Referát z předmětu Teoretická informatika

Téma: Algoritmus Coke-Younger-Kasami pro rozpoznávání bezkontextových jazyků

VŠB-TU Ostrava:Fakulta Elektrotechniky a informatiky

jaro 2011

Martin Dočkal
doc068
dockal.martin@gmail.com

1 Obsah

2 Zadání

Algoritmus (Cocke-Younger-Kasami) pro rozpoznávaní bezkontextových jazyků (aplikace metody dynamického programováni):
Mějme dánu bezkontextovou gramatiku *G* v tzv. Chomskeho normální formě, tedy s pravidly pouze typu $X \to YZ$ a $X \to a$. Algoritmus pro zadané (terminální) slovo *w* zjistí, zda $w \in L(G)$.
Nástin: Označme $w = a_1 a_2 \dots a_n$. Systematicky vyplňujeme (dvourozměrné) pole *D* tak, že na závěr bude $D[i,j] = (1 \leq i \leq n, 0 \leq j \leq n - i)$ obsahovat množinu pravě těch neterminálů *X*, z nichž lze odvodit $a_1 a_{i+1} \dots a_{i+j}$.
Vysvětlete tento algoritmus ilustraci na vhodném přikladu a objasněte jeho časovou složitost.

3 Základní pojmy

3.1 Jazyk

Obecně máme v této práci na mysli tzv. formální jazyk, naproti tomu speciálními jazyky jsou přirozené (mluvené) či jazyky programovací.
Na naše modely se v prvé řadě budeme dívat jako na rozpoznávače jazyků, tj. zařízení, která zpracují vstupní posloupnost písmen (symbolů) a rozhodnou, zda tato posloupnost je (správně utvořenou) větou příslušného jazyka [1].
Jazyk *L* můžeme formálně zapsat v oboru teoretické informatiky ve tvaru

$$L = \{w \in \sum^* | K\}$$

kde

- $w \in \sum^*$ je nějaké slovo w složené z abecední množiny $\sum$
- *K* je množina integritních omezeních platných nad každým slovem *w* patřící do jazyka *L*

Příklad zápisu jazyka generující jazyk palindromů:

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

3.2 Gramatika

S výše zmíněným pojmem jazyk bývá v asociaci další, kterým je gramatika *G*, jež obecně generuje právě jazyk *L*, který pak označujeme $L(G)$, do nějž pak patří pouze terminální slova odvozená z *S*, tedy z počátečního (inicializačního) neterminálu, který také bývá nazýván počátečním. Gramatiku G opět můžeme obecně a formálně zapisovat ve tvaru odvozeném z definice bezkontextové gramatiky [1 str. 127] :

$$G = (\Pi, \sum, S, P)$$

kde

- Π je konečná množina neterminálních symbolů (neterminálů)
- $\sum$ je konečná množina terminálních symbolů (terminálů), přičemž platí $\Pi \cap \sum = \{\}$
- $S \in \Pi$ je počáteční (startovací) neterminál
- *P* je konečná množina pravidel typu $\beta \rightarrow \beta$, kde
  - β je řetězec složený z terminálů a neterminálů, tedy $\beta \in (\Pi \cup \Sigma)*$.

Příklad zápisu gramatiky:

$$S \rightarrow \varepsilon \mid a \mid b \mid aSa \mid bSb$$

V tomto případě by podle námi zvoleného zápisu mohla taková gramatika vypadat ve formálním tvaru takto:

$$G = (\{S\}, \{a, b\}, \{S\}, \{S \rightarrow \varepsilon, S \rightarrow a, S \rightarrow b, S \rightarrow aSa, S \rightarrow bSb\})$$

3.3 Jazyk generovaný gramatikou

V předchozí definici u pojmu gramatiky jsme se setkali s výrazem "jazyk generovaný gramatikou G", pro který jsme si zavedlo formální značení $L(G)$ a platí [1 str. 128]:

$$L(G) = \{w \in \sum^* | S \Longrightarrow {}^* w\}$$

kde

- *w* je slovo vygenerované nad abecedou terminálů z množiny $\sum$
- *w* je slovo odvozené z počátečního neterminálu *S* v konečně mnoha krocích ($S \Longrightarrow {}^* w$)

3.4 Bezkontextová gramatika

V tomto referátu se věnujeme jejich podmnožině zvané bezkontextová gramatika, která generuje jazyky, jež jsou rozpoznávány zásobníkovými automaty, případně jsme se s nimi mohli setkat u definic syntaxí programovacích jazyků [1]. Podle definice [1 str. 127] je bezkontextová gramatika definována jako uspořádaná čtveřice

$$G = (\Pi, \sum, S, P)$$

kde

- Π je konečná množina neterminálních symbolů (neterminálů)
- ∑ je konečná množina terminálních symbolů (terminálů), přičemž platí Π ∩ ∑ = {}
- $S \in \Pi$ je počáteční (startovací) neterminál
- *P* je konečná množina pravidel typu A → β, kde
  - A je neterminál, tedy A ∈ Π
  - β je řetězec složený z terminálů a neterminálů, tedy β ∈ (Π∪Σ)∗.

Neformálně řečeno slovo "bezkontextová" v názvu gramatiky znamená, že na levé straně každého pravidla stojí jeden neterminál bez sousedních symbolů, který můžeme při jakémkoliv odvození přepsat podle uvedených pravidel nezávisle na jeho okolí (kontext).

Podle zvyklostí a konvencí používáme pro neterminály velká písmena (A, B, C, ...), naopak pro neterminály malá písmena (a, b, c, ...) či další symboly (+,(, apod.).

3.5 Bezkontextový jazyk

Podle definice [1 str. 128] je jazyk *L* generovaný gramatikou *G*, pak označovaný jako $L(G)$, bezkontexotvý, jestliže existuje bezkontextová gramatika *G* taková, že $L(G) = L$.

3.6 Zásobníkový automat

Zásobníkový automat představuje rozšíření konečného automatu obohaceného o datovou strukturu zásobník. Tyto zásobníkové automaty rozpoznávají třídu tzv. bezkontextových jazyků, což je nadtřída třídy regulárních jazyků [1].

3.7 Algoritmus

Matematickým pojmem algoritmus nazýváme posloupnost konečného počtu elementárních kroků vedoucí k vyřešení úlohy.

3.8 Dynamické programování

Odvětvím optimalizace je právě dynamické programování, jehož stěžejní myšlenkou je rozklad problému na podproblémy, které jsou řešeny a jejich řešení je ukládáno pro další potenciálně možné použití. Zejména se užívá u takových úloh, jenž se dají dělit na podúlohy, které jsou si vzájemně podobné a mohou se opakovat, přičemž obvykle existuje více způsobů takového rozkladu na podproblémy, což může mít vliv na efektivitu celého výpočtu. [2]

3.9 Chomského normální forma

Z důvodů další analýzy gramatik bývá v některých případech vhodné pokusit se ji transformovat do některé z definovaných norem. Podle definice platí [1 str. 160]:
Bezkontextová gramatika je v Chomského normální formě, zkráceně v CNF, jestliže každé její pravidlo je tvaru $X \to YZ$ nebo $X \to a$ a kde *X,Y,Z* jsou neterminální symboly a *a* terminální symbol.
Podle věty uvedené v [1 str. 161] pak platí následující tvrzení: Ke každé bezkontextové gramatice *G* lze sestrojit gramatiku G' v CNF tak, že platí:

$$L(G') = L(G) - \{\varepsilon\}$$

3.10 Časová složitost

Časovou složitostí algoritmu $O(n)$ v závislosti na velikosti testovaného vstupu o délce n vyjadřujeme funkcí s nekonečným definičním oborem, tzn. že tvar této funkce nelze zapsat zápisem jako výčtem hodnot, ale hledáme pro ni konečný popis, resp. odhad. Příklad zápisu časové složitosti pro (zde mj. pro algoritmus třídění buble sort) :

$$O(n) = n^2$$

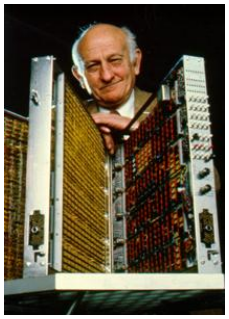
Protože vstupů se stejnou velikostí n ovšem může být různé množství s různou časovou složitostí pro jednotlivé vstupy. Výslednou funkcí pro časovou složitost algoritmu je přiřazení velikosti n funkci s nejvyšší časovou složitostí (tzv. worst-case případ [1 str. 232]), přičemž zanedbáváme konstanty. Tedy např. časové složitosti ve tvaru $O(n) = 100n + 1000$ a $O(n) = n$ pak považujeme za shodné. Naopak v případě komparace s výše zmíněnou časovou složitostí pro buble sort bychom upřednostňovali právě buble sort, neboť má vyšší časovou složitost a považovali bychom jej za výslednou časovou složitost celého algoritmu.

4 Autoři algoritmu CYK

Autory algoritmu, jenž je hlavním předmětem tohoto referátu, jsou John Coke (30.5. 1925 - 16.7.2002) a Tadao Kasami (12.4.1930 - 18.3. 2007). [3]

4.1 John Coke

Americký vědec v oboru informačních technologiích. Zaměřoval se na architekturu počítačů a návrhy optimalizačních kompilátorů. Mnohými je označován jako otec architektury RISC. Promoval ve svých 21 letech na univerzitě Duke University roku 1946 v oboru strojního inženýrství. Na téže univerzitě o 7 let později získal také doktorát z matematiky.



Obrázek 1: John Coke

Po studiích ztrávil celou svou pracovní kariéru ve výzkumném středisku IBM, kde působil nepřetržitě 36 let až do roku 1992, kdy odešel do penze.
Za svůj život získal řadu ocenění (Eckert-Mauchly Award 1985, ACM Turing Award 1987, National Medal of Technology 1991, National Medal of Science 1994, a mnoho dalších, poslední pak The Benjamin Franklin Medal v roce 2000). [4]

4.2 Tadao Kasami

Japonský vědec teoretické informatiky, převážně se zabýval korekci chybových kódů (Kasami code).

Obrázek 2: Tadao Kasami

Studoval elektrotechniku na univerzitě Osaka University, kde promoval ve 30 letech a o tři roky později, v roce 1963, zde také získal doktorát. Svou profesionální kariéru pak zasvětil předáváním svých poznatků fakultním studentům jako profesor až do roku 2004.
Oceněn řadou vyznamenání z oboru informatiky, mezi jeho nejznámější patří Claude E. Shannon Award (1999). [5]

5 Algoritmus CYK

5.1 Princip

Algoritmus CYK určuje, zda slovo náleží do bezkontextového jazyka, který musí být zapsán gramatikou v Chomského normální formě. Algoritmus končí a ověří, jestliže ve výsledné množině neterminálů, kterou během svého chodu postupně vygeneruje, se nachází vstupní (startovní, inicializační) neterminál.

5.2 Postup

Na vstupu máme terminální slovo w a gramatiku v CNF (3.9 Chomského normální forma).

5.2.1 Formální (pseudokód)

Vytvoříme pole $T[i, j, x]$, pro $1 \leq i \leq j \leq n$, kde x jsou postupně všechny neterminály (nebo jejich odkazy v podobě např. celých čísel), a hodnoty všech jeho prvků nastavíme na 0.
Pro každý znak a na pozici i, a pro každé X takové, že v gramatice existuje pravidlo $X \rightarrow a$, nastavíme v poli $T[i, 1, X] = 1$
Pro každou délku podslova i od 2 do n:
    Pro každý začátek podslova j od 1 do $n - i + 1$:
        Pro každou délku první poloviny podslova k od 1 do $i - 1$:
            Jestliže v poli mají jedničkovou hodnotu $T[i, j, X]$ i $T[j + k, i - k, Y]$ a v gramatice existuje pravidlo $Z \rightarrow XY$, nastavíme v poli $T[j, i, z]$ jedničku.

Slovo náleží do jazyka, jestliže $T[1, n, S] = 1$, kde S je vstupní neterminál gramatiky. [6]

5.2.2 Neformální

Inicializujeme dvourozměrné pole B_{ij}, kde počet sloupců i řádků odpovídá velikosti délky vstupního slova w.

Buňky prvního řádku B_{1j} odpovídají vždy množině takových neterminálů z gramatiky CNF, ze kterých lze odvodit příslušný neterminál na *j*-té pozici ve slově *w*.
Pak pro každý následující řádek provádíme následující postup:
Množina neterminálů v buňce B_{ij} je rovna takové množině neterminálů z CNF, ze kterých lze odvodit vždy dvojice neterminálů vzniklých z dvojic buněk "nejspodnější" aktuální buňky a na "úhlopříčce" vpravo "nejvýše" od aktuální buňky B_{ij}.
Ve výsledku nám na "vrcholu schodišťové tabulky" vznikne jediná množina neterminálů *T*. Pokud se v ní nachází startovní neterminál, je slovo *w* přijato.
Postup pro vyplňování všech políček v tabulce množinou neterminálů nejlépe ilustruje následující obrázek (Obrázek 3), kde demonstrujeme vyplnění políčka na souřadnici 4. sloupce a 4. řádku, zvýrazněného žlutým kolečkem. Sestrojíme dvojice neterminálu vždy z takových buněk (množin) neterminálů, které mají stejnou barvu (na obrázku modrá, fialová a zelená). Z těchto tří dvojic nám vznikají variace množit napsaných v elipsách příslušné barvy. Takto vzniklé dvojice otestujeme, zda jsou generovány nějakým neterminálem (v zadání) a pokud ano, zapíšeme jej jako výsledek do množiny neterminálů v tomto případě do buňky označené žlutým kroužkem. V tomto případě se jedná o neterminály B a A, neboť dvojice CA (z modré elipsy) je generována právě neterminálem B, dvojice BA (zelená elipsa) je pak generována neterminálem A. Zbývající dvojice neterminálů nejsou generovány žádným neterminálem a výpočet pro tuto aktuální buňku končí. Dvojice druhé (fialové) elipsy nemusíme ani testovat, zda jsou generovány nějakým neterminálem, neboť žádnou dvojici neobsahuje, pouze jeden neterminál, což odporuje CNF.
Takto pokračujeme pro všechny buňky pod "hlavní diagonálou" (v ilustraci buňky s bílým pozadím) tabulky dokud nenarazíme na řádek, který už nelze dále takto redukovat (v našem případě buňka sedmého řádku a prvního sloupce.

5.3 Příklad

Mějme následující gramatiku v CNF

$S \rightarrow AA \mid BC \mid a \mid b$

$A \rightarrow BA \mid BB \mid a \mid c$

$B \rightarrow AB \mid CA \mid b$

$C \rightarrow BC \mid CC \mid SC \mid d$

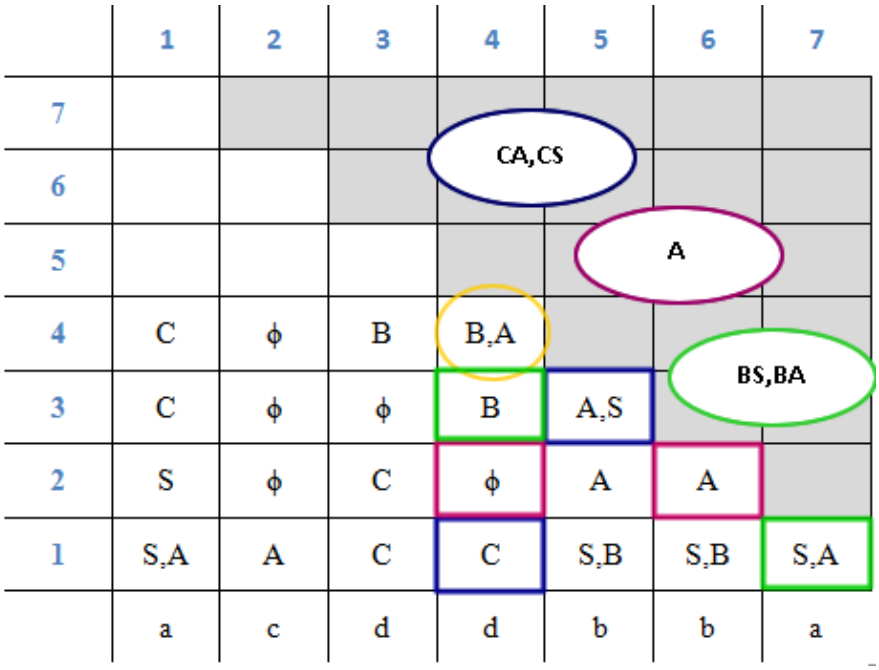
a na vstupu slovo

```
acddbba
```

Otestujme pomocí algoritmu CYK, zda je slovo přijato výše zmíněnou gramatikou.

	1	2	3	4	5	6	7
7							
6							
5							
4							
3							
2							
1							
	a	c	d	d	b	b	a

Tabulka 1: Tabulka před chodem algoritmu



Obrázek 3 : Ilustrace konkrétního odvození zvýrazněné buňky

	1	2	3	4	5	6	7
7	B,S,A						
6	B	B,S,A					
5	ϕ	B	B,A				
4	C	ϕ	B	B,A			
3	C	ϕ	ϕ	B	A,S		
2	S	ϕ	C	ϕ	A	A	
1	S,A	A	C	C	S,B	S,B	S,A
	a	c	d	d	b	b	a

Tabulka 2 : Vyplněná tabulka po skončení algoritmu

Z konečné výsledné tabulky (Tabulka 2) vidíme, že poslední buňka, tedy v tomto případě na pozici prvního sloupce a sedmého řádku, obsahuje množinu neterminálů B,S,A. Protože obsahuje počáteční neterminál S, tedy podle této syntaktické analýzy lze vygenerovat předloženým jazykem slovo w. Formálně zapsáno:
$S \in T_{1,7} = S, B, A$, tedy $acddbba \in L(G)$

5.4 Složitost

Zejména z formálního zápisu je zřejmé, že algoritmus obsahuje, vzhledem k délce slova na vstupu, celkem třikrát cyklus procházení každého prvku, celý proces má tedy kubickou časovou složitost, vyjádřeno zápisem:

$$O(n) = n^3$$

6 Literatura

[1]**Prof. RNDr. Jančar, Jan CSc.** *Teoretická informatika.* Ostrava : VŠB – Technická univerzita Ostrava, 2007. ISBN 978-80-248-1487-2.
[2] Dynamické programování. *Wikipedia, the free encyklopedia.* [Online] [Citace: 15. Duben 2011.] http://cs.wikipedia.org/wiki/Dynamick%C3%A9_programov%C3%A1n%C3%AD.
[3]CYK algorithm. *Wikipedia, the free encyklopedia.* [Online] [Citace: 15. Duben 2011.] http://en.wikipedia.org/wiki/CYK_algorithm.
[4]John Cocke - Wikipedia, the free encyclopedia. *Wikipedia.* [Online] [Citace: 15. Duben 2011.] http://en.wikipedia.org/wiki/John_Cocke.
[5]Tadao Kasami - Wikipedia, the free encyclopedia. *Wikipedia, the free encyclopedia.* [Online] [Citace: 15. Duben 2011.] http://en.wikipedia.org/wiki/Tadao_Kasami.
[6]Algoritmus Cocke-Younger-Kasami. *Wikipedia, free encyklopedia.* [Online] [Citace: 15. Duben 2011.] http://cs.wikipedia.org/wiki/Algoritmus_Cocke-Younger-Kasami.
[7]**Barták, Roman.** Předmět Automaty a gramatiky. [Online] MFF Praha, 2011. [Citace: 15. Duben 2011.] http://ktiml.mff.cuni.cz/~bartak.
[8] SMR - Syntaktická analýza Cocke-Younger-Kasami. *Katedra kybernetiky ZČU - Umělá inteligence.* [Online] ZČU. [Citace: 15. Duben 2011.] http://www.kky.zcu.cz/cs/courses/smr/sem-cyk.