VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYNTAKTICKÁ ANALÝZA ZALOŽENÁ NA NĚKOLIKA GRAMATIKÁCH

PARSING BASED ON SEVERAL GRAMMARS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ONDŘEJ KOUMAR**
AUTHOR

VEDOUCÍ PRÁCE **prof. RNDr. ALEXANDR MEDUNA, CSc.**
SUPERVISOR

BRNO 2024

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Citace

KOUMAR, Ondřej. *Syntaktická analýza založená na několika gramatikách.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Alexandr Meduna, CSc.

Syntaktická analýza založená na několika gramatikách

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondřej Koumar
18. března 2024

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Obsah

1 Úvod **3**

2 Abstrakt **4**

3 Základy teorie formálních jazyků **5**
3.1 Abeceda, řetězec a jazyk 5
3.2 Chomského hierarchie 5
3.3 Bezkontextová gramatika 5
3.4 Konečný automat 5
3.5 [[doplnit vice veci?]] 5

4 Důležité pojmy syntaktické analýzy [[rozdelit na vice kapitol? srazit nektere pojmy do jedne sekce/rozdelit do vice? jake dalsi pojmy doplnit?]] **6**
4.1 Derivační krok 6
4.2 Množiny potřebné k sestrojení LL tabulky 6
4.3 LL tabulka 6
4.4 Zásobníkový automat 7
4.5 Prediktivní syntaktická analýza 7
4.6 Precedenční tabulka 7
4.7 Precedenční syntaktická analýza 7
4.8 Abstraktní syntaktický strom 7

5 Cooperating distributed gramatický systém **8**
5.1 Derivační krok v CDGS 8
5.2 Jazyk generovaný CD gramatickým systémem 9

6 Implementace syntaktického analyzátoru pro jazyk Koubp [[rozdelit jinak sekce? co pripadne z implementace doplnit?]] **10**
6.1 Přijímaný jazyk 10
6.2 Gramatický systém definující syntax jazyka Koubp 11
6.3 Návrh řešení syntaktického analyzátoru 11
6.4 Lexikální analýza a nástroj Flex 12
6.5 Abstraktní syntaktický strom 12

7 Testování **13**

8 Závěr **14**

Literatura **15**

Seznam obrázků

6.1 [[Tady bude obrázek znázorňující oba dva parsery a šest gramatik a který s čím pracuje.]] 11
6.2 [[tady bude vytazek z kodu pro prepinani analyzatoru.]] 12
6.3 [[Tady bude obrazek, kde bude napsany jednoduchy kod volani funkce, jeho reprezentace v tokenech a navic vlozene pomocne tokeny, ktere parsery vyuzivaji. Bude tam zobrazene, v jakych mistech se provadi zmena analyz.]] 12

Kapitola 1

Úvod

Kapitola 2

Abstrakt

Kapitola 3

Základy teorie formálních jazyků

3.1 Abeceda, řetězec a jazyk

3.2 Chomského hierarchie

3.3 Bezkontextová gramatika

Definice 1. *Bezkontextová gramatika* je čtveřice $G = (N, T, P, S)$, kde:

- N je konečná množina neterminálních symbolů,
- T je konečná množina terminálních symbolů,
- P je konečná množina přepisovacích pravidel ve tvaru $A \to x$, $A \in N$ a $x \in (N \cup \Sigma)^*$,
- $S \in N$ je výchozí symbol gramatiky.

[[derivacni krok sem misto k syntakticke analyze?]]

3.4 Konečný automat

Definice 2.

3.5 [[doplnit vice veci?]]

Kapitola 4

Důležité pojmy syntaktické analýzy [[rozdelit na vice kapitol? srazit nektere pojmy do jedne sekce/rozdelit do vice? jake dalsi pojmy doplnit?]]

4.1 Derivační krok

Myšlenka *derivačního kroku* je aplikovat pravidlo z množiny pravidel bezkontextové gramatiky, čímž se část původního řetězce přepíše na novou.

Definice 3. Nechť $G = (N, T, P, S)$ je BKG, $u, v \in (N \cup T)^*$ a $p = A \to x \in P$.
Potom uAv přímo derivuje uxv za použití p v G, zapsáno $uAv \Rightarrow uxv[p]$, zjednodušeně $uAv \Rightarrow uxv$.
Je možné říci, že G provádí derivační krok z uAv do uxv.

Sekvence derivačních kroků

[[definici jsem nasel v prezentacich ifj, nicmene to nevypada nejformalneji; bude stacit nebo radsi z literatury?]]

4.2 Množiny potřebné k sestrojení LL tabulky

[[tady by mohlo byt Empty(x), First(x), Follow(x), Predict(x) + algoritmy pro jejich sestaveni?]]

4.3 LL tabulka

[[algoritmus pro sestaveni tabulky?]]

4.4 Zásobníkový automat

Zásobníkový automat je rozšíření konečného automatu, popsaného v definici 2, o zásobník, matematicky přesněji o zásobníkovou abecedu a počáteční symbol na zásobníku.

Definice 4. *Zásobníkový automat* (ZA) je sedmice $M = (Q, \Sigma, \Gamma, R, s, S, F)$, kde:

- Q je konečná množina stavů,
- Σ je vstupní abeceda,
- Γ je zásobníková abeceda,
- R je konečná množina pravidel tvaru $Apa \to wq$, kde $A \in \Gamma$, $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$,
- $s \in Q$ je počáteční stav,
- $S \in \Gamma$ je počáteční symbol na zásobníku,
- $F \subseteq Q$ je množina koncových stavů.

Rozšířený zásobníkový automat

Původní zásobníkový automat lze rozšířit o další chování. Například o možnost čtení více symbolů ze zásobníku než původního jednoho, tedy při přechodech mezi stavy měnit celé řetězce na vrcholu zásobníku.

Definice 5. *Rozšířený zásobníkový automat* (RZA) je sedmice $M = (Q, \Sigma, \Gamma, R, s, S, F)$, kde:

- $Q, \Sigma, \Gamma, s, S, F$ jsou definovány stejně jako u ZA,
- R je konečná množina pravidel ve tvaru $vpa \to wq$, kde $v, w \in \Gamma^*$, $p, q \in Q$, $a \in \Sigma \cup \{\varepsilon\}$.

4.5 Prediktivní syntaktická analýza

4.6 Precedenční tabulka

[[algoritmus pro sestrojeni tabulky?]]

4.7 Precedenční syntaktická analýza

4.8 Abstraktní syntaktický strom

Kapitola 5

Cooperating distributed gramatický systém

[[nejsem si jisty prekladem, zatim nechavam v anglictine, ale vypada to divne]] Cooperating distributed gramatický systém (CDGS) stupně n je systém gramatik, které mezi sebou sdílejí množinu neterminálů i terminálů a startovací symbol.

Definice 6. CDGS je n-tice $\Gamma = (N, T, S, P_i, \ldots, P_n)$ pro $1 \leq i \leq n$, kde:

- N, T, a S jsou definovány stejně jako v definici 1,
- P_i je konečná množina pravidel ve tvaru $A \to x$, kde A i x jsou definovány stejně jako v definici 1, nazývaná *komponentou* systému,
- i-tá gramatika systému se zapisuje jako $G_i = (N, T, S, P_i)$

Alternativní definice pro CDGS je $\Gamma = ((N, T, S, P_1), \ldots, (N, T, S, P_n))$.

[[staci takovyto popis? mam se vyhnout vysvetlovani "vlastnimi slovy"nebo je naopak dobre, ze pred definici +- uvedu, o co se jedna?]]

5.1 Derivační krok v CDGS

Notace derivačního kroku v CDGS je

$$x_i \Rightarrow^f y,$$

což znamená, že řetězec $x \in (N \cup T)^*$ derivuje řetězec $y \in (N \cup T)^*$ v i-té komponentě za použití *derivačního režimu* f.

Derivační režimy

Prvním a nejpřirozenějším příkladem je režim $*$ **[[Jak tento režim nazvat?]]**. V tomto případě stačí, aby řetězec y byl derivovatelný z řetězce x v i-té komponentě, zapsáno $x \Rightarrow^* y$ v $G_i = (N, T, P_i, S)$.

Podobným příkladem je režim *ukončovací*, který spočívá v nutné derivaci řetězce v dané komponentě, dokud je to možné. Značí se písmenem t. Jsou dvě nutné podmínky, aby y bylo derivovatelné z x v komponentě G_i režimem t.

1. $x \Rightarrow^* y$ v $G_i = (N, T, P_i, S)$ – v dané komponentě lze posloupností derivačních kroků získat řetězec y z řetězce x,
2. $y \not\Rightarrow z$ pro všechna $z \in (N \cup T)^*$ – není jiný další řetězec, který by z y šel odvodit.

Další derivační režimy:

- alespoň k derivací, tedy $x_i \Rightarrow^{\geq k} y$,
- nejvíce k derivací, tedy $x_i \Rightarrow^{\leq k} y$,
- právě k derivací, tedy $x_i \Rightarrow^{=k} y$,

kde $k \in \mathbb{N} \cup \{0\}$ a i symbolizuje i-tou komponentu gramatického systému.

Derivační režimy mohou být reprezentovány jako množina, což pomůže definovat další pojmy v následující podkapitole o generovaných jazycích.

Definice 7. Nechť $k \in \mathbb{N}$ a $*$, t představují derivační režimy.
Potom množina $D = \{*, t\} \cup \{\leq k, \geq k, = k\}$ reprezentuje derivační režimy použitelné v CD gramatických systémech.

5.2 Jazyk generovaný CD gramatickým systémem

Než bude definován samotný jazyk, je vhodné definovat pomocnou množinu, která reprezentuje *možné derivace* z řetězců.

Definice 8. Nechť $\Gamma = (N, T, S, P_i, \ldots, P_n)$.
Potom $F(G_j, u, f) = \{v : u_j \Rightarrow^f v\}$, $1 \leq j \leq n$, $f \in D$, $u \in (N \cup T)^*$ je množina všech řetězců v derivovatelných z u v j-té komponentě za použití derivačního režimu f.

Definice 9. Nechť $\Gamma = (N, T, S, P_i, \ldots, P_n)$.
Jazyk generovaný systémem Γ za derivačního režimu f, $L_f(\Gamma) = \{w \in T^* :$ existují $v_0, v_1, \ldots, v_m$ takové, že $v_k \in F(G_{j_k}, v_{k-1}, f)$, $1 \leq k \leq m$, $1 \leq j_k \leq n$, $v_0 = S$, $v_m = w$ pro $m \geq 1\}$.

[[jsou definice v tomto formatu v poradku?]]

Výsledný řetězec w, který vznikl postupnou derivací startovacího symbolu v_0. Měl několik mezikroků, které jsou reprezentovány řetězci $v_1, \ldots, v_{m-1}$. Každý řetězec v_k, kde $1 \leq k \leq m$ byl zderivován z řetězce v_{k-1} v komponentě G_{j_k}, kde $1 \leq j_k \leq n$, za derivačního režimu f.

Kapitola 6

Implementace syntaktického analyzátoru pro jazyk Koubp [[rozdelit jinak sekce? co pripadne z implementace doplnit?]]

[[uvod do teto kapitoly - navod na pouziti a build system?]]

6.1 Přijímaný jazyk

Jazyk *Koubp* je založený na jazyce IFJ22, který je podmnožinou jazyka PHP 8, jenž byl specifikován v rámci zadání projektu do předmětu Formální jazyky a překladače v akademickém roce 2022/2023. **[[idealne citovat zadani projektu?]]**

Některé aspekty jazyků jsou společné. Oba dva jazyky jsou strukturované, podporují definici proměnných a funkcí. Hlavní tělo programu se skládá z prolínání sekvence příkazů a definic funkcí, které se mohou vzájemně rekurzivně volat. Vstupním bodem programu není funkce `main()`, jak lze nalézt například u jazyka C [1], analýza probíhá od začátku souboru. V uživatelem definovaných funkcích může být větvení, iterace a další běžné konstrukce. Veškeré proměnné jsou lokální, i v rámci hlavního těla programu. Soubory se zdrojovým kódem nelze slučovat a vytvářet tak jediný modul, který by bylo možné zkompilovat. **[[doplnit veci, ktere nejsou spolecne (nezabihat do detailu, vse bude specifikovano v podnadpisech)]]**

Deklarace a definice funkcí

Příkazy

- Přiřazení
- Větvení
- Cyklus while
- Cyklus for
- Volání funkcí

Výrazy

- Operátory
- Priorita operátorů
- Volání funkcí

6.2 Gramatický systém definující syntax jazyka Koubp

[[tady si nejsem uplne jisty - gramaticky system bych dal urcite do prilohy, ale rad bych zminil princip indexace + jak to vypada v ll tabulce a implementacne vyreseny deadlock.]]

Indexace neterminálů a význam pro implementaci

Deadlock mezi neterminály statement a codeBlock

6.3 Návrh řešení syntaktického analyzátoru

Implementace syntaktické analýzy je silně objektově orientována, veškeré datové struktury jsou reprezentovány třídami. Třídy reprezentující neterminály a terminály mají společnou nadtřídu, díky čemuž je možné je ukládat do jednoho zásobníku. Podobně je to i s jednotlivými analyzátory, a proto je možné implementovat předávání řízení pomocí výjimek bez nutnosti neustálého vzájemného volání při analýze volání funkcí. Jednu nadtřídu mají také gramatiky, jednotlivé instance jsou poté konstruovány pomocí tovární metody.

Práce s gramatikami

Gramatiky jsou rozděleny tak, aby každá z nich tvořila ucelenou část jazyka Koubp. Mimo čtvrtou gramatiku jsou všechny využívány pouze pro prediktivní analýzu. Ačkoliv téma čtvrté gramatiky je analýza výrazů, je využívána jak precedenčním, tak prediktivním analyzátorem. Důvod je zvolený postup prediktivní analýzy volání funkcí – o samotné výrazy bez volání funkcí, tedy pouze s konstantami, proměnnými a podobně, se postará analýza precedenční.

Obrázek 6.1: **[[Tady bude obrázek znázorňující oba dva parsery a šest gramatik a který s čím pracuje.]]**

Předávání řízení prediktivní a precedenční analýzy

V programu je implementován systém výjimek, pro potřeby této podkapitoly je třeba znát výjimku (třídu) `ChangeParser`. Třída `Parser` deklaruje čistě virtuální metodu `Parse()`, která je přepsána oběma analyzátory. Kdykoliv se chytí výjimka `ChangeParser`, do ukazatele na třídu `Parser` se uloží adresa správného analyzátoru.

V programu je několik podnětů, které tuto výjimku vyvolají. Jedním z nich je neterminál `Expression` na vrcholu zásobníku pro prediktivní analýzu. V tomto případě je zřejmé, že

Obrázek 6.2: **[[tady bude vytazek z kodu pro prepinani analyzatoru.]]**

musí být předáno řízení precedenční analýze, aby mohla být provedena analýza výrazu. Pokud precedenční analýza narazí na token `tFuncName`, reprezentující počátek volání funkce, musí opět předat řízení zpět prediktivní analýze.

[[presny popis zatim nemohu poskytnout, je to posledni vec, ktera mi chybi k dokonceni SA. nicmene bude doplnen.]]

Obrázek 6.3: **[[Tady bude obrazek, kde bude napsany jednoduchy kod volani funkce, jeho reprezentace v tokenech a navic vlozene pomocne tokeny, ktere parsery vyuzivaji. Bude tam zobrazene, v jakych mistech se provadi zmena analyz.]]**

6.4 Lexikální analýza a nástroj Flex

Pro usnadnění práce byl lexikální analyzátor automaticky vygenerován nástrojem *Flex*[1] ze souboru s lexémy popsanými regulárními výrazy. S každým úspěšně analyzovaným lexémem následuje vložení tokenu do vstupní pásky pro syntaktickou analýzu. Znamená to tedy, že se nejdříve v celém zdrojovém souboru ověří lexikální správnost, až poté správnost syntaktická. Při konstrukci tokenu se inicializuje i jeho datová část.

6.5 Abstraktní syntaktický strom

[1] Manuál k programu k dispozici na https://westes.github.io/flex/manual/.

Kapitola 7

Testování

Kapitola 8

Závěr

Literatura

[1] International Organization for Standardization (ISO). *ISO/IEC 9899:2018, Programming languages - C*. International Organization for Standardization (ISO), 2018. Dostupné z: https://www.open-std.org/jtc1/sc22/wg14/www/docs/n2310.pdf.