, \delta, q_0, F)$, kde:

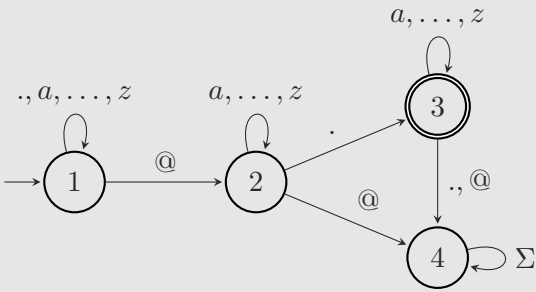
1. Q je konečná množina stavů (proto název *konečný* automat)
2. Σ je abeceda
3. δ je přechodová funkce ve tvaru $\delta : Q \times \Sigma \to Q$, která stavu a symbolu přiřadí stav
4. q_0 je počáteční stav (platí $q_0 \in Q$)
5. F je množina koncových/akceptujících stavů (platí $F \subseteq Q$),

- přechodová funkce δ je **úplná** (je definovaná pro každou dvojici stavu a znaku)
  - nestane se, že by automat nevěděl jak pokračovat ve výpočtu
- přechodová funkce δ je **deterministická** (výsledkem je vždy jeden stav)
  - máme zaručeno, že automat je vždy právě v jednom stavu

Motivační příklad – popis automatu

Příklad

Chceme ověřovat zda řetězce nad $\Sigma = \{., @, a, \ldots, z\}$ obsahují emailovou adresu



Zkonstruujeme DFA $A_1 = (Q, \Sigma, \delta, q_0, F)$ tak, že popíšeme jeho jednotlivé části:

- $Q = \{1, 2, 3, 4\}$, $\Sigma = \{., @, a, \ldots, z\}$, $q_0 = 1$ a $F = \{3\}$
- přechodovou funkci δ lze reprezentovat jako relaci $\delta \subseteq Q \times \Sigma \times Q$:
  - $\{(1, ., 1), (1, a, 1), \ldots, (1, z, 1), (1, @, 2), (2, a, 2), \ldots, (2, z, 2), (2, ., 3), (2, @, 4), \ldots\}$

Výpočet DFA

Intuice: výpočet si lze představit jako průchod grafem automatu, v němž hrany volíme podle přechodové funkce (pro každý řetězec existuje právě jedna cesta z každého stavu)

Definice

Konfigurace je dvojice stavu a řetězce $\langle q, w \rangle \in Q \times \Sigma^*$, která popisuje aktuální stav DFA.

Průběh výpočtu DFA

- výpočet začíná v konfiguraci $\langle q_0, w \rangle$ s řetězcem $w = w_1 \dots w_n \in \Sigma^*$ na vstupu
- v každém kroku DFA, který je v konfiguraci $\langle q, w_i \dots w_n \rangle$, zpracuje první znak vstupu
  - přečte první symbol w_i a ze vstupu jej odebereme
  - na základě přechodové funkce se vypočítá nový stav $p = \delta(q, w_i)$
  - DFA přejde do nové konfigurace $\langle p, w_{i+1} \dots w_n \rangle$
- výpočet končí v konfiguraci s prázdným vstupem, tj. ve tvaru $\langle q, \varepsilon \rangle$
  - pokud $q \in F$ je koncový stav, pak daný DFA **přijímá** w
  - v opačném případě $q \notin F$ daný DFA **zamítá** w

Rozšířená přechodová funkce

Definice

Rozšířenou přechodovou funkci $\hat{\delta}\colon Q \times \Sigma^* \to Q$ definujeme pro dvojici stavu $q \in Q$ a řetězce $s = s_1 \dots s_n \in \Sigma^*$ následovně:

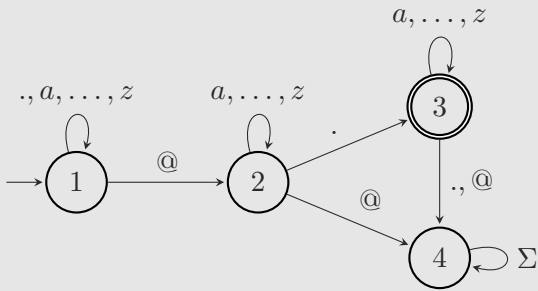
$$\hat{\delta}(q, s) = \begin{cases} q & \text{pro } s = \varepsilon \\ \hat{\delta}(\delta(q, s_1), s_2 \dots s_n) & \text{pro } s \neq \varepsilon \end{cases}$$

- jedná se pouze o syntaktický cukr, který nám usnadní zápis
  - například místo $\delta(\delta(\delta(q, s_1), s_2), s_3)$ můžeme psát jen $\hat{\delta}(q, s_1 s_2 s_3)$
- většinou nebudeme rozlišovat mezi δ a $\hat{\delta}$, tj. budeme psát rovnou $\delta(q, s_1 s_2 s_3)$

Motivační příklad – průběh výpočtu

Příklad

Výpočet DFA $A_1 = (\{1,2,3,4\}, \{., @, a, \ldots, z\}, \delta, 1, \{3\})$, který ověřuje emailové adresy

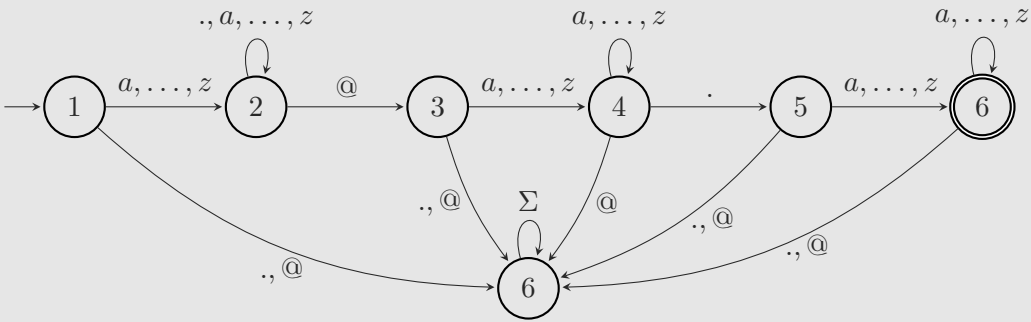


- A_1 zamítá řetězec $@b@$, protože $\delta(\delta(\delta(1, @), b), @) = \delta(\delta(2, b), @) = \delta(2, @) = 4 \notin F$
- A_1 přijímá řetězec $jiri.balun@upol.cz$, protože $\hat{\delta}(1, jiri.balun@upol.cz) = 3 \in F$
- A_1 přijímá i řetězec $@.$, protože $\hat{\delta}(1, @.) = 3 \in F$ ... to ale není emailová adresa

Motivační příklad – oprava

Příklad

Nový DFA $A_2 = (\{1,2,3,4\}, \{., @, a, \ldots, z\}, \delta, 1, \{3\})$ ověřující emailové adresy



- nyní A_2 zamítá řetězec .@, protože $\hat{\delta}(1, .@) = 7 \notin F$
- A_2 vyžaduje aby každá část adresy (lokální i domény) byla neprázdná

Regulární jazyk

Definice

Jazyk DFA $A = (Q, \Sigma, \delta, q_0, F)$ definujeme jako $L(A) = \{s \mid \hat{\delta}(q_0, s) \in F\}$.

- říkáme, že A **rozhoduje/rozpoznává** jazyk $L(A)$
- třída jazyků, které jsou rozhodovány pomocí DFA jsou tzv. **regulární jazyky**
  - každý DFA rozpoznává nějaký regulární jazyk
  - každý regulární jazyk rozpoznává nějaký DFA
- DFA lze tedy chápat jako formalismus pro definování regulárních jazyků
  - je to matematický model – nic neříká o konkrétní implementaci
  - implementovaný ve všech současných programovacích jazycích
  - verifikaci regulárních jazyků můžeme provádět na počítači
- z uvedených příkladů:
  - každý konečný jazyk je regulární
  - jazyk emailových adres je regulární (sestrojili jsme DFA, který jej rozhoduje)
  - jazyk druhých mocnim nebo $\{a^p \mid p \text{ je prvočíslo}\}$ nejsou regulární (ukážeme později)

Neúplná přechodová funkce

- DFA můžeme definovat i s neúplnou přechodovou funkcí
  - v přechodové funkci nemusí být všechny přechody definovány
  - pokud má automat provést nedefinovaný přechod, okamžitě daný řetězec zamítne

Věta

Ke každému DFA A existuje DFA A' s úplnou přechodou funkcí takový, že $L(A) = L(A')$.

Důkaz

Zkonstruujeme A' dodefinováním chybějících přechodů v původním $A = (Q, \Sigma, \delta, q_0, F)$.

1. Do A přidáme nový nekoncový stav $q^\star$, tzv. *sink state*.
2. Pro každý nedefinovaný přechod $\delta(q, a)$ definujeme $\delta'(q, a) = q^\star$.
3. Definujeme přechod $\delta'(q^\star, a) = q^\star$ pro každý znak $a \in \Sigma$.
4. Získali jsme požadovaný automat $A' = (Q \cup \{q^\star\}, \Sigma, \delta \cup \delta', q_0, F)$.

Očividně platí $L(A) = L(A')$, protože upravujeme jen výpočet pro zamítnuté řetězce. □