Model syntaktické analýzy formálních jazyků

Martin Hrubý *

Petr Vurm*

Alexander Meduna*
{hrubym,vurm,meduna}@fit.vutbr.cz

Abstract: Abstract Tento článek pojednává o principech řízených gramatik a o možnostech jejich zpracování. Zabývá se především jedním konkrétním typem řízené gramatiky obecně známé pod názvem Cooperating Distributed (CD) Grammar (CDG). Byla navržena jednoduchá metoda syntaktické analýzy CDG jazyků, článek popisuje teoretické předpoklady pro analyzovatelnost CDG a prototyp syntaktického analyzátoru CDG.

Keywords: Cooperating Distributed Grammar, syntaktická analýza, levá derivace

1 Úvod

Bezkontextové gramatiky tvoří významnou skupinou gramatik podle Chomského. Jsou to nejčastěji prakticky používané gramatiky, protože syntaktická analýza bezkontextových jazyků je jednoduchá a průhledná. Stačí na ni stroj s jedním zásobníkem. Tyto analyzátory se velmi snadno implementují v programovacích jazycích a v podstatě bezkontextové gramatiky postačí v drtivé většině prakticky řešených aplikací gramatik a překladačů. Myslí se tím, že při návrhu gramatiky pro určitý programovací jazyk, se obvykle vystačí s rozsahem bezkontextových gramatik a pravidel. Občas se ale (a to snad výjimečně) objeví situace, kdy určitou jazykovou konstrukci opravdu nelze zapsat v bezkontextových přepisovacích pravidlech.
V posledních letech probíhá výzkum gramatických systémů a teorií gramatik s cílem přinést gramatiku s vyšší generovací silou, která zároveň udržuje jednoduchost a lidskou přístupnost bezkontextových pravidel. Proto vznikly gramatické systémy a řízené gramatiky. Je jich celkem rozsáhlá skupina a všechny v podstatě řeší stejný problém: jak ve přepisovacích pravidlech a v relaci derivace skrýt a zamaskovat *kontext*, který fakticky řídí průběh generování věty a omezuje definiční obor relace derivace.
V tomto článku definujeme gramatiky nazývané Cooperating Distributed Grammars (CDG), jejich modifikovanou podobu (mCDG) vhodnou pro deterministickou syntaktickou analýzu a model syntaktického analyzátoru jazyků generovaných mCDG gramatikami.

2 Cooperating Distributed Grammar (CDG)

Bývá nesnadné přeložit do našeho jazyka anglické technické termíny, tady lze název *Cooperating Distributed Grammars* (CDG) přeložit na *splupracující rozložené gramatiky* a snad to vystihuje i jejich princip. Začneme jejich definicí podle [1]:

* Faculty of Information Technology, Technical Univeristy of Brno, Božetěchova 2, Brno, 61200, The Czech Republic

CDG je konstrukce:

$$G = (N, T, S, P_1, P_2, \dots, P_k) \tag{1}$$

kde N, T, S mají obvyklý význam a $P_i, 1 \leq i \leq k$ jsou konečné množiny bezkontextových přepisovacích pravidel nad $N \cup T$, tedy tvaru $A \rightarrow (N \cup T)^*$, kde $A \in N$. Množiny P_i jsou nazývány *komponenty gramatiky G*. Písmenem k bude označován počet komponent v gramatice.
V [1] je popisováno několik způsobů (módů) generování vět a diskutuje se jejich vliv na generovací sílu gramatiky. V našem případě se zaměříme na tak zvaný *t(terminating)-mód*. Princip generování vět v *t-módu* ($\Rightarrow^t$) z CDG popíšeme nejdříve neformálně.
Kontextem generování věty je vždy rozpracovaná větná forma a číslo jedné aktivní komponenty, ostatní komponenty se nezahrnují do přepisování. V daném kontextu lze aplikovat na přepisování pouze pravidla aktivní komponenty a komponenta je aktivní, dokud existuje aplikovatelné přepisovací pravidlo aktivní komponenty. Není-li už v aktivní komponentě aplikovatelné pravidlo, aktivitu přebere jiná libovolná komponenta. Uvidíme, že tento princip je klíčový pro CDG.
Formálně [1] je jazyk generovaný CDG gramatikou v *t-módu*:

$$L_t(CDG) = \{w | w \in T^*, S \Rightarrow^t_{P_{i_1}} w_1 \Rightarrow^t_{P_{i_2}} ... \Rightarrow^t_{P_{i_m}} w_m = w, m \geq 1, 1 \leq i_j \leq k, 1 \leq j \leq m\} \tag{2}$$

Příklad 1:

$$G = (\{S, A, C, X, Y\}, \{a, b, c\}, S, P_1, P_2, P_3) \tag{3}$$

kde

$P_1 = \{S \rightarrow AC, X \rightarrow A, Y \rightarrow C\}$,
$P_2 = \{A \rightarrow aXb, C \rightarrow cY\}$,
$P_3 = \{A \rightarrow ab, C \rightarrow c\}$

ukázka generování věty

$$S \Rightarrow^t AC \Rightarrow^t aXbC \Rightarrow^t aXbcY \Rightarrow^t aAbcY \Rightarrow^t aAbcC \Rightarrow^t aabbcC \Rightarrow^t aabbcc$$

Tato gramatika (3) generuje jazyk $\{a^n b^n c^n \mid n \geq 1\}$. V [1] je dokázáno, že třída jazyků generovaných CDG je ekvivalentní třídě rekurzivně vyčíslitelných jazyků (RE, třída 0 podle Chomského hierarchie).

Poznámka: všimněme si nedeterminismu gramatiky způsobeného pravidly A_2 a A_3. C-pravidla nedeterminismus ale v podstatě nezpůsobují, protože o výběru C-pravidla rozhodne kontext zavedený A-pravidlem.

3 Modifikovaná CDG (mCDG)

V průběhu následujícího textu budeme používat gramatiku (4) jako demonstrační příklad. Budeme definovat modifikovanou CDG (mCDG) gramatiku a pro ni specifickou relaci derivace.

Demonstrační gramatika (již deterministická)

$$G = (\{S, A, C, X, Y\}, \{a, b, c\}, S, P_1, P_2, P_3) \tag{4}$$

$P_1 = \{S \rightarrow AC, X \rightarrow A, Y \rightarrow C\}$
$P_2 = \{A \rightarrow aXb, C \rightarrow cY\}$
$P_3 = \{A \rightarrow d, c \rightarrow d\}$

Tato gramatika (4) generuje jazyk $\{a^n db^n c^n d | n \geq 1\}$. Tento jazyk nesporně přesahuje rámec třídy bezkontextových jazyků.

3.1 Úvodní definice

Zatím nepovolíme výskyt e-pravidel, protože to komplikuje následující definice. Pro navigaci gramatikou zavedeme návěští pravidel (labels). Komponentu budeme chápat jako seřazenou posloupnost pravidel. Pak můžeme psát: $P_{i,j} : A$ je j-té pravidlo i-té komponenty.
Kontext derivace (pracovní větná forma) je V^+, kde

$$V \in (T \cup (N \times Z^*)) \tag{5}$$

$$Z = \{1, .., k\} \tag{6}$$

Z^* je posloupnost (řetězec) čísel, které budou odpovídat číslům komponent.

Příklady větných forem:

$abAbC$, $abAbC(1, 2)$

Pro další definice je nutné upřesnit relaci derivace mCDG gramatiky a zavést *levou derivaci* (je nezbytná pro deterministickou sekvenční syntaktickou analýzu). Původní definice derivace připouští přepisování libovolného nonterminálu. Proto je definice levé derivace komplikovaná - do větné formy se za nejlevějším nonterminálem musí zaznačit odložené přepsání a zachovat tak kontext mezi pravidly.

Definice 1: Aplikace čísla komponenty na větnou formu

Pomocná funkce $apply : V^* \times Z \to V^*$ aplikuje číslo komponenty i na větnou formu w tak, že:

$apply(e, i) = e$
$apply(ab, i) \bigwedge (a \in T) = a.apply(b, i)$
$apply(ab, i) \bigwedge (a = N_x() \bigwedge N_x \in P_i) = a(i).apply(b, i)$
$apply(ab, i) \bigwedge (a = N_x(list)) = a(list, i).apply(b, i)$

Pro zápis volání funkce *apply(w,n)* zavedeme notaci $w[n]$. Aplikaci řetezce čísel na větnou formu $w[a_1, ..., a_b]$ je $(((w[a_1])[a_2])...)[a_b]$, tedy se postupně aplikují čísla a_1 až a_b.

Příklad 2: (používá gramatiku (4))

$abc[1] = abc$
$abC[1] = abC$
$abC(2, 1)[2] = abC(2, 1, 2)$

Definice 2: Levá derivace v mCDG

Levá derivace $v_1 \Rightarrow_L v_2; v_1, v_2 \in V^+, \alpha \in T^*, \beta \in V^*$. Řetězec v_1 se derivuje na v_2, pokud platí jedna z možností:

1. $v_1 = \alpha A \beta$ a existuje $i, j : P_{i,j} : A \to \gamma$, pak $v_2 = \alpha\gamma[i]\beta[i]$
   (příklad: $AC \Rightarrow_L aXbC(2)$)
2. $v_1 = \alpha A(i, tail)\beta$ a existuje $j : P_{i,j} : A \to \gamma$, pak $v_2 = \alpha\gamma[i, tail]\beta[i]$
   (příklad: $aadbbcC(2, 1, 3) \Rightarrow_L aadbbccY(2, 1, 3)$)
3. $v_1 = \alpha A(i, tail)\beta$ a neexistuje $j : P_{i,j} : A \to \gamma$, pak $v_2 = \alpha A(tail)\beta[i]$
   (příklad: $aadbbcY(2, 1, 3) \Rightarrow_L aadbbccY(1, 3)$)

Případ 1 modeluje přepnutí kontextu z jedné komponenty na druhou, druhý případ definuje vynucené použití pravidla z konkrétní komponenty a poslední případ je ignorování kontextu a přechod na další.

Existence levé derivace je klíčová pro deterministickou syntaktickou analýzu [3, 5]. Následuje ukázka generování věty levou derivací (zjednodušeně):

$S \Rightarrow_L AC \Rightarrow_L aXbC(2) \Rightarrow_L aAbC(2,1) \Rightarrow_L aaXbbC(2,1,2) \Rightarrow_L aaAbbC(2,1,2,1) \Rightarrow_L aadbbC(2,1,2,1,3) \Rightarrow_L aadbbcY(1,2,1,3) \Rightarrow_L aadbbcC(2,1,3) \Rightarrow_L aadbbccY(1,3) \Rightarrow_L aadbbccC(3) \Rightarrow_L aadbbccd$

Definice 3: Modifikovaná CD gramatika (mCDG) je konstrukce $G = (N, T, S, P_1, ..., P_k)$, kde

1. komponenty neobsahují e-pravidla
2. generování vět probíhá levou derivací podle definice 2

Jazyk generovaný mCDG je $L_l(mCDG) = \{w \in T^+ | S \Rightarrow_L^+ w\}$.
Zatím nemluvíme o deterministických CDG, v každém případě však podle našich posledních výzkumů platí, že $L_l(mCDG) \subset L(CDG)$. Existují konkrétní gramatiky CDG, které při levé derivaci nedávají jednoznačný výsledek (negenerují stejný jazyk jako při generování v t-módu). V současné době pracujeme na vymezení rozsahu třídy mCDG jazyků a na stanovení podmínek pro konkrétní gramatiku, za jakých platí naše levá derivace.
Na závěr této kapitoly shrneme, že třída $mCDG$ je třída jazyků generovaných CDG gramatikou levou derivací (tedy mCDG gramatikou).

4 Model syntaktické analýzy jazyků generovaných mCDG

Na úvod podotkněme, že pracujeme s třídou jazyků tvořící hranice vyčíslitelnosti. Přinejmenším jsou to jazyky, které nejsou všechny rozhodnutelné a tedy syntakticky analyzovatelné. Pro začátek se pokusíme vymezit jejich deterministickou podtřídu CDG_{det}.
CDG jsou metodicky gramatiky odvozené od bezkontextových gramatik. Proto byla snaha na ně použít metody analýzy bezkontextových jazyků rozšířené o řízení kontextem. Chtěli jsme zachovat filosofii těchto gramatik a tou je především vypadat na první pohled bezkontextově.
Jako výchozí bod pro náš syntaktický analyzátor použijeme model analyzátoru bezkontextových LL(1) gramatik - rekurzivní sestup. Stejně jako u LL(1) gramatik, musí i mCDG gramatika, analyzovatelná touto metodou, splňovat kritéria determinismu (podmínka 7 pro LL(1); připomeneme, že se nepřipouští e-pravidla).

$$A \rightarrow a_1|a_2|...|a_n \Rightarrow \forall i,j \in \{1,2,...,n\}, i \neq j : FIRST(a_i) \cap FIRST(a_j) = \emptyset \qquad (7)$$

4.1 Determinismus v mCDG

Potřebujeme definovat množinu $FIRST$ pro (m)CDG. Pro determinismus CDG gramatiky musí určitě platit, že každá komponenta je $LL(1)$. Navíc musí platit prázdnost průniků množin $FIRST$ pravých stran pravidel stejného nonterminálu přes všechny komponenty gramatiky (globální FIRST podmínka).
Vyšetřování determinismu je relativně komplikované: můžeme rozlišovat gramatiky *staticky* deterministické (platí globální podmínka FIRST) a *dynamicky* deterministické (z obecně komplikované analýzy stavového prostoru generování vět zjistíme nekoliznost souborů pravidel stejné levé strany).
Zavedeme množinu $gFIRST(w,n), w \in N$ (n je celé číslo, označuje číslo komponenty) .

Definice 4: Mějme mCDG gramatiku $G = (N, T, S, P_1, ..., P_k)$. Množina $gFIRST(w,n)$ je definována:

1. $n > 0$, pak $gFIRST(w,n) = \{FIRST(right(P_{n,i})) | left(P_{n,i}) = w; i \in \{1,...,|P_n|\}^1$
2. $n = 0$, pak $gFIRST(w,0) = \bigcup_{i=1}^k gFIRST(w,i)$

---

[1] $right(P_{n,i}) = \gamma$,
$left(P_{n,i}) = A$, kde $P_{n,i} \in P_n$ a $P_{n,i} : A \rightarrow \gamma$

4.2 Syntaktická analýza CDG rekurzivním sestupem

Rozdíl je ale podstatný: je nutné dopředu rozpracovávat větnou formu napravo od aktuálně přepisovaného nonterminálu (v programu: od aktuální procedury), což je definováno v levé derivaci.
Pro snadnou implementovatelnost analyzátoru v programovacích jazycích je základ stejný: rekurzivní sestup sledující levou derivaci, zbytek větné formy sledující průchody různými komponentami je umístěn explicitně v určité datové struktuře s charakterem seznamu (pásky).

4.3 Velikost třídy jazyků analyzovatelných uvedenou metodou

Pravděpodobně platí $CDG_{det} \subset CDG$. Ještě nevíme, jestli $mCDG = CDG_{det}$ nebo $mCDG \subset CDG_{det}$.

5 Konstrukce překladače

Mějme mCDG gramatiku $G = (N, T, S, P_1, ..., P_k)$ a jazyk L_G generovaný touto gramatikou. Syntaktický analyzátor (a pak překladač) každého takového jazyka L_G je složen ze tří základních částí:

1. **datová:** obsahuje kompletní předpis gramatiky G (je potřeba pro jádro)
2. **jádro:** obsahuje speciální pomocné funkce pro řízení syntaktické analýzy (apply, insert,call)
3. **překladač:** z gramatiky vygenerovaný soubor procedur

Kontextem překladu je dvojice $(input, wf)$, kde $input$ je sekvenčně zpracovávaná vstupní věta a wf je pracovní pomocná generovaná větná forma.

5.1 Jádro analyzátoru

Jádro analyzátoru tvoří tyto funkce pracující nad pomocnou větnou formou wf
apply(wf,n) aplikuje číslo komponenty n na větnou formu wf (podle definice 1)
insert(w,i) simuluje derivaci větné formy $A(list)\beta \Rightarrow_L^{P_{i,j}} \gamma[list]\beta[list]$
call() zprostředkovává volání procedury překladače.

$vf = A()\beta \Rightarrow$ `volej proceduru` A `s parametrem` 0, `A(0)`
$vf = A(i, tail)\beta \wedge A$ `existuje v` $P_i \Rightarrow$ `volej` $A(i)$
$vf = A(i, tail)\beta \wedge A$ `neexistuje v` $P_i \Rightarrow vf := A(tail)\beta$ `a znovu call`

5.2 Generování analyzátoru

Každý nonterminál gramatiky je v analyzátoru zastoupen jednou procedurou. Procedure zahrnuje všechna přepisovací pravidla nonterminálu ze všech komponent.

$$A \rightarrow (a_{11}|a_{12}|...)|(a_{21}|a_{22}|...)|...|(a_{k1}|a_{k2})$$

. To znamená, že pro každý nonterminál A se vygeneruje procedura:

```
void A(int n, Attr attrs) {

if (n==0) allA;

else

if (n==1) allA1;

else

...

if (n==k) allAk;
}
```

s parametry:

n číslo komponenty (číslovány od jedné) nebo nulu (výběr přes všechny komponenty)
attrs může obsahovat zděděné atributy v libovolné formě

Každé přepisovací pravidlo i-té komponenty $A \rightarrow w,\ w = a_1 a_2 ... a_b$ se převede na

```
insert(w,i);
code(a_1);
code(a_2);
...
code(a_b);
```

kde $code(a_r)$ je

expect(a_r) je-li $a_r \in T$
call() je-li $a_r \in N$

Takto sestavený analyzátor je schopen (přinejmenším) přijímat mCDG jazyky.

6 Sumarizace a otevřené problémy

Byla navržena metoda syntaktické analýzy CD gramatik. Funkčnost metody byla ověřena prakticky na příkladu gramatiky (4). Výhoda navržené metody je zcela zřejmě v její jednoduché implementovatelnosti. Možná nevýhoda je snad v paměťové náročnosti při budování seznamu odložených derivací, tady je další bod pro optimalizace a zdokonalování. Jádro této metody může být použito při návrhu analyzátorů založených na příbuzných gramatikách se srovnatelnou generovací silou (Programmed Grammars, Random Context Grammars a dalších).

6.1 Otevřené problémy

1. Lze ke každé nedeterministické CDG zkonstruovat její ekvivalentní deterministickou gramatiku?
2. Jaký je vztah třídy mCDG jazyků k důležitým podtřídám třídy jazyků typu 0, jako například třída rekurzivních jazyků

Poznámka: Tato práce je podporována grantem FRVŠ: *Moderní metody překladu jazyků*, rok 2002.

Bibliography

1. Dassow, J., Paun, Gh., Rozenberg, G.: Grammar Systems. Kapitola 4 v Handbook of Formal Languages, Volume 2, Springer, London, 1997
2. Mitrana, V.: Parsability approaches in CD Grammar Systems, In: Proceedings of International Workshop Grammar Systems 2000 July 3 - 7, 2000 Bad Ischl, Austria
3. Mihalache, V., Mitrana, V.: Deterministic Cooperating Distributed Grammar Systems, In: Turku Centre for Computer Science, TUCS Technical Report No 63, ISBN 951-650-886-3, ISSN 1239-1891
4. Middendorf, M.: Supersequences, Runs and CD Grammar Systems, In: J. Dassow, A. Kelemenova (Eds.), Developments in Theoretical Computer Science, 101-114, (Topics in Computer Science, Vol. 6), Gordon and Breach, Amsterdam (1994) 101-114.
5. Fernau, H.: Regulated Grammars under Leftmost Derivation, In: Grammars: A Journal of Mathematical Research on Formal and Natural Languages, Volume 3, 2000