PARSING BASED ON STATE GRAMMARS

Petr NAVRÁTIL, Master Degree Programme (5)
Dept. of Information Systems, FIT, BUT
E-mail: xnavra16@stud.fit.vutbr.cz

Supervised by: Dr. Alexander Meduna

ABSTRACT

This paper introduces a concept of a state grammar, LL(1) state grammar and new parsing algorithm based on this grammar. A state grammar is a context-free grammar with states where the class of allowable derivations is restricted by the states requirements.

1 ÚVOD

Některé syntaktické struktury, které se nalézají v mnoha programovacích jazycích nelze popsat pomocí bezkontextových gramatik. Toto nás vede k přemýšlení o silnějších gramatikách, které povolí specifikaci uvnitř bezkontextových gramatik jako pořadí, ve kterém budou používána pravidla v průběhu generování. V této práci je uvedena stavová gramatika, jedná se o řízenou gramatiku. V tomto případě jde o bezkontextovou gramatiku řízenou v podstatě automatem, který nám pomocí svých stavů určuje, kdy je při generalizaci možno použít které pravidlo gramatiky.

2 LL(1) STAVOVÉ JAZYKY

Pro syntaktickou analýzu stavových jazyků, jsme museli omezit třídu stavových jazyků na LL(1) stavové jazyky, abychom mohli jednoznačně určit podle symbolu vstupního řetězce, které pravidlo použít.

Definice 2.1:

Stavová gramatika je 6-tice $G = (K, V, \sum, P, p_0, \sigma)$, kde

K je neprázdná množina stavů

V je konečná množina symbolů gramatiky

$\sum$ je konečná množina terminálních symbolů a je podmnožinou V

σ je element z $V - \sum$

p_0 je element z K, jde o počáteční stav gramatiky

P je podmnožina z $K \times (V - \sum) \times K \times V^+$

Element (p, ξ, q, u) z P se nazývá stavové pravidlo (nebo jen pravidlo) a obvykle se zapisuje v tomto tvaru (p, ξ) → (q, u). Element z V-Σ, je nazýván proměnná (terminál) a ξ je aplikovatelná pod stavem p, je-li (p, ξ) → (q, u) v množině pravidel P pro nějaké q z K a u z V^+.

Definice 2.2:

Je dána stavová gramatika G = (K, V, ∑, P, p_0, σ), nechť => je relace na K x V^+ definovaná takto: nechť p je z K a ω = xξy z V^+. Jestliže ξ je nejlevější výskyt aplikovatelných proměnných (terminálů) ve ω pod p a (p, ξ) → (q, u) je v P, pak píšeme (p, xξy) => (q, xuy). Jestliže ξ je j-tá proměnná (terminál) v ω, pak někdy píšeme $=>^j$ namísto =>.

Pro definici LL(1) stavové gramatiky potřebujeme množiny FIRST, EMPTY, FOLLOW a PREDICT, které vychází ze stavové gramatiky a jejich obsahem jsou symboly stavové gramatiky.

Je-li α řetězec symbolů gramatiky a (q,α) $=>^*$ (p,ε), kde p,q jsou z K, potom množina EMPTY(α) obsahuje symbol ε, jinak je tato množina prázdná.

Je-li α řetězec symbolů gramatiky, potom FIRST(α) je množina terminálních symbolů, jimiž mohou začínat řetězce derivované z α.

Množinu FOLLOW(A), kde A je nonterminální symbol, definujeme jako množinu všech terminálních symbolů a, které se mohou vyskytovat bezprostředně vpravo od A v nějaké větné formě. Může-li být A nejpravějším symbolem v nějaké větné formě, je ve FOLLOW(A) rovněž symbol $, který představuje konec vstupního řetězce.

Množinu PREDICT pak pomocí předchozích množin definujeme takto:

Jestliže (r, X) =>* (p, ε) (tedy EMPTY(X) = {ε}),

pak PREDICT((q, A) → (r, X)) = FIRST(X) ∪ FOLLOW(1,A)

jinak PREDICT((q, A) → (r, X)) = FIRST(X)

Definice 2.3:

Nechť G=(K, V, Σ, P, p_0, S) je stavová gramatika. G je LL(1) stavová gramatika, jestliže pro každý pár pravidel (q, A) → (r, x) ∈ P a (q, A) → (p, y) ∈ P, kde y ≠ x, platí: PREDICT((q, A) → (r, x)) ∩ PREDICT((q, A) → (p, y)) = ∅.

Definice 2.3:

Jazyk generovaný LL(1) stavovou gramatikou G, je pak definován jako

$$L(G) = \{\omega \text{ z } \Sigma^+ \mid (p_0, \sigma) =>^* (q, \omega) \text{ pro libovolné q z K}\}$$

tomuto jazyku říkáme LL(1) stavový jazyk.

3 PREDIKTIVNÍ TABULKOU ŘÍZENÁ SYNTAKTICKÁ ANALÝZA LL(1) STAVOVÝCH JAZYKŮ

Syntaktický analyzátor pracuje se vstupní pamětí obsahující analyzovaný řetězec zakončený speciálním symbolem $. Dále se zásobníkem, který obsahuje posloupnost symbolů gramatiky, přičemž dno zásobníku je indikováno symbolem #. Z našeho zásobníku nebudeme brát vždy jen první prvek, ale i další v pořadí. Algoritmus ke své činnosti potřebuje také

rozkladovou tabulku což je dvojrozměrné pole $M[(q_i,A_i),a]$, kde A_i je nonterminál ($A_i \in V - \Sigma$ pro všechna i), q_i je stav gramatiky ($q_i \in K$ pro všechna i) a a je terminální symbol ($a \in \Sigma$) nebo symbol \$, prvkem této tabulky je buď pravidlo typu $((q_i, A_i) \rightarrow (p, y))$ to v případě je-li $a_j \in PREDICT((q, A_i) \rightarrow (p, y))$ nebo error. Potom je také třeba uchovávat aktuální (v dané fázi aktivní) stav gramatiky.

Samostatnou částí analyzátoru je řídící program, který opakovaně prohlíží symbol X ze zásobníku a současný vstupní symbol a, na základě nichž se rozhoduje o své další činnosti. Algoritmus rozhodování je následující:

Do zásobníku vlož #S a do q ulož počáteční stav;

Nechť X je symbol ze zásobníku, a je vstupní symbol a q aktuální stav;

Dokud zásobník $\neq \varepsilon$ prováděj

Výskyty X:

X = #: je-li a = \$ pak úspěšný konec (prázdný zásobník)
jinak chyba

$X \in T$: je-li X=a pak X vyjmi ze zásobníku a vezmi za a další symbol vstupního řetězce (ne nutně symbol následující v pořadí)
jinak chyba

$X \in N$: neexistuje-li pravidlo na jehož levé straně je (q,X) pak vezmi za X následující symbol ze zásobníku. Hloubka zásobníku, ve které se nachází X může být rovna maximálně n, které udává n-omezení LL(1) stavové gramatiky.

jinak je-li v $M[(q,X),a]$ nějaké pravidlo $(q,X) \rightarrow (r,b\beta) \in P$ pak
je-li $b \in \sum$ a a=b pak vezmi za a další symbol vstupního řetězce a vyměň X v zásobníku za reversované(β)

jinak vyměň X v zásobníku za reversované(β) a reversované(b)

a do q vlož nový stav r

jinak chyba

4 ZÁVĚR

Pomocí LL(1) stavových gramatik nelze popsat celou třídu kontextových jazyků a dokonce asi jen část třídy bezkontextových jazyků. Ale myslím, že je dobré dokázat jednoduchou syntaktickou analýzu alespoň části třídy kontextových jazyků.

LITERATURA

[1] Meduna, A.: Automata and Languages, Springer, London 2000

[2] Češka, M.: Teoretická informatika, FIT VUT, Brno, 2002

[3] Kasai, T.: An hierarchy between context-free and context-sensitive Languages, Waseda University, Tokyo, 1969