SYNTAKTICKÁ ANALÝZA A TVORBA JEDNODUCHÝCH GENERÁTORŮ

Schenk Jiří
University of Ostrava, Faculty of Science, Departmen of Computers,
30. Dubna 22, 701 33 Ostrava, +420 597 092 186, jiri.schenk@osu.cz

Abstrakt

Tento článek se zabývá z části problematikou lexikální a syntaktické analýzy. Další část je zaměřena na postup konstrukce jednoduchých syntaktických a lexikálních generátorů za použití dané gramatiky a formy zápisu této gramatiky. A v poslední části je uvedeno zdůvodnění, proč použít generátory a ne jenom statické analyzátory.

Klíčová slova: *syntaktická analýza; lexikální analýza; generátor; BNF; gramatika.*

Úvod

Problematika syntaktické analýzy spadá do oboru teoretické informatiky a je jedna z nejdůležitějších stavebních prvků jakéhokoliv překladače. Syntaktickou analýzu v překladačích provádí syntaktický analyzátor (častěji se setkáme s názvem parser) a má za úkol určit strukturu daného programu pomocí syntaktického stromu. Aby parser mohl pracovat, musí v sobě obsahovat lexikální analyzátor, popřípadě k němu musí být připojen. Lexikální analyzátor (provádí lexikální analýzu) má za úkol rozdělit zdrojový program na nejmenší logickou jednotku v kódu programu (token). Tyto tokeny dále předává syntaktickému analyzátoru, který může následně požádat lexikální analyzátor o další token.

Lexikální a syntaktická analýza

Lexikální analýza (LA) je proces, při kterém se ze vstupního souboru rozpoznají jednotlivé lexémy a ty jsou poté převedeny na tokeny. Pojem lexém znamená posloupnost symbolů (znaků), které jsou obsaženy ve vstupním souboru. Každý lexém je vzorem pro konkrétní symbol. Za lexémem tedy považujeme řetězec znaků, u kterých budeme předpokládat, že se jedná o jediný symbol (token). Token je nejmenší logická jednotka a každému tokenu je přidělena identifikace. Například znaménku + bude přidělena identifikace - operátor. Lexém se tedy skládá ze dvou částí. První je identifikace (o jaký typ symbolu jde) a druhá je hodnota lexému (řetězec, číslo a podobně). [2]

Syntaktická analýza (SA) je proces, který se snaží zjistit, jestli vstupní soubor (zdrojový text) tvoří takovou větu, aby odpovídala gramatice překládaného jazyka. V podstatě můžeme říct, že syntaktická analýza kontroluje syntaxi daného jazyka. Syntaktická analýza zahrnuje druhou fázi překladače a je pokládána za nejdůležitější funkcí překladače. Existují dvě metody, jak tvořit syntaktickou analýzu, které se liší směrem vytváření derivačního stromu. [2, 3]

První metoda je jedna ze základních metod syntaktické analýzy a nazývá se „zhora dolů“. Vychází ze startovacího symbolu gramatiky a postupuje k terminálním symbolům, které tvoří analyzovanou větu. Derivační strom se vytváří podle levé derivace od kořene k listům. Této analýze se také někdy říká proces hledání levého rozkladu analyzované věty. Což je posloupnost čísel pravidel použitých při levé derivaci. Princip této metody spočívá z myšlenky postupného dopředného odvozování na zásobníku od počátečního neterminálu S a srovnávání analyzovaného slova po termínálních symbolech, které se objeví na zásobníku, až nastane situace, kdy nezbude

nic ke srovnání, jak v původním slově, tak i v přepisovaných řetězcích od S na zásobníku. Kroky, ve kterých přepisujeme od S, nazýváme expanze. Expanze nastává, protože neterminály rozšiřujeme na řetězce podle pravidel. [1, 2]

Druhá metoda pro syntaktickou analýzu se nazývá „zdola nahoru“. Jedná se o opačný přístup než u metody „shora dolů“. Při konstrukci derivačního stromu postupujeme od listů pomocí pravých derivaci až ke kořeni stromu. Tuto metodu můžeme nazvat jako proces hledání pravého rozkladu analyzované věty. Výsledkem je posloupnost čísel pravidel použitých při pravé derivaci v obráceném pořadí. V principu jde o postupné zpětné odvozování. Symboly vkládáme do zásobníku až se nám vyskytne řetězec, který se vyskytuje u některého z pravidel a provede se odvození na neterminál. Tomuto postupu se říká redukce. Jakmile je slovo celé přečteno a na zásobníku zbyl pouze neterminál S, slovo je přijato. [1, 2]

Konstrukce generátorů

Při tvorbě obou generátorů (lexikálního a syntaktického) je zapotřebí si ujasnit dvě věci. S jakou gramatikou budou generátory pracovat a v jaké formě bude gramatika zapsána.

Pro vytváření jednoduchých generátorů je vhodný zápis gramatiky v Backus-Naurově formě (BNF) díky tomu, že zápis je velmi podobný bezkontextové gramatice, ale bližší programátorům. Podobně jako bezkontextová gramatika obsahuje neterminály, které se zapisují do úhlových závorek přes symbol ::= na slova složená buď z terminálů nebo neterminálů, popřípadě z terminálů i neterminálů zároveň. Jeden neterminál může obsahovat více než jedno pravidlo. Pravidla se od sebe oddělují symbolem |. Jedná se tedy o zápis ve tvaru:

<název neterminálu> ::= slovo | slovo | slovo | … | $slovo_n$

Obrázek 1. Zápis v BNF.

Gramatika s jakou budou generátory pracovat, závisí na zvolení metody syntaktické analýzy. Gramatiky využívající metody „shora dolů“ se nazývají LL gramatiky a gramatiky využívající metody „zdola nahoru“ se nazývají LR gramatiky. Tyto gramatiky lze dále rozdělit na silné a slabé podle toho, zda potřebují informaci o průběhu analýzy. LL gramatiky můžeme dále dělit na: Jednoduché LL gramatiky, Q-gramatiky, LL(k) gramatiky. Každá z těchto gramatik má jisté omezení a různou náročnost na tvorbu. Nejsložitější jsou LL(k) gramatiky, které obsahují funkce FIRST a FOLLOW, ovšem mají nejvyšší mírů vyjádření. [3, 4]

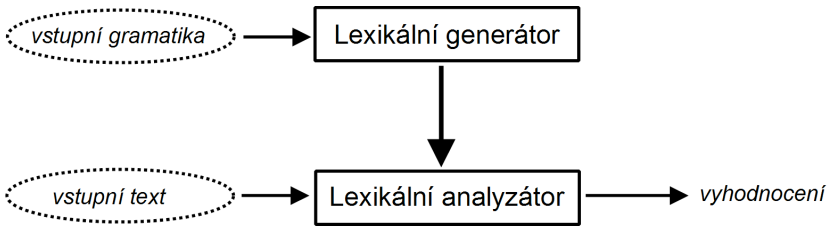
```
<dotaz> ::= select <atribut> <umisteni> | select <atribut> <umisteni> <razeni>
<umisteni> ::= from <tabulka>
<atribut> ::= jmeno | prijmeni | rodne_cislo | id_cislo
<tabulka> ::= zamestnanec | zamestnavatel | zakaznik
<razeni> ::= order by <atribut>
```

Obrázek 2. Příklad zápisu gramatiky pro SQL dotaz na konkrétní databázi.

Lexikální generátor

Lexikální generátor (LG) má za úkol vygenerovat konkrétní lexikální analyzátor pro zadanou vstupní gramatiku od uživatele v BNF nebo EBNF. Vygenerovaný lexikální analyzátor

kontroluje vstupní text a provádí na něm lexikální analýzu. Zjednodušené schéma pro generování vypadá takto:



Obrázek 3. Schéma generování LA.

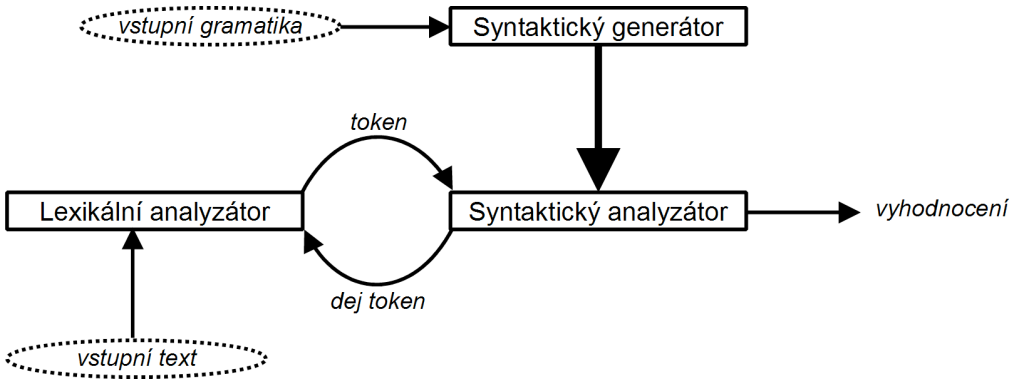
LG lze rozdělit do dvou fází. V první fázi si lexikální generátor načte celou vstupní gramatiku a začne ji prohledávat. Jakmile nalezne symbol pro terminál, začne postupně načítat následující symboly (pokud se jedná o terminál složen z více než jednoho symbolu) a vytvoří z něj lexém. Poté se podívá do tabulky tokenů, jestli nějaký token neobsahuje stejný lexém. Pokud ne, vytvoří z tohoto lexému token a ten uloží do tabulky tokenů. Po přečtení celé gramatiky je seznam naplněn všemi tokeny, které se můžou vyskytovat ve vstupním textu.

V druhé fázi lexikální generátor vytvoří konkrétní lexikální analyzátor za pomocí tabulky tokenů a algoritmů potřebných k lexikální analýze.

Vygenerovaný lexikální analyzátor čte vstupní text znak po znaku a hledá pro konkrétní znak nebo řetězec znaků vhodný automat, popřípadě sjednocené automaty. Pokud ani jeden z použitých automatů nebude chybový, proběhla lexikální analýza v pořádku.

Syntaktický generátor

Syntaktický generátor (SG) má za úkol vygenerovat konkrétní syntaktický analyzátor pro zadanou vstupní gramatiku od uživatele v BNF nebo EBNF. Vygenerovaný syntaktický generátor kontroluje vstupní text (respektive kontroluje tokeny, které mu předává lexikální analyzátor) a provádí na něm syntaktickou analýzu. Zjednodušené schéma pro generování vypadá takto:



Obrázek 4. Schéma generování SA.

SG lze opět rozdělit na dvě fáze. V první fázi si syntaktický generátor načte celou vstupní gramatiku a začne ji prohledávat. Hledá všechny neterminální symboly a ty ukládá do seznamu pro neterminály za předpokladu, že se již daný neterminál v seznamu nenachází. První nalezený neterminál (neterminál na prvním místě v seznamu) je generátorem automaticky pokládán za tzv.

startovní neterminál. To znamená, že syntaktický (derivační) strom bude za kořen pokládat, právě tento neterminál.

V druhé fázi syntaktický generátor vytvoří konkrétní syntaktický analyzátor za pomocí seznamu neterminálů, funkcí FIRST a FOLLOW (pokud jsou zapotřebí) a algoritmů potřebných k syntaktické analýze.

Vygenerovaný syntaktický analyzátor požaduje tokeny od lexikálního analyzátoru, který mu je následně předává, a podle těchto tokenů vytváří syntaktický strom.

Použití

Mnohé syntaktické analyzátory, popřípadě celé překladače jsou tvořeny staticky. Tím je myšleno, že analyzátory jsou tvořeny pro konkrétní zadání (problém). Pokud takové analyzátory je třeba změnit, musí být proveden zásah přímo do zdrojových kódů těchto analyzátorů. A to platí i pro sebemenší změnu v gramatice jazyka. Při použití generátorů se tento problém vyřeší, protože nám stačí změnit pouze soubor se vstupní gramatikou (tím se vygeneruje „nový“ analyzátor“) a není třeba zasáhnout do kódu.

Závěr

Syntaktická analýza a její využití provází programátory téměř na každém kroku jejich práce. Je škoda, že mnozí z nich nevyužívají známé a odzkoušené metody pro jejich správnou tvorbu s využitím automatů. Proto byl vytvořen tento článek, aby nastínil problematiku syntaktické analýzy s možností vytváření generátorů syntaktické analýzy, které mohou vytvářet různé analyzátory, dle zadané vstupní gramatiky. Jinými slovy lze říct, že stačí vytvořit jeden generátor, který dokáže vyprodukovat nekonečně mnoho analyzátorů, které můžou být okamžitě použity v daných programech.

Literatura

[1] CHYTIL, Michal. *Automaty a gramatiky*. Spálená 51, 11302 Praha 1 : Nakladatelství technické literatury, 1984. 336 s.
[2] HABIBALLA, Hashim. Regulární a bezkontextové jazyky I [online]. 2005 [cit. 2014-03-15]. ISBN 80-7042-852-X. Dostupné z: http://www1.osu.cz/home/Habibal/files/xrab1.pdf
[3] HABIBALLA, Hashim. Regulární a bezkontextové jazyky II [online]. 2005 [cit. 2014-03-15]. Dostupné z: http://www1.osu.cz/home/Habibal/files/rabj2.pdf
[4] AHO, Alfred V. Compilers: principles, techniques and tools. Vyd. 2. New Jersey: Prentice-Hall, 1988, 796 s. ISBN 02-011-0088-6.

Abstract

One of the main objectives of this article is pertaining to the lexical and syntactic analysis. Another part of this work focuses on simple lexical and syntactic generator design with a help of using given grammar and also mentions forms how to record this grammar. Last part explains reasons as to why incorporate generators and not just use static analyzers on their own.