STATE GRAMMARS PARSING OPTIMIZATION BASED ON TERMINAL PREDICTION

Miroslav Paulík
Bachelor Degree Programme (3), FIT BUT
E-mail: xpauli00@stud.fit.vutbr.cz

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

Abstract: This paper describes a mechanism of a predictive detection of dead branches during state grammars parsing. The prediction is based on a comparison between the remaining input, counts and relative positions of terminal symbols generated by application of state grammar rules. Simply put, it allows us to speed up the process of parsing.

Keywords: parsing, state grammars, terminal prediction, formal languages

1 ÚVOD

Tento článek nastiňuje novou optimalizační metodu z oblasti formálních jazyků a překladačů. Pro plné porozumnění podstaty optimalizace se předpokládá základní znalost principů formálních jazyků, gramatik, syntaktické a lexikální analýzy a matematických operací s nimi spojených [1, 2].

Terminálová predikce popisuje mechanismus optimalizace syntaktické analýzy založené na stavových gramatikách. Jejím účelem je redukce počtu aplikovaných pravidel gramatiky nutných pro zamítnutí vstupního řetězce a tím i snížení počtu větvení samotné analýzy. Ty jsou umožněny již z podstaty vlastností stavových gramatik, které povolují aplikaci pravidel na neterminální symboly, které nejsou nejlevější resp. nejpravější dle směru derivace [3]. V takových případech však není možné jednoznačně určit jediné správné pravidlo, které povede k přijetí vstupního řetezce. Je tedy nutné analýzu rozdělit na tzv. větve, aby bylo umožněno úplné prohledání. Každá větev aplikuje právě jedno z vyhovujících pravidel. Od tohoto místa probíhá analýza v dané větvi zcela nezávisle na ostatních.

2 PRINCIP PREDIKCE

Jak již bylo zmíněno v úvodu, cílem predikce je v konečném důsledku omezení počtu větví se snahou o zrychlení syntaktické analýzy. Predikce využívá informací získaných přímo z pravidel gramatiky a statistických dat poskytnutých lexikální analýzou. Ta je pro tuto optimalizaci klíčová, takže je celá vyžadována ještě před začátkem analýzy syntaktické. Predikci lze podle způsobu kontroly rozdělit do dvou kategorií - na bezkontextovou a kontextovou. Obě vyžadují různé informace z lexikální analýzy viz. kapitoly 3 a 4. Tyto získané údaje pak slouží k porovnání vygenerované posloupnosti terminálových symbolů s reálným vstupním řetězcem symbolů. V případě stavových gramatik, tedy gramatik umožňujících větvení analýzy, je navíc potřeba adekvátně upravit definici konfigurace tak, aby obsahovala aktuální stav tabulky s údaji o pozicích výskytů jednotlivých terminálů ve vstupním řetězci (dále jen tabulka).

3 BEZKONTEXTOVÁ PREDIKCE

Bezkontextová terminálová predikce je základní jednoduchou variantou této optimalizace. Využívá skutečného počtu výskytů jednotlivých typů terminálů ve vstupním řetězci a porovnává je s počty

symbolů v použitých pravidlech. Ke srovnání dochází vždy až po aplikaci pravidla, aby nedošlo k narušení principu původního algoritmu analýzy. Pro každý terminální symbol nacházející se v derivovaném řetězci pravidla je nutné odečíst jeho výskyt z tabulky. V případě, že výsledný počet výskytů některého z terminálů klesne pod nulu, nemůže syntaktická analýza skončit přijetím vstupního řetězce, neboť je po něm vyžadován neodpovídající (větší) počet výskytů konkrétního terminálu, než se v něm doopravdy nachází.

4 KONTEXTOVÁ PREDIKCE

Poněkud komplexnějším zpracováním údajů z lexikální analýzy lze získat tabulku s pozicemi konkrétních typů terminálů. To umožní kontrolu vzájemných pozic terminálů v derivovaném řetězci aplikovaného pravidla. Kontextová predikce tedy spočívá v ověřování existence požadované posloupnosti terminálů ve vstupním řetězci. Aby toho nebylo málo, uchováním informací z předcházejících kontrol lze omezit oblast prohledávání na rozmezí dané již nalezenými symboly. Akce, které jsou nutné provést po aplikaci pravidel, se dají shrnout následujícími body:

- Nechť X udává pozici posledního přijatého terminálního symbolu nebo 0, pokud dosud nebyl žádný přijat. Nechť Y je derivovaný neterminál, pak $X = (\ p \mid AYB$ je obsah derivačního zásobníku, $A, B \in \Sigma^*$, $p = Max(X, LastTerminal(A)))$ je nejnižší možná pozice hledaného terminálu.
- Pro každý terminální symbol v derivovaném řetězci aplikovaného pravidla nalezni v tabulce nejbližší číslo větší, než X. Neexistuje-li v tabulce taková pozice terminálu, zamítni vstupní řetězec. V opačném případě zapiš do zásobníku nalezenou pozici tohoto terminálu.
- Aktualizuj hodnotu X pozicí nalezeného terminálu.
- Prohledej zásobník dolů od pozice nalezeného terminálu dokud nenarazíš na symbol konce zásobníku anebo terminální symbol s větší pozicí, než X. Pro všechny terminály s pozicí menší nebo rovnou X nalezni novou pozici v tabulce a aktualizuj hodnotu v zásobníku a číslo X. Opakuj předchozí bod, dokud nebude přeskočen žádný terminál anebo vstup zamítnut.

Příklad 1: Nechť M je modifikovaný hluboký zásobníkový automat [4] takový, že M = ({*s, x, y, z, x', y', z', f, T*}, {*a, b, c*}, {*S, A, B, C, #*}, *R, s, S, f*) s R obsahujícím pravidla $sS \to xABC$, $xA \to yaA$, $yB \to zbB$, $zC \to xcC$, $xA \to y'\varepsilon$, $y'B \to z'\varepsilon$, $z'C \to f\varepsilon$ pro analýzu jazyka $L = a^n b^n c^n$ se vstupní tabulkou $T = (a \to \{1,2\}, b \to \{5,6\}, c \to \{3,4\})$.

$(s, aaccbb, S\#, S\#)$	$\Rightarrow$	$(x, aaccbb, ABC\#, ABC\#)$	$[sS \to xABC]$
	$\Rightarrow$	$(y, aaccbb, aABC\#, 1ABC\#)$	$[xA \to yaA]$
	$\Rightarrow$	$(y, accbb, ABC\#, ABC\#)$	
	$\Rightarrow$	$(z, accbb, AbBC\#, A5BC\#)$	$[yB \to zbB]$
	$\Rightarrow$	$(x, accbb, AbBcC\#, A5B-C\#)$	$[zC \to xcC]$
	$\Rightarrow$	$(y, accbb, aAbBcC\#, -)$	$[xA \to yaA]$
	$\Rightarrow$	$(y, ccbb, AbBcC\#, -)$	
	$\Rightarrow$	$(z, ccbb, AbbBcC\#, -)$	$[yB \to zbB]$
	$\Rightarrow$	$(x, ccbb, AbbBccC\#, -)$	$[zC \to xcC]$
	$\Rightarrow$	$(x, ccbb, bbBccC\#, -)$	$[xA \to y'\varepsilon]$
	$\Rightarrow$	$(x, cbb, bBccC\#, -)$	$b \neq c$

Výše zmíněný příklad ukazuje syntaktickou analýzu nevyhovujícího vstupního řetězce. Zvýrazněný řádek je místem, kde terminálová predikce zamítne vstup, což je o polovinu aplikovaných pravidel

dříve, než by došlo k zamítnutí původním algoritmem. Důvodem, proč predikce ukončila analýzu, byla neexistene terminálu c na pozici vyšší jak hodnota 5 daná předcházejícím terminálem b. V tabulce se sice vyskytují ještě 2 terminály c, nicméně jejich pozice jsou nižší, než požadovaná hodnota.

4.1 Ověření existence podmnožiny n-tého stupně neterminálového uzávěru

Výše zmíněnou metodu lze rozšířit o ověření samotných neterminálů. Z pravidel gramatiky lze zjistit, které terminální symboly případně posloupnosti symbolů jsou derivovatelné z jednotlivých neterminálů. Pro přijetí vstupního řetězce je nutné, aby minimálně jedna taková existovala. Tyto skupiny nazvěme neterminálovým uzávěrem. Ten lze upřesnit stanovením počtu povolených derivací z původního symbolu. Jde tedy o neterminálový uzávěr hloubky n. Jelikož je ale množina úplně všech možných derivací velmi obecná, bylo by vhodné ji pokud možno zpřesnit. To lze provést další analýzou pravidel gramatiky a to vytvořením tzv. stavového uzávěru, tedy množiny vstupních neterminálů, které se mohou objevit v daném stavu.

5 ZÁVĚR

Terminálová predikce je volitelnou modifikací syntaktické analýzy urychlující detekci nevyhovujícího vstupu. Vyžaduje předzpracování lexikální analýzy k vytvoření statických údajů. Podle typu těchto dat a způsobu ověřování lze predikci rozdělit na bezkontextovou, využívající pouze počty vstupních terminálů, a kontextovou, která porovnává relativní pozice derivovaných terminálů se vstupní posloupností symbolů. Terminálová predikce má předpoklad pro urychlení syntaktické analýzy, nicméně je potřeba provést další výzkum k prokázání jejího skutečného vlivu. Predikce je však dostatečně obecná, aby ji bylo možné aplikovat i na jiné typy gramatik, ale efektivita již nemusí být tak velká, jako u stavových gramatik a jejich možnému větvení. Co se týká neterminálových uzávěrů, další výzkum je nutně podmíněn pozitivním přínosem kontextové predikce jako takové.

REFERENCE

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

[2] Meduna, A.: Automata and Languages: Theory and Applications [Springer, 2000], London, GB, Springer,

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

[4] Meduna, A.: Deep Pushdown Automata, In: Acta Informatica, roč. 2006, č. 98, DE, s. 114-124, ISSN 0001-5903 2005, s. 892, ISBN 1-85233-074-0