Automaty a formální jazyky

Přednáška II.

Regulární výrazy

Ekvivalence a minimalizace automatů

Regulární výrazy - motivace

- Jiný způsob pro definici jazyka.
- Na rozdíl od automatů jsou popisné – definují jedním výrazem celý jazyk.
- V praxi: všude, kde potřebujeme vymezit, jak má vypadat nějaký textový řetězec
  - vstup programu,
  - heslo,
  - jméno,
  - pravidla pro vyhledávání…
- Jazyk definovaný regulárním výrazem se nazývá **regulární jazyk**.

Regulární výraz teoreticky

Množina regulárních výrazů *RV(∑)* nad abecedou *∑= {a_1,…,a_n}* je nejmenší množina slov v abecedě *Ω = {a_1,…,a_n, ø, ε, +, ., *,(,)}*, pro kterou platí:

1. Obsahuje výraz *ø* a výraz *ε*
2. Pro každé a *∈ ∑* obsahuje výraz a
3. Pokud α,β ∈ RV(∑) potom:
   - α + β ∈ RV(∑)
   - α .β ∈ RV(∑)
   - α* ∈ RV(∑)

Hodnota regulárního výrazu teoreticky

Hodnotou regulárního výrazu je množina slov [α] definovaná následovně:

1. $[\emptyset] = \emptyset$, $[\varepsilon] = \varepsilon$, $[a] = a$
2. $[(\alpha + \beta)] = [\alpha] \cup [\beta]$ (nebo)
3. $[(\alpha \, . \, \beta)] = [\alpha] \, . \, [\beta]$ (zřetězení)
4. $[(\alpha \, *)] = [\alpha]^*$ (opakování)

Co to znamená?

- **Regulární výraz** je maska, která nám pomáhá definovat podobu slov:
- Příklad: *∑= {a, b, c}:*
  - *a*+b**: Vyhovují slova, která obsahují jen a nebo jen b
  - *abc**: Vyhovují slova, která začínají na řetězec abc a pokračují jen písmeny c
- Příklad: *∑= {a-z, A-Z}*
  - *Lenka:* Definuje jazyk, který obsahuje pouze slovo Lenka
  - (l+L)(e+E)(n+N)(k+K)(a+A): Definuje jazyk, který obsahuje slovo Lenka psané libovolnou kombinací velkých a malých písmen

Jak vyhodnocovat regulární výraz

Existuje tato dohoda - pravidla pro operátory „podobná“ algebře:

- Vnější závorky se vynechávají
- Závorky u operátorů +, * a . se také mohou vynechat
- Tečka se obvykle nepíše
- Priorita vyhodnocení: *, ., +

Doplnění pro „reálné“ jazyky nad latinkou

- Začátek řetězce: ^
- Konec řetězce: $
- Možná: ?
- Skupina: []
- „Jen písmena“: [a-z], [A-Z], [a-z,A-Z],[a-b]
- „Jen čísla“: [0-9]
- Jak zapsat znak, který má syntaktický význam? S lomítkem: * není opakování, ale hvězdička
- Definice počtu opakování: v „jiných“ (nejčastěji složených) závorkách číslo
- Mnoho programovacích jazyků navíc definuje „zkratky“:
  - **\w nebo \word** je slovo (jen písmena v libovolném opakování, bez mezery)
  - **\n** je znak konce řádky
  - **\t** jsou tabulátory nebo obecně mezery

Praktické příklady:

- Vzorec pro datum, první je den, druhý měsíc:
  [1-3]?[0-9]\.[1-2]?[0-9].
  Nevýhoda: vyhoví i výraz 39.25.
- Lepší vzorec pro datum:
  (([1-2]?[0-9])+(3[0-1]))\.([1-9]+(1[0-2]))
  Nevýhoda: povolí i výraz 31.2.
- Vymazlený vzorec pro datum:
  (([1-2]?[0-9])+(3[0-1])\.([1,3,5,7,8,10,12])) +
  (([1-2]?[0-9])+(30)\.([4,6,9,11])) +
  (([1-2]?[0-9])\.2)

Vztah mezi RV a konečnými automaty

Kde je konečný automat, tam lze nalézt regulární výraz.

Věta: Je-li jazyk L přijímán nějakým konečným automatem A, potom také existuje nějaký regulární výraz, který definuje jazyk L.

Důkaz – princip:

- Předpokládáme, že KA má *n* stavů, seřadíme je a očíslujeme
- Definujeme si dílčí regulární výrazy, které popisují „cestu“ automatem ze stavu *i* do stavu *j*, přičemž cesta nevede skrz stav s vyšším indexem než *k*
- Postupně hledáme jednotlivé dílčí výrazy od k=0 až ke k=n
- Hledaný regulární výraz je „součet“ všech takto konstruovaných cest.

Vlastnosti regulárních jazyků

- Sjednocení dvou RJ je RJ.
- Průnik dvou RJ je RJ.
- Zřetězení dvou RJ je RJ.
- Rozdíl dvou RJ je RJ.
- Doplněk k RJ je také RJ.

Optimalizace KA I.

- **Definice:** Automaty A a B jsou si **ekvivalentní**, pokud definují stejný jazyk, tedy L(A) = L(B).
- **Definice:** Stav q je **dosažitelný,** jestliže existuje slovo $w \in \Sigma^*$ takové, že $\delta^*(q_0,w)=q$
- **Definice:** Stav q je **ekvivalentní** se stavem r, jestliže

  $\delta^*(q,w) \in F$ právě tehdy, když $\delta^*(r,w) \in F$ pro všechna $w \in \Sigma^*$.

Optimalizace KA II.

- Jeden jazyk mohou definovat různé automaty.
- Intuitivně tíhneme k automatům s minimálním počtem stavů:
  - Odstraňujeme nedosažitelné stavy (automat se do nich nedostane).
  - Odstraňujeme ekvivalentní stavy (výpočet z nich jde stejnou cestou, stačí jen jeden).

Hledání dosažitelných stavů - algoritmus

- Princip: K již známým dosažitelným stavům přidáváme postupně v jednotlivých krocích ty stavy, do nichž se lze dostat jedním symbolem z již známých dosažitelných stavů.
- Počátek: počáteční stav je vždy dosažitelný.
- Konec: v daném kroku už nepřibude žádný další stav.

Hledání dosažitelných stavů - algoritmus

- Princip: K již známým dosažitelným stavům přidáváme postupně v jednotlivých krocích ty stavy, do nichž se lze dostat jedním symbolem z již známých dosažitelných stavů.
- Počátek: počáteční stav je vždy dosažitelný.
- Konec: v daném kroku už nepřibude žádný další stav.

Redukovaný automat

- **Definice:** Konečný automat je **redukovaný**, když jsou všechny jeho stavy dosažitelné a nejsou si ekvivalentní.
- **Definice:** Automat A je **reduktem** automatu B, jestliže jsou si automaty ekvivalentní a A je redukovaný automat.
- **Věta**: Ke každému konečnému automatu existuje jeho redukt.

S pomocí redukce rozhodujeme

- **Definuje automat vůbec nějaký jazyk?** Jen tehdy, má-li dosažitelný alespoň jeden koncový stav.
- **Definuje automat celý uzávěr nad abecedou?** Pokud ano, potom jej lze redukovat na jeden jediný stav.
- **Jsou dva automaty ekvivalentní?** Pokud ano, jdou oba redukovat na stejný automat.