UNIVERZITA KOMENSKÉHO, BRATISLAVA
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

MODERNÉ REGULÁRNE VÝRAZY

Bakalárska práca

2013
Tatiana Tóthová

Univerzita Komenského, Bratislava
Fakulta Matematiky, Fyziky a Informatiky

Moderné regulárne výrazy

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra Informatiky
Školiteľ:	RNDr. Michal Forišek, PhD.

Bratislava, 2013
Tatiana Tóthová

83491340

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta:	Tatiana Tóthová
Študijný program:	informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor:	9.2.1. informatika
Typ záverečnej práce:	bakalárska
Jazyk záverečnej práce:	slovenský

Názov: Moderné regulárne výrazy

Cieľ: Spraviť prehľad nových konštrukcií používaných v moderných knižniciach s regulárnymi výrazmi (ako napr. look-ahead a look-behind assertions). Analyzovať tieto rozšírenia z hľadiska formálnych jazykov a prípadne tiež z hľadiska algoritmickej výpočtovej zložitosti.

Vedúci:	RNDr. Michal Forišek, PhD.
Katedra:	FMFI.KI - Katedra informatiky
Vedúci katedry:	doc. RNDr. Daniel Olejár, PhD.
Dátum zadania:	23.10.2012

Dátum schválenia: 24.10.2012

doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

Poďakovanie

Tatiana Tóthová

Abstrakt

Abstrakt po slovensky

Kľúčové slová: napíšme, nejaké, kľúčové, slová

Abstract

Abstract in english

Key words: some, key, words

Obsah

Úvod

Kapitola 1

Názov kapitoly 1

V tejto kapitole formálne definujem operácie z uvedenej dokumentácie jazyka Python [doc12] a ukážem ich silu. Budem používať nasledovné zápisy:

L_1L_2 – zreťazenie jazykov L_1 a L_2

L^* – iterácia ($L^* = \cup_{i=0}^{\infty} L^i$, kde $L^0 = \{\varepsilon\}$, $L^1 = L$ a $L^{i+1} = L^iL$)

$\mathcal{R}$ – tradičné označenie triedy regulárnych jazykov

DKA/NKA – deterministický/nedeterministický konečný automat

1.1 Podnadpis 1

Definícia 1.1.1 (Greedy iterácia)**.**

$$L_1 \circledast L_2 = \{uv \mid u \in L_1^* \wedge v \in L_2 \wedge u \textit{ je najdlhšie také}\}$$

Definícia 1.1.2 (Minimalistická iterácia)**.**

$$L_1 {*?} L_2 = \{uv \mid u \in L_1^* \wedge v \in L_2 \wedge u \textit{ je najkratšie také}\}$$

Veta 1.1.3. $L_1 \circledast L_2 = L_1{*?}L_2 = L_1^*L_2$

Dôkaz. $\subseteq$: Nech $w \in L_1 \circledast L_2$. Potom z definície $w = uv$ vieme, že $u \in L_1^*$ a $v \in L_2$, teda $uv \in L_1^*L_2$. Analogicky ak $x = yz \in L_1{*?}L_2$, potom $yz \in L_1^*L_2$.

$\supseteq$: Majme $w \in L_1^*L_2$ a rozdeľme na podslová u, v tak, že $u \in L_1^*, v \in L_2$ a $w = uv$. Takéto rozdelenie musí byť aspoň jedno. Ak je ich viac, vezmime to, kde je u najdlhšie. Potom $uv \in L_1 \circledast L_2$. Ak zvolíme u najkratšie, tak zasa $uv \in L_1{*?}L_2$. □

Dôsledok 1.1.4. *Trieda $\mathcal{R}$ je uzavretá na operácie $\circledast$ a $*?$.*

Definícia 1.1.5 (Lookahead)**.**

$$L_1(? = L_2)L_3 = \{uvw \mid u \in L_1 \wedge v \in L_2 \wedge vw \in L_3\}$$

Operáciu $(? = \dots)$ *nazývame lookahead.*

Veta 1.1.6. *Nech $L_1, L_2, L_3 \in \mathcal{R}$. Potom $L = L_1(? = L_2)L_3 \in \mathcal{R}$.*

Dôkaz. Nech L_1, L_2, L_3 sú regulárne, nech $A_i = (K_i, \Sigma_i, \delta_i, q_{0i}, F_i)$ sú DKA také, že $L(A_i) = L_i, i \in \{1, 2, 3\}$. Zostrojím NKA $A = (K, \Sigma, \delta, q_0, F)$ pre L, kde $K = K_1 \cup K_2 \times$

$K_3 \cup K_3$ (predp. $K_1 \cap K_3 = \emptyset$), $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \Sigma_3$, $q_0 = q_{01}$, $F = F_3 \cup F_2 \times F_3$, δ funkciu definujeme nasledovne:

$$
\begin{aligned}
\forall q \in K_1, \forall a \in \Sigma &: \quad \delta(q, a) \ni \delta_1(q, a) \\
\forall q \in F_1 &: \quad \delta(q, \varepsilon) \ni [q_{02}, q_{03}] \\
\forall p \in K_2, \forall q \in K_3, \forall a \in \Sigma_2 \cap \Sigma_3 &: \quad \delta([p, q], a) \ni [\delta(p, a), \delta(q, a)] \\
\forall p \in F_2, \forall q \in K_3 &: \quad \delta([p, q], a) \ni \delta(q, a)
\end{aligned}
$$

$L(A) = L$.

$\supseteq$: Máme $w \in L$ a chceme preň nájsť výpočet na A. Z definície L vyplýva $w = xyz$, kde $x \in L_1, y \in L_2$ a $yz \in L_3$, teda existujú akceptačné výpočty pre x, y, yz na DKA A_1, A_2, A_3. Z toho vyskladáme výpočet pre w na A nasledovne. Výpočet pre x bude rovnaký ako na A_1. Z akceptačné stavu A_1 vieme na ε prejsť do stavu $[q_{02}, q_{03}]$, kde začne výpočet pre y. Ten vyskladáme z A_2 a A_3 tak, že si ich výpočty napíšeme pod seba a stavy nad sebou budú tvoriť karteziánsky súčin stavov v A (keďže A_2 aj A_3 sú deterministické, tieto výpočty na y budú rovnako dlhé). $y \in L_2$, teda A_2 skončí v akceptačnom stave. Podľa δ funkcie v A vieme pokračovať len vo výpočte na A_3, teda doplníme zvyšnú postupnosť stavov pre výpočet z. Keďže $yz \in L_3$ a $F_3 \subseteq F$ (resp. $F_2 \times F_3 \subseteq F$ pre $z = \varepsilon$), automat A akceptuje.

$\subseteq$: Nech $w \in L(A)$, potom existuje akceptačný výpočet na A. Z toho vieme w rozdeliť na x, y a z tak, že x je slovo spracovávené od začiatku po prvý príchod do stavu $[q_{02}, q_{03}]$, y odtiaľto po posledný stav reprezentovaný karteziánskym súčinom stavov a zvyšok bude z. Nevynechali sme žiadne znaky a nezmenili poradie, teda $w = xyz$. Do $[q_{02}, q_{03}]$ sa A môže prvýkrát dostať len vtedy, ak bol v akceptačnom stave A_1. Prechod do $[q_{02}, q_{03}]$ je na ε, takže $x \in L_1$. Práve tento stav je počiatočný pre A_2 aj A_3. Ak $z = \varepsilon$, tak akceptačný stav A je z $F_2 \times F_3$ a $y \in L_2, y \in L_3$ a aj $yz \in L_3$. Z toho podľa definície vyplýva, že $xyz = w \in L$. Ak $z \neq \varepsilon$, potom je akceptačný stav A z F_3. Podľa δ funkcie sa z karteziánskeho súčinu stavov do normálneho stavu dá prejsť len tak, že A_2 akceptuje, teda $y \in L_2$. A_3 akceptuje na konci, čo znamená $yz \in L_3$. Znova podľa definície operácie lookahead $xyz = w \in L$. □

Definícia 1.1.7 (Lookbehind)**.**

$$L_1(? <= L_2)L_3 = \{uvw \mid uv \in L_1 \wedge v \in L_2 \wedge w \in L_3\}$$

Operáciu $(? <= \dots)$ *nazývame lookbehind.*

Veta 1.1.8. *Nech* $L_1, L_2, L_3 \in \mathcal{R}$. *Potom* $L = L_1(? <= L_2)L_3 \in \mathcal{R}$.

Dôkaz. Podobne ako pri lookahead. (Karteziánsky súčin stavov L_1 a L_2, ale A_2 sa pripája v každom stave A_1 - celkový NKA si potom nedeterministicky zvolí jeden moment tohto napojenia.) □

TODO!!!lookbehind - podobne ako lookahead

Veta 1.1.9. $\mathcal{L}_{CF}$ *nie je uzavretá na operácie lookahead a lookbehind.*

Dôkaz. Nech $L_1, L_2, L_3, L_4 \in \mathcal{L}_{CF}$. $L_1 = \{a^n b^n \mid n \geq 1\}, L_2 = \{a * b^n c^n \mid n \geq 1\}, L_3 = \{a^n b^n c * \mid n \geq 1\}, L_4 = \{ab^n c^n \mid n \geq 1\}$. Potom $d(? = L_1)L_2 = \{da^n b^n c^n \mid n \geq 1\}$ a $L_3(? <= L_4)d = \{a^n b^n c^n d \mid n \geq 1\}$, čo nie sú bezkontextové jazyky. □

Veta 1.1.10. *$\mathcal{L}_{CS}$ je uzavretá na operáciu lookahead.*

Dôkaz. Pre $L_1, L_2, L_3 \in \mathcal{L}_{CS}$ a slovo z $L = L_1(? = L_2)L_3$ zostrojíme LBA A z LBA A_1, A_2, A_3 pre dané kontextové jazyky. Najprv sa pozrime na štruktúru vstupu – prvé je slovo z L_1 a za ním nasleduje slovo z L_3, pričom jeho prefix patrí do L_2. Preto, aby A mohol simulovať dané lineárne ohraničené automaty, je potrebné označiť hranice jednotlivých slov.

Na začiatku výpočtu A prejde pásku a nedeterministicky označí 2 miesta – koniec slov pre A_1 a A_2. Následne sa vráti na začiatok a simuluje A_1. Ak akceptuje, A pokračuje a presunie sa za označený koniec vstupu pre A_1. Inak sa zasekne. V tomto bode sa začína vstup pre A_2 aj A_3, teda slovo až do konca prepíše na 2 stopy. Najprv na hornej simuluje A_2. Pokiaľ A_2 neskončí v akceptačnom stave, A sa zasekne. Inak sa vráti na označené miesto a simuluje A_3 na spodnej stope až do konca vstupu. Akceptačný stav A_3 znamená akceptáciu celého vstupného slova. $\square$

1.2 Podnadpis 2

Záver

Literatúra

[Cox07] Russ Cox. *Regular Expression Matching Can Be Simple And Fast (but is slow in Java, Perl, PHP, Python, Ruby, ...)*, 2007. `http://swtch.com/~rsc/regexp/regexp1.html` [Online; accessed 30-December-2012].

[CSY03] CEZAR CÂMPEANU, KAI SALOMAA, and SHENG YU. A formal study of practical regular expressions. *International Journal of Foundations of Computer Science*, 14(06):1007–1018, 2003. `http://www.worldscientific.com/doi/abs/10.1142/S012905410300214X` [Online; accessed 19-March-2013].

[doc12] Python documentation. *Regular expressions operations*, 2012. `http://docs.python.org/3.1/library/re.html` [Online; accessed 30-December-2012].