PARALLEL STATE GRAMMAR PARSING

Miroslav Paulík
Bachelol Degree Programme (4), FIT BUT
E-mail: xpauli00@stud.fit.vutbr.cz

Supervised by: Alexander Meduna
E-mail: meduna@fit.vutbr.cz

Abstract: This article discusses the possibility of parallel state grammar parsing. Briefly summarizes the basic features of state grammar with emphasis on the problems that entails analysis of such grammars. It also describes the principle of classical analysis, sets out its key points and then outlines alternative solutions for them by using parallel processing.

Keywords: state grammars, parallel parsing, backtracking

1 PREREKVIZITY

Pro plné porozumnění tohoto článku se předpokládá základní znalost principů formálních jazyků, gramatik, syntaktické analýzy a matematických operací s nimi spojených [2, 3].

2 STAVOVÉ GRAMATIKY

Stavová gramatika [1, 4] je kontextová gramatika obsahující speciální stavový mechanismus, jež se velmi podobá konečným automatům [4]. Jednotlivé stavy takovéto gramatiky slouží pro omezení výběru pravidel, jež mohou být v dané konfiguraci, a tedy i v daném stavu, použity. To, spolu s konvencí, že derivace je uplatněna na takový nejlevější neterminál, pro který existuje v daném stavu pravidlo (tedy tzv. "zanořený"neterminál), což umožňuje zvýšit derivační sílu až po rekurzivně vyčíslitelné jazyky.

Takové vlastnosti ovšem způsobují mnoho problémů při analýze. Je nutné rozlišovat hloubku zanoření. Zatímco pro derivaci zcela nejlevějších neterminálů je možné aplikovat pravidla v podstatě stejně, jako je tomu u LL(1) gramatik, pro zanořenou derivaci to však již neplatí. V případě zanořené derivace totiž není možné určit, jakou část vstupní větné formy reprezentuje. Tento problém je řešen postupnou aplikací všech pravidel a uložením vzniklých konfigurací. Pro úplnou analýzu v případě chybového vstupního řetězce je nutné ověřit postupně všechny konfigurace. Ty se však mohou průběžně hromadit, proto je nutné přidat mechanismus pro efektivní procházení - backtracking.

3 ASPEKTY ANALÝZY

Z použití algoritmu na prohledávání stavového prostoru vyplývá, že dochází k redundantním žádostem o lexikální analýzu jednotlivých vstupních symbolů. To lze optimalizovat předzpracováním lexikální analýzy. Nevýhodou je ale nutnost zpracovat celý vstup i případě, že dojde k syntaktické chybě a lexikální analýza zbylých symbolů by tedy nebyla nezbytně nutná. Další nevýhodou je také požadavek na paměťové nároky, neboť vstupní řetězec může být neomezeně dlouhý. To je ovšem kompenzováno rychlým přístupem k těmto operacím, redukncí duplicitních analýz a hlavně to umožní další optimalizaci (např. terminálová predikce [5]), jež by bez takových dat nebyla možná.

Po takové úpravě je tedy nutné jen ukládat konfiguraci, realizovat mechanismus návratu k předchozím konfiguracím a následně po jednom kroku procházet možné konfigurace. Tento postup je tedy značně

pomalý.

4 KONCEPT PARALELNÍ ANALÝZY

Paralelní zpracování by se také neobešlo bez úpravy lexikální analýzy. V tomto se od předchozí metody neliší. Mechanismus návratu, tzv. backtracking, by však byl nahrazen alternativním mechanismem, jež by spravoval všechny vzniklé větve a zajistil mezi nimi komunikaci, která je nanejvýš vhodná, neboť zabrání zbytečné analýze zbylých větví v případě úspěšného přijetí vstupního řetězce. Tento mechanismus správy je navíc možné využít jako formu detekce pravděpodobného výskytu chyby (každá chybná větev ukazuje na místo, kde daná větev skončila, což nemusí být skutečným místem vzniku chyby). Algoritmus vyhodnocování v rámci každé větve analýzy je popsán v algoritmu 1.

Nároky na zdroje by se tedy o moc nelišily oproti klasickému řešení, avšak lze očekávat mnohem rychlejší analýzu díky separovanému zpracování na více jádrech dnešních procesorů. Nicméně je nutné podotknout, že překlad takovou metodou implementace je 3-fázový. Nejprve je nutná lexikální analýza, posléze syntaktická analýza a až na základě jejího výsledku se buď provede nebo neprovede překlad kódu.

5 ZÁVĚR

Tento článek velmi zevrubně popisuje možnost, jak provést paralélní analýzu jazyka popsaného stavovými gramatikami. Oproti klasické analýze má paralelní zpracování předpoklady mít výrazně větší požadavky na zdroje. Reálná implementace v základní podobě popsané v tomto článku funguje, nicméně efektivita této implementace je stále ještě zkoumána.

REFERENCE

[1] Kasai, T.: An hierarchy between context-free and context-sensitive languages. Journal of Computer and System Sciences, 1970, 4.5: 492-508.

[2] Meduna, A.: Elements of Compiler Design, New York, US, T & F, 2008, s. 304, ISBN 978-1-4200-6323-3

[3] Meduna, A.: Automata and Languages: Theory and Applications [Springer, 2000], London, GB, Springer, 2005, s. 892, ISBN 1-85233-074-0

[4] Meduna, A., Zemek, P.: Regulated Grammars and Their Transformations, Brno, CZ, VUT v Brně, 2010, s. 239, ISBN 978-80-214-4203-0

[5] Paulík, M. STATE GRAMMARS PARSING OPTIMIZATION BASED ON TERMINAL PREDICTION, In: Proceedings of the 18th Conference STUDENT EEICT 2012, Brno, CZ, VUT v Brně, 2012

```
while TRUE do
    if zásobník symbolů je prázdný then
        if vstupní řetězec je prázdný then
            ukonči analýzu jako úspěšnou;
        else
            ukonči analýzu jako neúspěšnou;
        end
    else
        if vrchol zásobníku je terminál then
            if symbol na vrcholu zásobníku odpovídá prvnímu symbolu aktuálního vstupu then
                proveď přijetí symbolu;
            else
                ukonči analýzu jako neúspěšnou;
            end
        else
            nalezni takový nejlevější neterminál N, jež lze v dané konfiguraci derivovat;
            if existuje N then
                if N je právě symbol vrcholu zásobníku then
                    // nejlevější derivace;
                    aplikuj pravidlo;
                else
                    // zanořená derivace;
                    R = množina aplikovatelných pravidel;
                    r_0 ∈ R;
                    R = R − {r_0};
                    while R je neprázdné do
                        r_n ∈ R;
                        R = R − {r_n};
                        vytvoř novou větev a aplikuj v ní pravidlo r_n;
                    end
                    aplikuj pravidlo r_0;
                end
            else
                ukonči analýzu jako neúspěšnou;
            end
        end
    end
end
```

Algoritmus 1: Pseudoalgoritmus analýzy v rámci samotné větve.