SYNTAX ANALYSIS BASED ON STATE GRAMMARS

Filip Konečný, Bachelor Degree Programme (3)
Dept. of Information Systems, FIT, BUT
E-mail: xkonec39@stud.fit.vutbr.cz

Supervised by: Prof. Alexander Meduna

ABSTRACT

This paper discusses an approach to syntax analysis that is based on state grammars and suggests a possible application of state grammars in the field of compilers - as a ground of a syntax analyzer.

First of all, properties of state grammars are described and reasons for their application are given. Then, the formalism of a state grammar and an example of a state grammar are shown. Finally, the method of a syntax analysis based on state grammars is described.

1 ÚVOD

Bezkontextové gramatiky jsou po dlouhou dobu základem přístupu k syntaktické analýze programovacích jazyků. V průběhu několika posledních dekád byly představeny některé modifikace (maticové gramatiky, programované gramatiky a další). Jednou z nich je i *stavová gramatika* poprvé představena v [1]. Stavová gramatika je rozšířením gramatiky bezkontextové, které spočívá v zahrnutí množiny stavů do definice gramatiky. Při derivaci se tedy každá větná forma nachází v určitém stavu. U stavové gramatiky dochází při derivaci větné formy k situacím, kdy se nepřepisuje nejlevější neterminál, nýbrž neterminál, který se nachází hlouběji ve větné formě. Stavy jsou zde příčinou této situace a pravidla nástrojem, který si tuto skutečnost vynutí. Princip zmíněného přinucení spočívá v tom, že pro žádný z předchozích neterminálů neexistuje za daného stavu větné formy pravidlo.

V [1] byl zaveden termín stavová gramatika *stupně n*, s jehož využitím se definuje nekonečná hierarchie tříd stavových jazyků. Tato hierarchie začíná třídou bezkontextových jazyků a je omezena třídou kontextových jazyků.

Zmíněné vlastnosti stavových gramatik zvyšují jejich vyjadřovací sílu. Důsledkem je tudíž i zvýšení síly syntaktické analýzy, která na stavových gramatikách staví. Syntaktická analýza založená na stavových gramatikách totiž umožní ve větší míře monitorovat kontext vstupního řetězce.

2 STAVOVÁ GRAMATIKA

Definice *Stavová gramatika* je pětice $G = (V, W, T, P, S)$, kde V je úplná abeceda, konečná množina, W neprázdná konečná množina stavů, $T \subset V$ množina terminálů, $P \subset (W \times (V - T) \times W \times V^+)$ množina pravidel, S startovací neterminál; $S \in V - T$.

Místo $(p, X, q, v) \in P$, kde $p, q \in W, X \in (V - T), v \in V^+$ obvykle píšeme $(p, X) \to (q, v)$. Řekneme, že neterminál X je *aplikovatelný* ve stavu p, pokud $(p, X) \to (q, v) \in P$ pro libovolné $q \in W$ a $v \in V^+$. Dále nechť $w = xXy, w \in V^+$. Pokud $(p, X) \to (q, v) \in P, x, y \in V^+$ a X je nejlevější aplikovatelný neterminál ve stavu p, potom G dělá derivační krok z (p, xXy) do (q, xvy), symbolicky $(p, xXy) \Rightarrow (q, xvy)$. Pokud platí, že neterminál X je ve w nejvýše n-tý zleva, říkáme, že derivační krok je *n-ohraničený* a píšeme $(p, xXy) \ {}_n\Rightarrow (q, xvy)$. Nechť $\alpha, \beta \in (W \times V^+)$. Pro vyjádření, že každý derivační krok v derivaci je n-ohraničený, píšeme $\alpha \ {}_n\Rightarrow^* \beta$ a říkáme, že derivace je *n-ohraničená*.

Stavový jazyk: $L(G) = \{w \in T^+ | (p, S) \Rightarrow^* (q, w), q, p \in W\}$.

Stavový jazyk stupně n: $L(G, n) = \{w \in T^+ | (p, S) \ {}_n\Rightarrow^* (q, w), q, p \in W\}$.

Stavová gramatika je *stupně n* (n je celé kladné číslo), pokud platí $L(G, n) = L(G)$.

Nechť pro každé kladné celé číslo n značí $\mathcal{L}_n$ třídu stavových jazyků stupně n. Pak následujícím způsobem definujeme nekonečnou hierarchii jazyků: $\mathcal{L}_1 \subset \mathcal{L}_2 \subset \cdots \subset \mathcal{L}_\infty \subset \mathcal{L}_\omega$, kde $\mathcal{L}_1$ je třída bezkontextových jazyků bez ε-pravidel, $\mathcal{L}_\omega$ je třída kontextových jazyků.

Příklad 1 $G = (V, W, T, P, S); T = \{a, b, c\}, V = \{a, b, c, S, C, X\}, W = \{s_0, s_1, s_2\}, P = \{1 : (s_0, S) \to (s_0, XC), 2 : (s_0, X) \to (s_1, aXb), 3 : (s_1, C) \to (s_0, cC), 4 : (s_0, X) \to (s_2, ab), 5 : (s_2, C) \to (s_2, c)\}$. Jazyk generovaný gramatikou $L(G) = \{a^n b^n c^n : n > 0\}$. Jedná se o stavový jazyk stupně 2. Tedy $L(G) = L(G, 2)$.

Příklad derivace řetězce gramatikou G: $(s_0, S) \Rightarrow (s_0, XC)[1] \Rightarrow (s_1, aXbC)[2] \Rightarrow (s_0, aXbcC)[3] \Rightarrow (s_1, aaXbbcC)[2] \Rightarrow (s_0, aaXbbccC)[3] \Rightarrow (s_2, aaabbbccC)[4] \Rightarrow (s_2, aaabbbccc)[5]$.

Příklad 2 $G_2 = (V, W, T, P, S); T = \{a, b\}, V = \{a, b, S, X, Y\}, W = \{s_0, s_1, s_2, s_3, s_4\}, P = \{1 : (s_0, S) \to (s_0, XY), 2 : (s_0, X) \to (s_1, aX), 3 : (s_1, Y) \to (s_0, aY), 4 : (s_0, X) \to (s_2, bX), 5 : (s_2, Y) \to (s_0, bY), 6 : (s_0, X) \to (s_3, a), 7 : (s_3, Y) \to (s_0, a), 8 : (s_0, X) \to (s_4, b), 9 : (s_4, Y) \to (s_0, b)\}$.

Jazyk generovaný gramatikou $L(G_2) = \{ww : w \in \{a, b\}^+\}$. Jedná se o stavový jazyk stupně 2. Tedy $L(G_2) = L(G_2, 2)$.

Příklad derivace řetězce gramatikou G_2: $(s_0, S) \Rightarrow (s_0, XY)[1] \Rightarrow (s_1, aXY)[2] \Rightarrow (s_0, aXaY)[3] \Rightarrow (s_2, abXaY)[4] \Rightarrow (s_0, abXabY)[5] \Rightarrow (s_4, abbabY)[8] \Rightarrow (s_0, abbabb)[9]$.

3 SYNTAKTICKÁ ANALÝZA ZALOŽENÁ STAVOVÝCH GRAMATIKÁCH

Metoda Navržená metoda syntaktické analýzy založené na stavových gramatikách používá *hluboký zásobníkový automat* (viz [2]). Jedná se o obecnou syntaktickou analýzu, která využívá *backtracking*. Vedle pravidel stavové gramatiky (expanzní pravidla) implicitně používá porovnávací pravidla (nejsou v analyzátoru fyzicky reprezenována), která

najdou uplatnění v momentě, kdy je na vrcholu zásobníku terminál. Dále se předpokládá, že lexikální analyzátor dodává syntaktickému analyzátoru tokeny. Syntaktický analyzátor zaznamenává svou činnost na zvláštním zásobníku, aby mohl dle principů backtrackingu rekonstruovat svou dřívější konfiguraci. Schéma algoritmu, který analyzátor používá, je následující:

```
syntax_ok := false;
do {
   if ( na vrcholu hlubokého zásobníku je terminál ) {
      if (na vstupu není žádný další token) {
         vyjmi vrchol zásobníku history a proveď rekonstrukci konfigurace
      } else {
         if (tento token je stejný jako terminál na vrcholu hlubokého zásobníku) {
            aplikuj porovnávací pravidlo a
            na zásobník history vlož informaci o jeho provedení
         } else {
            vyjmi vrchol zásobníku history a proveď rekonstrukci konfigurace
         }
      }
   } else {
      if (pro první aplikovatelný neterminál existuje první/další expanzní pravidlo)
         proveď tuto expanzi v hlubokém zásobníku a
         na zásobník history vlož informaci o jejím provedení
      } else {
         vyjmi vrchol zásobníku history a proveď rekonstrukci konfigurace
      }
   }
   if (hluboký zásobník je prázdný) syntax_ok := true;
} while( (zásobník history není prázdný) and not(syntax_ok) );
```

Poznámky k metodě V definici stavové gramatiky není zahrnuta informace o *počátečním stavu*. Počáteční stav je takový stav, který je v relaci se startovacím neterminálem u levé strany libovolného pravidla stavové gramatiky. Proto syntaktický analyzátor nejprve tyto stavy zjistí a postupně s nimi provádí výše popsanou metodu syntaktické analýzy. Analýza skončí nalezením prvního řešení, kdy analyzátor oznámí, že větná forma patří do jazyka. K provedení metody pro všechny počáteční stavy tedy nemusí dojít.

REFERENCE

[1] Kasai, T.: An Hierarchy Between Context-Free and Context-Sensitive Languages, Journal of Computer and System Sciences 4, 492-508, 1970

[2] Meduna, A.: Deep Pushdown Automata, Acta Informatica, roč. 2006, č. 98, DE, s. 114-124, ISSN 0001-5903