Syntaktická analýza založená na multigenerativních systémech

Závěrečná práce z předmětu TJD

Jakub Martiško

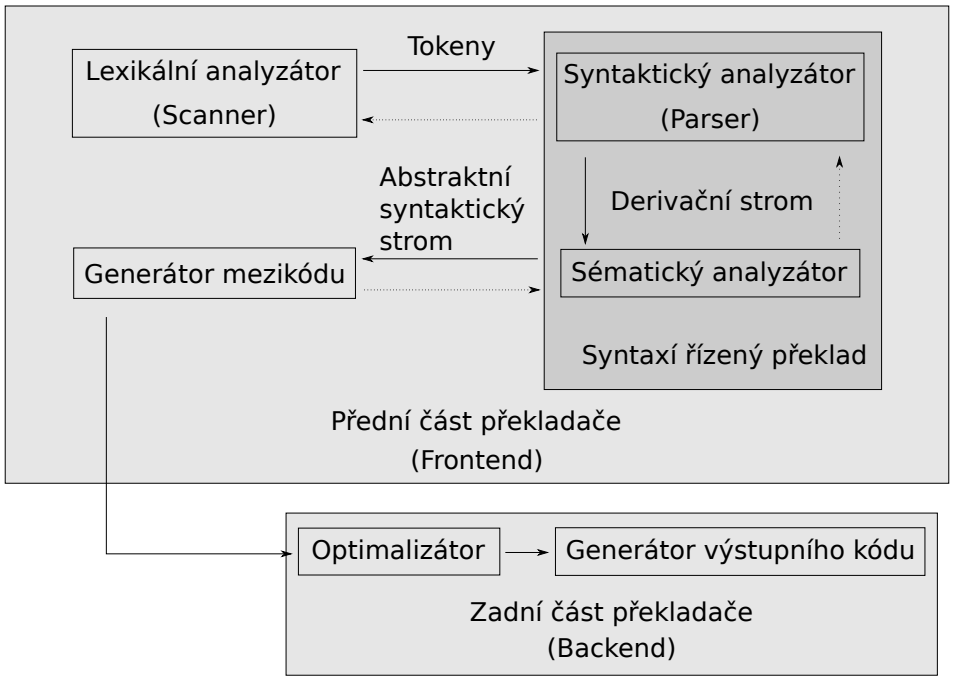
29. ledna 2016

Obsah

Kapitola 1

Úvod

Syntaktická analýza (parsing) je zásadní částí překladu vstupních zdrojových kódů v jednom jazyce na ekvivalentní reprezentaci v jazyce jiném (typicky nějaká forma binárního souboru pro daný procesor, případně instrukce daného virtuálního stroje). Syntaktická analýza většinou řídí celou činnost přední části překladače a jejím cílem je zjistit, je-li vstupní soubor sestaven na základě symbolů a pravidel požadovaného jazyka. Obrázek 1.1 pak zachycuje strukturu typického překladače.



Obrázek 1.1: Struktura typického překladače

Běžně používané metody syntaktické analýzy jsou založeny na bezkontextových gramatikách a jim ekvivalentních zásobníkových automatech. Tyto modely mají tu výhodu, že jsou poměrně jednoduché a mají dostatečnou vyjadřovací sílu na popis většiny konstrukcí, běžně využívaných v programovacích jazycích. Na druhou stranu ale existují jisté jazyky, které tyto gramatiky nejsou schopny popsat, příkladem takového jazyka je jazyk, který sestává ze tří symbolů a v každém řetězci tohoto jazyka je počet výskytů všech tří symbolů stejný.

Další nevýhodou standardních syntaktických analyzátorů je, že bezkontextové gramatiky přímo nevyužívají nějaké formy paralelismu a jedná se o sekvenční model. Zavedení paralelismu v rámci konstrukce překladačů je ovšem poměrně praktické - lze pak například provádět překlad do více jazyků zároveň, případně alespoň zrychlit samotný překlad.

Existuje více různých modelů, které se snaží tyto „nedostatky“ nějakým způsobem odstranit a zachovat přitom jednoduchost bezkontextových gramatik. Jedním z těchto modelů jsou pak například gramatické systémy, o kterých pojednává i tato práce. Konkrétní variantou, kterou se tato práce zabývá, jsou pak tzv. multigenerativní systémy. Tyto systémy jsou pak schopny řešit jak první z výše popsaných nevýhod, tedy vyjadřovací sílu modelu, tak i nevýhodu druhou – jsou schopny generovat více řetězců paralelním způsobem.

Práce je pak členěna na dvě části. V rámci kapitoly 2 jsou popsány základní používané metody syntaktické analýzy založené na bezkontextových gramatikách. Dále jsou, v rámci této kapitoly, také popsány gramatické systémy, především pak jejich multigenerativní varianta. Druhá část práce, která sestává z kapitoly 3, se pak již zabývá samotnou analýzou založenou na těchto systémech.

Kapitola 2

Definice a pojmy

V rámci této kapitoly budou zavedeny základní pojmy z oblasti syntaktické analýzy a některých formálních modelů. Předpokládá se, že je čtenář seznámen se základními pojmy z oblasti teorie formálních jazyků, především pak se zásobníkovými automaty a bezkontextovými gramatikami. Tato problematika je popsána např. v [4]. Tato kapitola pak vychází z [1] (část o syntaktické analýze), [5] (část o gramatických systémech) a [3] (část o multigenerativních systémech).

2.1 Syntaktická analýza

Cílem syntaktické analýzy je rozhodnout, patří-li zadaný řetězec do vstupního jazyka, na jehož základě je daný analyzátor sestaven. Výstupem takového analyzátoru je pak přijetí nebo zamítnutí vstupního řetězce a v případě přijetí pak také posloupnost pravidel gramatiky, která daný řetězec generuje[1]. Většina standardních metod syntaktické analýzy je založena na bezkontextových gramatikách a jim odpovídajících zásobníkových automatech. Z pohledu průběhu analýzy rozlišujeme dva základní přístupy:

1. Analýzu shora dolů.
2. Analýzu zdola nahoru.

V případě analýzy shora dolů, začíná analyzátor od počátečního symbolu dané gramatiky. Tento symbol postupně rozgenerovává a ukládá na zásobník. Následně porovnává takto vzniklé terminály se symboly vstupního řetězce, a pokud se tyto symboly shodují, pak je ze zásobníku odstraňuje. Analýza je pak považována za úspěšnou, pokud je přečten celý vstupní řetězec a ze zásobníku jsou odstraněny všechny vygenerované symboly.

[1] V praxi pak analyzátor spíše provede nějaké akce spojené s jednotlivými pravidly, než aby vracel pouze jejich posloupnost.

Analýza zdola nahoru pak pracuje opačným způsobem. Ze vstupního řetězce postupně načítá jednotlivé symboly a ukládá je na zásobník. Jakmile se na zásobníku vyskytuje posloupnost symbolů, která odpovídá pravé straně některého z pravidel příslušné gramatiky, je tato posloupnost nahrazena stranou levou. Cílem pak je opět přečíst celý řetězec a na zásobníku získat pouze jediný symbol, a sice symbol počáteční.

2.1.1 LL(1) syntaktická analýza

Příkladem analýzy shora dolů je LL(k) analýza, kde první L značí čtení řetězce zleva doprava, druhé L značí přepisování nejlevějšího neterminálu a k značí počet vstupních symbolů, které je potřeba znát v jednotlivých krocích výpočtu. V rámci této práce bude popsána pouze LL(1) syntaktická analýza (LL(k) pro $k > 1$ se zabývá např. [2]) a nepovede-li to k možným nejasnostem, bude dále pod zkratkou LL myšlena právě LL(1). Pro sestavení analyzátoru, pracujícího touto metodou je potřeba sestrojit tzv. LL tabulku. Tato tabulka popisuje, jak se má systém chovat na základě vstupního symbolu a neterminálu nacházejícího se na vrcholu zásobníku. Řádky tabulky jsou indexovány pomocí neterminálů dané gramatiky, sloupce pak pomocí terminálů a symbolu $ značícího konec vstupu. Průsečíky jednotlivých řádků a sloupců pak obsahují pravidlo, které má být použito pro rozgenerování daného neterminálu, případně jsou prázdné, což značí syntaktickou chybu v rámci vstupního řetězce. Pokud žádná z buněk tabulky neobsahuje více než jedno pravidlo, pak je příslušná gramatika LL(1).

Pro sestavení LL tabulky pro gramatiku $G = (N, T, P, S)$ je potřeba zavést několik pomocných relací. První z těchto relací je relace $First(\alpha)$. Tato relace přiřazuje každému řetězci $\alpha \in (N \cup T)^*$ množinu symbolů[2], kterými může začínat řetězec, vygenerovaný pomocí posloupnosti pravidel z řetězce α. Formálně tedy $First(\alpha) = \{a | a \in T, \alpha \Rightarrow^* a\alpha'; \alpha, \alpha' \in (N \cup T)^*\}$.

Další relací je relace $Empty(\alpha)$. Tato relace určuje, je-li možno řetězec α přepsat na řetězec ε a tím jej odstranit. Pro řetězec $\alpha \in (N \cup T)^*$ platí, že $Empty(\alpha) = \{\varepsilon\}$ tehdy a jen tehdy pokud platí, že $\alpha \Rightarrow^* \varepsilon$, v opačném případě $Empty(\alpha) = \emptyset$.[3]

Relace $Follow(A)$, kde $A \in N$, obsahuje terminály, které se mohou objevit bezprostředně vpravo od neterminálu A. Formálně tedy $Follow(A) = \{a | a \in T, S \Rightarrow^* \alpha A a \beta; \alpha, \beta \in (N \cup T)^*, A \in N\}$.

Poslední relací je relace $Predict(p)$, kde $p : A \to x \in P$. Tato relace obsahuje terminály, kterými může začínat řetězec, vzniklý použitím pravidla p na nejlevější neterminál (používáme LL analýzu). Tato relace je pak sestavena pomocí následujících pravidel[4].

- Je-li $Empty(x) = \{\varepsilon\}$ pak $Predict(A \to x) = First(x) \cup Follow(A)$.

[2] Většinou nás zajímají právě tyto množiny a často se pak pojem relace a množina zaměňuje

[3] Relace *Empty* se často slučuje s relací *First* (např. v [1]). Relace *First* pak obsahuje symbol ε pokud by jej obsahovala relace *Empty*. Relace *Empty* se pak v takovém případě nepoužívá.

[4] Jelikož je tato relace založena na relacích předchozích, tak obdobně jako relace *Empty*, se často vynechává a LL tabulka se pak sestavuje pouze z relací *First* a *Follow*

- Je-li $Empty(x) = \emptyset$ pak $Predict(A \to x) = First(x)$.

Na základě relace *Predict* se pak sestavuje výsledná LL tabulka. Buňka tabulky, s indexy $A \in N$ a $a \in T \cup \{\$\}$ obsahuje pravidlo $p : A \to x$ tehdy a jen tehdy, pokud $a \in Predict(p)$.

Existují dvě základní varianty implementace LL analyzátoru na základě LL tabulky:

- Prediktivní syntaktická analýza.
- Rekursivní sestup.

Prediktivní analýza využívá přímo LL tabulky a zásobníkového automatu. Nejprve je na zásobník vložen počáteční symbol. Ten je následně rozgenerováván pomocí pravidel daných LL tabulkou. Nalézá-li se na vrcholu zásobníku terminál, je tento terminál porovnán se vstupním symbolem a jsou-li stejné je odstraněn. V opačném případě dochází k syntaktické chybě. Nalézá-li se na vrcholu neterminál, je tento symbol rozgenerován pomocí pravidla určeného pomocí buňky LL tabulky s indexy danými tímto neterminálem a vstupním symbolem.

Rekurzivní sestup pak neimplementuje přímo zásobníkový automat. Místo toho přiřazuje každému neterminálu funkci, která simuluje aplikaci pravidla s tímto symbolem na levé straně. V rámci těchto funkcí se na základě terminálů na vstupu určí, které z možných pravidel je aplikováno. V případě, že zvolené pravidlo obsahuje další neterminály, je zavolána funkce, která provádí simulaci tohoto nového neterminálu. Jakmile funkce úspěšně porovná očekávanou posloupnost symbolů a vstupních symbolů, vrátí funkci, která ji zavolala, informaci o úspěchu. Jakmile takto vrátí úspěch i funkce, která odpovídá počátečnímu symbolu, je považována analýza za úspěšnou.

2.1.2 LR syntaktická analýza

Jak již bylo řečeno, druhým možným přístupem k syntaktické analýze je analýza zdola nahoru. Na tomto principu je postavena LR analýza, kde L značí opět čtení zleva doprava a R pak značí tvorbu pravého rozboru. Existuje více metod tvorby LR tabulky, které se liší složitostí výsledného automatu a jeho vyjadřovací silou. Základní variantou je tzv. SLR varianta, která je popsána např. v [1]. Algoritmus, který provádí samotnou analýzu, se pak neliší, rozdílná je pouze tvorba tabulky. Oproti LL analýze, která pracuje jen se zásobníkem, využívá LR analýza také vnitřních stavů automatu. Pomocí těchto stavů je pak analyzátor schopen určit, kde v rámci pravé strany pravidla se momentálně nachází. Výsledná LR tabulka pak sestává ze dvou částí – akční části a GOTO části. Řádky obou těchto částí jsou indexovány pomocí vnitřních stavů automatu. Sloupce jsou pak indexovány symboly dané gramatiky, v případě akční části se jedná o terminály a symbolu pro konec vstupu, v případě GOTO části pak o neterminály. Jednotlivé buňky GOTO části obsahují stav nebo jsou prázdné. Akční část tabulky pak obsahuje jednu z následujících možných položek:

- Položku tvaru sq, kde s značí shift (vložení symbolu na zásobník) a q stav automatu.
- Položku tvaru rp, kde r značí redukci a p pravilo, podle kterého se provede.
- Položku značící úspěch syntaktické analýzy.
- Prázdnou buňku, značící syntaktickou chybu.

Oproti LL analýze, kde se na zásobníku pracuje pouze se samotnými symboly, pracuje LR analyzátor s dvojicemi tvaru $(symbol, stav)$. Nejprve je na zásobník vložena dvojice $(\$, q_0)$, kde \$ značí dno zásobníku a q_0 je počáteční stav. Aktuální stav je pak také nataven na stav počáteční. Následně probíhá analýza na základě tabulky. Obsahuje-li tabulka pro aktuální kombinaci stavu a vstupního terminálu x položku tvaru sq, je aktuální stav nastaven na stav q a na zásobník je uložena dvojice (x, q). Obsahuje-li naopak buňka položku značící redukci dle zadaného pravidla, jsou z vrcholu zásobníku odstraněny symboly, které odpovídají pravé straně toho pravidla (pokud se symboly na vrcholu zásobníku neshodují s pravou stranou, jedná se o syntaktickou chybu). Na základě levé strany pravidla a stavu uloženého na vrcholu zásobníku (po odstranění symbolů, které odpovídají pravé straně použitého pravidla) je pomocí GOTO části tabulky určen nový aktuální stav. Tento nový stav je pak uložen společně s neterminálem z levé strany pravidla na vrchol zásobníku. Analýza pak pokračuje tak dlouho, dokud není dosaženo takové kombinace vstupního symbolu a stavu, který odpovídá buňce značící úspěch.

2.2 Gramatické systémy

Gramatické systémy jsou jedním z možných řešení problému, kdy se snažíme zvýšit generativní sílu bezkontextových gramatik, ale na druhou stranu zachovat jejich jednoduchost. Gramatické systémy se dělí na dva základní druhy: CD (cooperating dristributed) a PC (parallel computing) gramatické systémy. Oba tyto druhy se skládají z několika gramatik (většinou bezkontextových), nazývaných komponenty, které spolu určitým způsobem spolupracují a generují tak výsledný řetězec resp. jazyk.

V případě CD gramatických systémů, pracují všechny komponenty nad společnou větnou formou. Jednotlivé komponenty pracují sekvenčně, nejprve je jedna komponenta vybrána, vybraná komponenta pak provádí přepisování větné formy na základě svých pravidel. V určitém momentu je právě aktivní komponenta deaktivována a aktivována je komponenta nová. Způsob této deaktivace je dán tzv. módem derivace, přičemž rozlišujeme tři základní varianty:

1. Komponenta musí před svou deaktivací přepsat všechny symboly větné formy, které je schopna přepsat.
2. Komponenta se může deaktivovat kdykoliv.

3. Počet přepsání větné formy při aktivaci komponenty je nějakým způsobem ohraničen konstantou k.

V případě poslední z těchto variant je pak nutné, aby tato podmínka byla splněna i během poslední aktivace, která vede na vygenerování věty. V opačném případě dojde k zaseknutí systému.

Jak již bylo řečeno, druhým základním druhem gramatických systémů jsou PC gramatické systémy. Ty, na rozdíl od CD gramatických systémů, nepracují sekvenčně nad společnou větnou formou. Místo toho, pracují všechny komponenty paralelně, každá nad vlastní větnou formou. Tento typ systému pak kromě abeced terminálů a neterminálů obsahuje i abecedu třetí, která obsahuje tzv. komunikační symboly, přičemž existuje bijekce, mezi touto abecedou a množinou jednotlivých komponent. Derivační krok je dvojího druhu, v případě prvního typu aplikuje každá komponenta (s výjimkou komponent, jejichž větná forma je zároveň větou) nějaké ze svých pravidel na svou větnou formu. Druhý způsob derivačního kroku je použit v případě, že se v některé z větných forem vyskytuje komunikační symbol. Tato druhá varianta pak nahradí výskyty všech komunikačních symbolů aktuální větnou formou komponenty, která tomuto symbolu přísluší. Tato komponenta pak pokračuje svůj výpočet od počátečního symbolu. V případě, že dvě (případně více) komponent takto na sebe odkazují vzájemně, dojde k zaseknutí systému. Tento druhý typ derivačního kroku má pak vyšší prioritu než typ první. Výsledný řetězec resp. jazyk, je pak dán řetězcem první komponenty systému.

2.3 Multigenerativní gramatické systémy

Tento druh gramatických systémů vychází z PC gramatických systémů. Oproti nim, však tato varianta neobsahuje abecedu komunikačních symbolů. Místo toho je zavedena jistá forma synchronizace použitých pravidel. Na základě této synchronizace pak rozlišujeme dva druhy těchto gramatických systémů:

1. Pravidlově synchronizované.
2. Neterminálově synchronizované.

V závislosti na volbě přepisovaného neterminálu v jednotlivých komponentách pak rozlišujeme:

1. Obecné multigenerativní systémy.
2. Kanonické multigenerativní systémy.
3. Hybridní multigenerativní systémy.

2.3.1 Obecné pravidlově synchronizované multigenerativní gramatické systémy

Obecný n-generativní pravidlově synchronizovaný multigramatický systém (n-OGP) Γ je $n+1$-tice

$$\Gamma = (G_1, \ldots, G_n, Q)$$

- $G_i = (N_i, T_i, P_i, S_i), 1 \leq i \leq n$ je bezkontextová gramatika.
- Q je množina kontrolních n-tic tvaru $(p_1, \ldots, p_n), p_i \in P_i$ pro všechna $1 \leq i \leq n$.

Obdobně jako se v případě standardních gramatik zavádí větná forma, zavádí se v případě multigenerativních gramatických systémů tzv. multiforma. Nechť $\Gamma = (G_1, \ldots, G_n, Q)$ je n-OGP, multiforma je pak n-tice $\xi = (x_1, \ldots, x_n)$, kde $x_i \in (N_i \cup T_i)^*$. Multiforma je tedy n-tice obsahující aktuální větnou formu každé komponenty.

Mějme OGP $\Gamma = (G_1, \ldots, G_n, Q)$ a multiformy $\chi_1 = (u_1 A_1 v_1, u_2 A_2 v_2, \ldots, u_n A_n v_n)$ a $\chi_2 = (u_1 x_1 v_1, u_2 x_2 v_2, \ldots, u_n x_n v_n)$, kde pro všechna $1 \leq i \leq n$ platí $A_i \in N_i$, $u_i, v_i, x_i \in (N_i \cup T_i)^*$ a $p_i : A_i \to x_i \in P_i$ a $(p_1, p_2, \ldots, p_n) \in Q$. Pak říkáme, že χ_1 přímo derivuje χ_2. Tuto skutečnost pak zapisujeme jako $\chi_1 \Rightarrow \chi_2$. Standardním způsobem pak $\Rightarrow^*$ a $\Rightarrow^+$ značí tranzitivní a reflexivní, resp. tranzitivní uzávěr této relace.

Přímý derivační krok pak sestává ze dvou částí, nejprve je z množiny Q vybrána n-tice q. Tato n-tice obsahuje n prvků, kde každý tento prvek odpovídá pravidlu jedné z komponent. Následně aplikuje každá komponenta takto zvolené pravidlo na svou část multiformy podobným způsobem, jako standardní bezkontextová gramatika. V případě obecných GS není kladen žádný požadavek na volbu konkrétního neterminálu, který má být přepsán.

Obdobně jako multiformu, definuje autor i tzv. n-jazyk. Jedná se o množinu takových multiforem, ve kterých jsou všechny řetězce tvořeny pouze terminály jednotlivých komponent, případně se jedná o prázdné řetězce. Jazyk generovaný těmito systémy je pak závislý na tzv. *módu*. Tento mód pak provádí nějaké základní operace na jednotlivými n-ticemi tvořícími n-jazyk. Těmito operacemi pak jsou:

1. Sjednocení – do výsledného jazyka jsou přidány všechny řetězce vygenerované jednotlivými komponentami.
2. Konkatenace – jednotlivé řetězce terminálů v rámci multiforem n-jazyka jsou konkatenovány. Výsledný jazyk pak sestává z těchto konkatenací.
3. Výběr 1. komponenty – výsledný jazyk obsahuje pouze řetězce, které vygeneruje první komponenta systému.

Příklad 1. *Předpokládejme pravidlově synchronizovaný gramatický systém se dvěma komponentami $G_1 = (N_1, T_1, P_1, S)$ a $G_2 = (N_2, T_2, P_2, Z)$ a s množinou řídicích dvojic Q. Přičemž platí, že:*

$N_1 = \{S, A, B\}$

$T_1 = \{a, b, c\}$

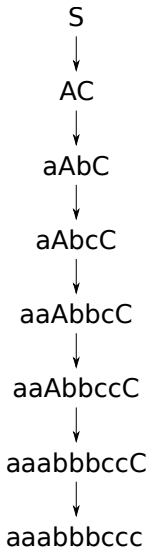
$P_1 = \{1 : S \to AC, 2 : A \to aAb, 3 : C \to cC, 4 : A \to ab, 5 : C \to c\}$

$N_2 = \{Z, D, E, F\}$

$T_2 = \emptyset$

$P_2 = \{1 : Z \to D, 2 : D \to E, 3 : E \to D, 4 : D \to F, 5 : F \to \varepsilon, \}$

$Q = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5)\}$



Obrázek 2.1: Ukázka činnosti 2-OGP

Ukázka činnosti tohoto systému je znázorněna na obrázku 2.1. Nezávisle na použitém módu, generuje tento systém jazyk $L = \{a^n b^n c^n : n > 0\}$*, který není bezkontextový. Druhá komponenta zde slouží jako jistý mechanismus, který určuje, bude-li systém dále pokračovat v generování jednotlivých terminálů nebo bude-li již s generováním končit. Neterminál* D *zde zastupuje jakýsi výchozí stav daného systému. Tento neterminál je možno dále přepsat na terminál* E *nebo* F*, v závislosti na pravidle zvoleném první komponentou (pravidlo 2, které znamená, že generování bude dále pokračovat a pravidlo 4, které značí konec generování a odstranění symbolu* A *z větné formy první komponenty). Symbol* E *je pak možno přepsat pouze zpět na symbol* D*, čehož je docíleno použitím pravidla 3 první komponentou. Naopak symbol* F *je možno přepsat pouze na prázdný řetězec, čímž dojde k odstranění jediného symbolu druhé větné formy. S tímto odstraněním je pak také spojeno*

odstranění symbolu C z první větné formy. První větná forma pak sestává pouze z terminálních symbolů a druhá větná forma odpovídá prázdnému řetězci. Tím je pak ukončeno generování výstupního řetězce.

2.3.2 Obecné neterminálově synchronizované multigramatické systémy

Jak již bylo řečeno dříve, druhou variantou multigenerativních systému jsou tzv. neterminálově synchronizované GS. Jejich obecná varianta (značeno jako n-OGN) pak opět sestává z n gramatik a množiny, která reguluje výpočet systému. Jak již název napovídá, toto řízení již není založeno na pravidlech nýbrž na neterminálech.

Formálně je tedy obecný n-generativní neterminálově synchronizovaný multigramatický systém (n-OGN) Γ definován jako $n + 1$-tice

$$\Gamma = (G_1, \ldots, G_n, Q)$$

- $G_i = (N_i, T_i, P_i, S_i), 1 \leq i \leq n$ je bezkontextová gramatika.
- Q je množina kontrolních n-tic tvaru $(A_1, \ldots, A_n), A_i \in N_i$ pro všechna $1 \leq i \leq n$.

Stejně jako u OGP, je i u této varianty gramatických systému zavedena multiforma. definice multiformy je pak stejná jako u OGP.

Mějme OGN $\Gamma = (G_1, \ldots, G_n, Q)$ a multiformy $\chi_1 = (u_1A_1v_1, u_2A_2v_2, \ldots, u_nA_nv_n)$ a $\chi_2 = (u_1x_1v_1, u_2x_2v_2, \ldots, u_nx_nv_n)$, kde pro všechna $1 \leq i \leq n$ platí $A_i \in N_i$, $u_i, v_i, x_i \in (N_i \cup T_i)^*$ a $p_i : A_i \to x_i \in P_i$ a $(A_1, A_2, \ldots, A_n) \in Q$. Pak říkáme, že χ_1 přímo derivuje χ_2. Tuto skutečnost pak zapisujeme jako $\chi_1 \Rightarrow \chi_2$. Standardním způsobem pak $\Rightarrow^*$ a $\Rightarrow^+$ značí tranzitivní a reflexivní, resp. tranzitivní uzávěr této relace.

Přímý derivační krok sestává opět ze dvou částí. V první části je opět vybrán jeden prvek množiny Q. Tato vybraná n-tice pak obsahuje právě jeden neterminál příslušející jednotlivým komponentám. Ve druhém kroku jsou pak v komponentách tyto neterminály vybrány a přepsány pomocí některého z pravidel dané komponenty. Výběr konkrétního neterminálu v rámci větných forem jednotlivých komponent pak není nijak omezen.

Obdobně jako v případě OGP je definován multijazyk OGN. Stejně tak jsou zavedeny i všechny tři možné módy generovaní výsledného jazyka.

2.3.3 Kanonické a hybridní varianty multigenerativních systémů

Jak již bylo řečeno v úvodu této kapitoly, kromě rozdělení multigramatických systémů na základě zavedení řízení výpočtu, lze je také dělit na základě volby neterminálů přepisovaných v jednotlivých krocích. V předchozích sekcích této kapitoly, jsme se setkali s tzv. obecnou verzí systémů. Tato verze nekladla žádné požadavky na volbu neterminálu, který bude přepsán. V rámci této sekce budou popsány kanonické varianty výše popsaných systémů.

Jelikož je omezení zavedené kanonickou verzi systému obdobné pro nonterminálově i pravidlově synchronizované systémy, bude v rámci této sekce popsána pouze první z těchto dvou možností. Kanonická varianta pravidlově synchronizovaných multigenerativních systému (n-KGP) pak zavádí omezení stejná, jako kanonická neterminálově synchronizovaná varianta (n-KGN).

Rozdíl, mezi obecnou a kanonickou verzí neterminálově synchronizovaných systému spočívá pouze v definici derivačního kroku. V případě obecné verze byl z větných forem jednotlivých komponent vybrán náhodný neterminál. Z těchto neterminálů pak byla sestavena n-tice, která se musela nacházet v množině Q. Kanonická varianta pak nevybírá neterminál náhodný, ale z větně formy každé komponenty vybere neterminál nejlevější. Pokud se v množině Q nalézá n-tice vzniklá z těchto *prvních* neterminálů, mohou být tyto symboly přepsány. V opačném případě dojde k zaseknutí systému a ten nevygeneruje žádný výstupní řetězec. V případě pravidlové synchronizace jsou pak nejlevější neterminály porovnávány s levou stranou pravidel řídících n-tic. Zcela analogicky lze pak také uvažovat kanonickou variantou obou těchto systémů, která vždy vybírá nejpravější neterminál ve větných formách.

Kombinací obecných a kanonických systému jsou pak hybridní systémy (n-HGN resp. n-HGP). V těchto systémech pak pracují některé komponenty obecným způsobem a některé kanonickým. V rámci definice systému pak musí být uvedeno, které komponenty jsou kanonické a které obecné.

Kapitola 3

Syntaktická analýza založená na multigenerativních systémech

V rámci [3] je zavedeno několik variant syntaktické analýzy založené na multigenerativních systémech. Tyto metody mají oproti metodám založeným na bezkontextových gramatikách dvě hlavní výhody. První výhodou je, že jsou takto pracující analyzátory schopny zpracovávat i některé jazyky, které nejsou bezkontextové. Druhou výhodou pak je, že tyto systémy jsou schopny na základě jedné vstupní gramatiky, provádět překlad do více jazyků zároveň. Obecný princip, na kterém jsou tyto metody založeny, spočívá v tom, že jedna z komponent je brána jako vstupní a ostatní jsou brány jako výstupní. Vstupní komponenta pak provádí syntaktickou analýzu vstupního řetězce a řídí komponenty výstupní, které generují jednotlivé výstupní řetězce. V rámci této kapitoly pak jsou popsány pravě metody zavedené v [3].

3.1 Deterministická paralelní syntaktická analýza shora dolů pro n-KGP

Vstupní gramatika je v tomto případě analyzována metodou shora dolů. Jelikož jsou použity pravidlově synchronizované systémy, ovlivňuje použité pravidlo i pravidla použitá ve výstupních komponentách.

3.1.1 Slabý deterministický LL n-KGP

Základní varianta tohoto typu syntaktické analýzy je postavena na tzv. slabém deterministickém LL n-KGP. Tento druh GS zavádí na jednotlivé komponenty určitá omezení, která zajistí, že bude s jejich pomocí možno provést deterministickou syntaktickou analýzu, tedy že v každém kroku analýzy, bude existovat jen jediný možný další postup. Tato omezení jsou dvě, první omezení spočívá v požadavku na vstupní gramatiku. Vstupní

gramatika musí být LL-gramatikou. Druhým požadavkem pak je, že nesmí existovat více řídicích n-tic, které by měly stejné pravidlo pro vstupní gramatiku. Tento požadavek pak zavádí jistou formu determinismu v rámci výstupních komponent.

Tato varianta pak pracuje podobným způsobem jako standardní LL analýza. Na zásobník každé komponenty je nejprve uložen symbol \$, značící dno zásobníku. Následně je na vrchol každého zásobníku vložen počáteční symbol příslušné komponenty. Následně je prováděna samotná syntaktická analýza, která pokračuje tak dlouho, dokud není celý vstupní řetězec přijat nebo dokud nedojde k chybě. V závislosti na druhu symbolu A na vrcholu zásobníku vstupní komponenty je prováděna jedna z následujících činností:

$A \in N$ Je-li na vrcholu zásobníku vstupní komponenty neterminál, pak je dle kombinace tohoto symbolu a vstupního symbolu vybráno pravidlo k expanzi na základě LL tabulky. Následně je vybrána řídicí n-tice s tímto pravidlem a na symboly na vrcholu všech zásobníku jsou aplikována pravidla dle této n-tice. Následně jsou z vrcholů zásobníků výstupních komponent odstraněny všechny terminály a jsou přidány k výstupním řetězcům příslušných komponent. Tím je zajištěno, že se na vrcholech zásobníků všech výstupních komponent opět vyskytuje neterminál.

$A \in T$ Je-li na vrcholu zásobníku vstupní komponenty terminál, pak je tento symbol porovnán se vstupním symbolem a v případě shody odstraněn. Pokud ke shodě nedojde, značí to syntaktickou chybu.

$A = \$$ Je-li na vrcholu zásobníku symbol \$ a zároveň je i na vrcholu všech zásobníků výstupních komponent a došlo-li k přečtení symbolu značícího konec vstupního řetězce, končí analýza úspěchem.

Příklad 2. *Uvažujme pravidlově synchronizovaný multigenerativní systém se dvěma komponentami* $G_1 = (N_1, T_1, P_1, S)$ *a* $G_2 = (N_2, T_2, P_2, Z)$*, kde množina pravidel komponenty* G_1 *je dána jako (malá písmena značí terminály, velká neterminály)* $P_1 = \{1 : S \to BD, 2 : B \to aBc, 3 : B \to C, 4 : C \to bC, 5 : C \to \varepsilon, 6 : D \to dD, 7 : D \to \varepsilon\}$*. Pravidla druhé komponenty jsou pak dána jako* $P_2 = \{1 : Z \to Z, 2 : Z \to X, 3 : X \to XX, 4 : X \to X, 5 : X \to \varepsilon\}$*. Množina kontrolních dvojic pak obsahuje prvky* $Q = \{(1,1), (2,1), (3,2), (4,3), (5,4), (6,5), (7,5)\}$*. LL tabulka vstupní gramatiky pak odpovídá tabulce 3.1.*

Tabulka 3.1: LL tabulka pro gramatiku G_1

	a	b	c	d	\$
S	1	1		1	
B	2	3			
C		4	5	5	5
D				6	7

Činnost toho systému pak popisuje tabulka 3.2. Vstupním řetězcem je řetězec aabccd. Vstupní komponenta pak provádí kontrolu počtu symbolů a a c. Druhá komponenta se pak stará o kontrolu počtu výskytů symbolů b a d, přičemž tato komponenta generuje při úspěchu prázdný řetězec. Tato kontrola je pak dána provázáním pravidel, která generují na zásobníku první komponenty symbol C. S těmito pravidly jsou provázána pravidla, která generují na druhém zásobníku neterminál X. Následně jsou tyto symboly X opět vymazávány, přičemž toto vymazávání je spojeno s generováním a odstraňováním symbolů D na zásobníku první komponenty. Vstupní komponenta pak přijímá jazyk $L = \{a^n b^m c^n d^m | n \geq 0, m \geq 0\}$, který není bezkontextový.

Tabulka 3.2: Ukázka činnosti slabého 2-KGP LL analyzátoru.

Vstup	Zásobník 1	Akce	Zásobník 2	Akce
aabccd	S	1	Z	1
aabccd	DB	2	Z	1
aabccd	$DcBa$	pop	Z	
abccd	DcB	2	Z	1
abccd	$DccBa$	pop	Z	
bccd	$DccB$	3	Z	2
bccd	$DccC$	4	X	3
bccd	$DccCb$	pop	XX	
ccd	$DccC$	5	XX	4
ccd	Dcc	pop	XX	
cd	Dc	pop	XX	
d	D	6	XX	5
d	Dd	pop	X	
$	D	7	X	5
$	$	úspěch	$	úspěch

3.1.2 Silný deterministický LL n-KGP

Možnou modifikací výše popsané metody je tzv. silný deterministický LL n-KGP. Tato modifikace uvolňuje některé požadavky na vstupní komponentu. Konkrétně pak nevyžaduje, aby vstupní komponenta striktně splňovala všechny požadavky na LL gramatiku (tedy aby pro danou kombinaci neterminálu a terminálu existovalo pouze jedno aplikovatelné pravidlo). Místo toho umožňuje, aby LL tabulka obsahovala v jednotlivých buňkách i několik možných pravidel a přesto bude výsledný systém pracovat deterministicky. Toho je docíleno využitím zásobníků ostatních komponent. Oproti slabé variantě, kde se příslušná n-tice vybírala pouze na základě vstupní komponenty. Silná varianta pak vybírá n-tici na základě neterminálů na vrcholu všech zásobníků. Tento mechanismus pak rozhoduje, které z pravidel obsažených v příslušné buňce dané LL tabulky použít. Takto pracující systém

pak požaduje, aby v případě, že levé strany pravidel pro jednotlivé komponenty dvou n-tic se shodují, pak musí platit, že $Predict(r) \cap Predict(p) = \emptyset$, kde p, q jsou pravidla vstupní komponenty daná těmito dvěma n-ticemi. Jinými slovy nesmí nastat případ, kdy buňka LL tabulky obsahuje více pravidel a pro tato pravidla navíc i existují stejné (z pohledu levých stran pravidel) n-tice.

3.2 Deterministická paralelní syntaktická analýza zdola nahoru pro n-KGP

Druhým přístupem k syntaktické analýze založené na multigenerativních gramatických systémech je analýza zdola nahoru. Ta využívá k zaručení determinismu LR analýzy. Opět je jedna z gramatik vstupní, a zbývající gramatiky jsou považovány za výstupní. Autor pak na rozdíl od analýzy shora dolů popisuje pouze slabou variantu analýzy zdola nahoru. Aby mohla analýza probíhat deterministicky, jsou na celý systém kladeny opět obdobné požadavky jako v předešlých případech. Konkrétně pak musí platit, že vstupní gramatika je LR a dále je požadováno, aby neexistovalo více řídicích n-tice se stejným pravidlem vstupní komponenty.

Oproti LL analýze, kde jednotlivé komponenty pracovali téměř stejně, v případě LR analýzy tomu tak není. Činnost komponent se liší v závislosti na tom, jedná-li se o komponentu vstupní či nikoliv. Vstupní komponenta provádí standardní LR analýzu. Jedinou změnou pak je, že je při použití redukčních pravidel třeba ověřit, že existuje příslušná řídicí n-tice. Výstupní komponenty pracují jiným způsobem. Na své zásobníky ukládají dvojice, které obsahují neterminál a část výstupního řetězce. Jakmile dojde ve vstupní komponentě k redukci, jsou na základě příslušné n-tice vybrána pravidla pro jednotlivé výstupní komponenty. Pravidla jsou zpracovávána směrem od konce pravé strany směrem k jejímu začátku po jednotlivých symbolech. Řetězec, který je v daném kroku ukládán na zásobník (značen jako w_{new}) je pak tvořen v závislosti na aktuálním zpracovávaném symbolu pravé strany pravidla. Je-li tímto symbolem (značen jako X) terminál, je tento symbol jednoduše konkatenován s aktuálním řetězcem, tedy $w_{new} = Xw_{new}$. Je-li tímto symbolem naopak neterminál, je porovnán zásobník, který obsahuje dvojice tvaru (A, w), kde A je neterminál a w je řetězec vygenerovaný v předešlých krocích. Platí-li že $A = X$, pak je tato dvojice odstraněna z vrcholu zásobníku a nový řetězec je vytvořen jako $w_{new} = ww_{new}$. Jakmile je takto zpracována celá pravá strana pravidla, je na vrchol zásobníku vložena dvojice (B, w_{new}), kde B odpovídá levé straně použitého pravidla. Jakmile vstupní komponenta dokončí analýzu, jsou prozkoumány zásobníky výstupních komponent. Pokud tyto zásobníky obsahují vždy právě jednu dvojici tvaru (S_i, w_i), kde S_i značí počáteční symbol příslušné komponenty, pak je analýza považována za úspěšnou a řetězce w_i jsou pak považovány za řetězce generované jednotlivými výstupními komponentami. V opačném případě je analýza považována za neúspěšnou. Tabulka 3.3 ukazuje činnost výstupní komponenty takto pracujícího systému.

Tabulka 3.3: Demonstrace činnosti výstupní komponenty LR n-KGP systému.

Pravidlo	Zásobník
$A \to \varepsilon$	
$A \to aAb$	$< A, \varepsilon >$
$A \to aAb$	$< A, a\varepsilon b >$
	$< A, aa\varepsilon bb >$

3.3 Deterministická paralelní syntaktická analýza pro n-HGP

Jak již název napovídá, nepoužívají tyto metody kanonické systémy (tedy ty, u kterých je v každé komponentě vždy přepsán nejlevější neterminál) ale systémy hybridní. Jednotlivé komponenty pak pracují tak, že vstupní komponenta pracuje kanonickým způsobem a komponenty ostatní způsobem obecným. Autor pak zavádí jak analýzu shora dolů, tak i zdola nahoru

3.3.1 Slabý deterministický LL n-HGP

První z těchto metod je založena na analýze shora dolů. Aby bylo možno takový analyzátor sestavit, jsou na jednotlivé komponenty kladeny jisté požadavky. Obdobně jako v případě kanonické varianty této metody je požadováno, aby vstupní gramatika byla LL gramatikou. Stejně tak je i nadále požadováno, aby byly všechny řídicí n-tice jednoznačně identifikovatelné na základě pravidla řídicí komponenty – tedy aby neexistoval více n-tic, které obsahují stejné pravidlo pro vstupní komponentu. Poslední požadavek je pak kladen na gramatiky výstupních komponent. V rámci těchto gramatik musí být zaručeno, že v rámci jednotlivých větných forem, které tvoří multiformu systému, se každý neterminál může vyskytovat vždy nejvýše jednou. Tím je pak zaručeno, že pravidla řídicí n-tice je možno aplikovat vždy jen jedním možným způsobem. Samotná analýza pak probíhá podobným způsobem jako v případě slabého LL n-KGP. Vstupní komponenta provádí na základě LL tabulky analýzu shora dolů. Jakmile je použito některé z expanzivních pravidel v rámci vstupní gramatiky, je vyhledána příslušná n-tice a na větné formy jednotlivých výstupních gramatik jsou aplikována pravidla přiřazená touto n-ticí. Díky požadavku na maximálně jeden výskyt každého neterminálu v každé možné větné formě, jsou tato pravidla aplikována vždy jediným možným způsobem. Analýza je pak úspěšná, jestliže vstupní gramatika úspěšně přečte celý řetězec a vrátí úspěch v rámci LL analýzy a pokud v tomto okamžiku neobsahuje větná forma žádné komponenty jakékoliv neterminály.

Příklad 3. *Mějme 2-HGP systém, kde první komponenta je dána pravidly* $P_1 = \{1 : S \to aS, 2 : S \to bS, 3 : S \to \varepsilon\}$ *a druhá pravidly* $P_2 = \{1 : Z \to aZ, 2 : Z \to bZ, 3 : Z \to \varepsilon\}$. *Množina kontrolních dvojic je pak tvaru* $Q = \{(1,1),(2,2),(3,3)\}$. *Z pravidel*

druhé komponenty je zřejmé, že podmínka o unikátním výskytu neterminálů v rámci větně formy je splněna. Stejně tak jsou i jednotlivé řídicí dvojice unikátní. LL tabulka pro vstupní komponentu je pak popsána tabulkou 3.4.

Tabulka 3.4: LL tabulka pro slabý 2-HGP LL analyzátor.

	a	b	$
S	1	2	3

Tento analyzátor pak přijímá jako vstupní řetězce všechny řetězce nad abecedou a, b. Výstupní komponenta pak generuje řetězce, ve kterých je tato posloupnost seřazena, tedy řetězce odpovídající regulárnímu výrazu a^*b^*. Činnost tohoto systému pro vstupní řetězec ababba je znázorněna tabulkou 3.5. Sloupec zásobník *popisuje obsah zásobníku vstupní komponenty. Vrchol zásobníku se nalézá vpravo. Sloupec výstup pak obsahuje větnou formu výstupní gramatiky. Výstupní řetězec je pak čten standardním způsobem zleva doprava.*

Tabulka 3.5: Ukázka činnosti slabého 2-HGP LL analyzátoru.

Vstup	Zásobník	Akce	Výstup	Akce
$ababba$	S	1	Z	1
$ababba$	Sa	pop	aZ	
$babba$	S	2	aZ	2
$babba$	Sb	pop	aZb	
$abba$	S	1	aZb	1
$abba$	Sa	pop	$aaZb$	
bba	S	2	$aaZb$	2
bba	Sb	pop	$aaZbb$	
ba	S	2	$aaZbb$	2
ba	Sb	pop	$aaZbbb$	
a	S	1	$aaZbbb$	1
a	Sa	pop	$aaaZbbb$	
$	S	3	$aaaZbbb$	3
$	$\$\varepsilon$	úspěch	$aaabbb$	úspěch

3.3.2 Slabý deterministický LR n-HGP

Poslední metodou popsanou v [3] je metoda syntaktické analýzy zdola nahoru, založená na n-HGP systémech. Tato metoda je tedy opět založena na hybridní variantě multigenerativních systémů. Na jednotlivé komponenty jsou kladeny obdobné požadavky jako v předešlém případě, konkrétně tedy musí být vstupní gramatika LR gramatikou, všechny kontrolní n-tice musí být jednoznačně identifikovatelné na základě pravidla vstupní komponenty a jednotlivé větné formy komponent nesmí obsahovat více výskytů jednotlivých

neterminálů. Analýza pak probíhá tím způsobem, že vstupní komponenta provádí standardní LR analýzu. Výstupní komponenty pak obsahují proměnné, které reprezentují jednotlivé neterminály. Na základě pravidel jednotlivých kontrolních n-tic jsou pak těmto proměnným přiřazovány části výstupního řetězce (formálně se jedná o homomorfismus z $N \cup T$ do T). Jakmile je v rámci vstupní komponenty použito redukční pravidlo, jsou na základě pravidla daného kontrolní n-ticí upraveny hodnoty obsažené v jednotlivých proměnných každé komponenty. Upravována je pak proměnná, která odpovídá levé straně daného pravidla. Její hodnota je pak nastavena v závislosti na symbolech na pravé straně tohoto pravidla. Terminální symboly jsou konkatenovány s hodnotou této proměnné, neterminály jsou pak nahrazeny hodnotou proměnné, která tento neterminál reprezentuje. Tato proměnná (s výjimkou případu, kdy se jedná o stejný symbol jako na levé straně právě používaného pravidla) je následně nastavena na hodnotu *nedefinováno*. Jakmile vstupní komponenta dokončí syntaktickou analýzu, obsahy proměnných, které odpovídají počátečním symbolům jednotlivých gramatik, jsou považovány zá výstupní řetězce těchto gramatik. Pokud pro některý z těchto řetězců w_i platí, že $w_i \notin T_i^*$, pak je analýza považována za neúspěšnou. Činnost výstupní komponenty je pak znázorněna tabulkou 3.6. Sloupce odpovídají proměnným odpovídajícím jednotlivým neterminálům (s vyjímkou sloupce pravidlo). Prázdné buňky pak odpovídají hodnotě *nedefinováno*.

Tabulka 3.6: Ukázka generování řetězce výstupní komponentou HGP systému.

Pravidlo	NP	N	DET	VT	PN	VP	S
$N \rightarrow man$							
$DET \rightarrow the$		man					
$NP \rightarrow DET\ N$		man	the				
$VT \rightarrow sees$	the man						
$VP \rightarrow VT\ NP$	the man			sees			
$NP \rightarrow John$						sees the man	
$S \rightarrow NP\ VP$	John					sees the man	
							John sees the man

Kapitola 4

Závěr

V rámci této práce byly představeny některé metody syntaktické analýzy, které nejsou založeny na standardních bezkontextových gramatikách. Tyto metody jsou založeny na gramatických systémech, konkrétně na tzv. multigenerativních systémech. Na základě těchto systémů pak bylo představeno několik metod syntaktické analýzy, přičemž některé z těchto metod pracují metodou shora dolů a některé metodou zdola nahoru.

Výhodou těchto metod oproti standardním metodám je, že jsou schopny analyzovat i některé jazyky, které není pomocí standardních metod možno úspěšně analyzovat. Další výhodou pak je, že tyto metody jsou navrženy pro práci s několika zadními částmi překladače. Díky tomu je pak například možné, generovat výstupní řetězce v několika variantách. Je tak například možno překládat vstupní soubor v programovacím jazyce do několika binárních souborů pro několik různých typů procesorů.

Literatura

[1] A.V. Aho, M.S. Lam, R. Sethi, and J.D. Ullman. *Compilers: Principles, Techniques, and Tools*. Alternative eText Formats Series. ADDISON WESLEY Publishing Company Incorporated, 2007.

[2] D. Kolář. Simulation of llk parsers with wide context by automaton with one-symbol reading head. In *Proceedings of 38th International Conference MOSIS '04 - Modelling and Simulation of Systems*, pages 347–354, 2004.

[3] R. Lukáš. *Multigenerativní gramatické systémy*. PhD thesis, Vysoké Učení Technické, v Brně, 2006.

[4] Alexander Meduna. *Automata and Languages: Theory and Applications [Springer, 2000]*. Springer Verlag, 2005.

[5] G. Rozenberg and A. Salomaa. *Handbook of Formal Languages: Volume 2. Linear Modeling: Background and Application*. Handbook of Formal Languages. Springer Berlin Heidelberg, 1997.