VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÁSOBNÍKOVÉ SYSTÉMY A SYNTAKTICKÁ ANALÝZA ZALOŽENÁ NA NICH

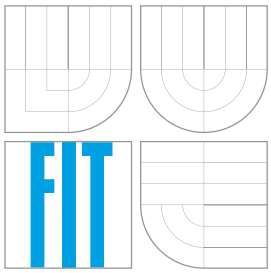
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZDENĚK KŘESŤAN

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

ZÁSOBNÍKOVÉ SYSTÉMY A SYNTAKTICKÁ ANALÝZA ZALOŽENÁ NA NICH

PUSHDOWN SYSTEMS AND PARSING BASED ON THEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ZDENĚK KŘESŤAN
AUTHOR

VEDOUCÍ PRÁCE — prof. RNDr. ALEXANDER MEDUNA, CSc.
SUPERVISOR

BRNO 2015

Abstrakt

Tato práce je zaměřena na využití hlubokého zásobníkového automatu v syntaktické analýze, který svou sílou přesahuje sílu bezkontextových gramatik. Zavádí modifikaci algoritmu prediktivní syntaktické analýzy řízené LL tabulkou o možnost práce s hlubokým zásobníkovým automatem. Jsou zde zavedeny také modifikované LL gramatiky rozšířené o hloubku, které jsou nutné pro práci s tímto automatem.

Abstract

This thesis focuses on the use of deep pushdown automaton in parsing. This method overcomes the power of traditional context-free grammars. The predictive parsing algorithm, driven by LL table, is modified by the use of deep pushdown automaton. The modifications extended LL tables by adding depth, in order for tables to collaborate with the deep pushdown automaton.

Klíčová slova

zásobníkové automaty, hluboké zásobníkové automaty, prediktivní syntaktická analýza, gramatiky, LL gramatiky, LL tabulka

Keywords

pushdown automata, deep pushdown automata, predictive parsing, grammars, LL grammars, LL table

Citace

Zdeněk Křesťan: Zásobníkové systémy a syntaktická analýza založená na nich, bakalářská práce, Brno, FIT VUT v Brně, 2015

Zásobníkové systémy a syntaktická analýza založená na nich

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. RNDr. Alexandra Meduny, CSc.

.....
Zdeněk Křesťan
11. května 2015

Poděkování

Chtěl bych zde poděkovat vedoucímu, profesoru RNDr. Alexandru Medunovi, CSc. za odborný dohled a konzultace.

Obsah

Kapitola 1

Úvod

V dnešní době využívá většina programovacích jazyků bezkontextové gramatiky. Bezkontextové gramatiky jsou díky své jednoduchosti oblíbené, ale jejich vyjadřovací síla by mohla časem nedostačovat. Naopak kontextové gramatiky, které na rozdíl od bezkontextových gramatik pracují i s okolním kontextem, disponují větší vyjadřovací sílou. Nevýhodou však zůstává poměrně složitá práce s jejich využitím při syntaktické analýze. Z tohoto důvodu je nutné vytvořit nové metody pro syntaktickou analýzu, které by přesahovaly vyjadřovací sílu bezkontextových gramatik a blížily se tak k síle kontextových gramatik.

Řešením tohoto problému je možnost využití v syntaktické analýze hlubokých zásobníkových automatů, které dovolují na rozdíl od zásobníkových automatů pracovat i s neterminálními symboly v hloubce zásobníku (viz 3.3). Hluboký zásobníkový automat poprvé představil Alexander Meduna v článku časopisu Acta Informatica [2] v roce 2006. V tomto článku je také dokázána ekvivalence síly hlubokého zásobníkového automatu se sílou n-omezených stavových gramatik.

N-omezené stavové gramatiky stojí mezi bezkontextovými a kontextovými gramatikami. Jejich vyjadřovací síla je větší než u bezkontextových gramatik, ale nepokrývají všechny kontextové jazyky.

Cílem této práce je tedy nahradit stávající zásobníkový automat v prediktivní syntaktické analýze hlubokým zásobníkovým automatem a tím přesáhnout vyjadřovací sílu bezkontextových gramatik, které jsou právě ekvivalentní se zásobníkovým automatem. Úkolem bylo tedy modifikovat algoritmus pro prediktivní syntaktickou analýzu tak, aby pracoval s hlubokým zásobníkovým automatem.

V teoretické části této práce jsou v prvních kapitolách představeny základní pojmy a definice z oblasti formálních jazyků a překladačů. Kapitoly 2 a 3 mimo jiné popisují rozdělení gramatik podle tzv. Chomského hierarchie, kterou vytvořil Noam Chomsky již v roce 1956, a následně automaty ekvivalentní k těmto gramatikám. Následující kapitola popisuje základní modely syntaktické analýzy a to zejména analýzu shora dolů, prováděnou metodou prediktivní syntaktické analýzy.

Jako podpůrné materiály byly využity knihy z oblasti formálních jazyků: Regulated Grammars and Automata [4] a Formal Languages and Computation [3]. Nicméně i přes popsání nejdůležitějších pojmů a definic je práce určena pro odborného čtenáře.

Po teoretické části této práce je v kapitole 5 zaveden modifikovaný algoritmus, který umožňuje expanzi neterminálních symbolů mimo vrchol zásobníku (v dané hloubce) a zaznamenává hodnotu aktuální hloubky nutnou pro provedení této expanze. Po úpravě algoritmu je nutné modifikovat i LL gramatiku, která bude mít odlišný tvar pravidel a LL tabulky. Pravidla musí navíc obsahovat údaj o hloubce, v jaké mohou být využita k expanzi

příslušného neterminálního symbolu. Z toho vyplývá potřeba přidání hodnoty hloubky do LL tabulky v podobě třetího rozměru.

Funkce algoritmu (kap. 6) je demonstrována na dvou vybraných jazycích, které na základě uvedených důkazů nejsou bezkontextové. Proto je tedy nelze přijímat syntaktickým analyzátorem, který využívá zásobníkový automat. Na základě vytvořených gramatik je ukázána krok za krokem funkce algoritmu při analýze vstupního řetězce (slova).

Poslední kapitola 7 je věnována popisu implementace algoritmu v podobě jednoduché demonstrační aplikace s uživatelským rozhraním. Implementace je provedena pomocí jazyku Java s využitím knihovny Swing pro tvorbu uživatelského rozhraní. Aplikace umožňuje výběr mezi zmíněnými jazyky a následně zadání vstupního řetězce (slova). Po potvrzení je provedena kontrola, zobrazen postup a vyhodnocení zda daný řetězec patří do vybraného jazyka.

Kapitola 2

Abecedy, jazyky a gramatiky

V této kapitole si představíme formální jazyky a popíšeme základní prostředky pro jejich definici. Hlavním prostředkem pro popis formálních jazyků jsou gramatiky, které jsou děleny podle vyjadřovací síly (Chomského hierarchie). Automaty, které jsou ekvivalentní k těmto typům gramatik si představíme samostatně v kapitole 3. Mezi základní části gramatik patří jednotlivé abecedy a slova.

2.1 Abeceda a slovo

Abeceda je konečná neprázdná množina symbolů, která je značena velkými písmeny řecké abecedy (například: $\sum$ nebo Γ). Základním prvkem jsou slova (řetězce), která jsou tvořena symboly z dané abecedy. Prázdné slovo (řetězec) značíme jako ϵ.

Definice 2.1.1.
Nechť $\sum$ je abeceda, kde

- *ϵ je řetězec (slovo) této abecedy*
- *pokud řetězce (slova) $x, y \in \sum$ pak i řetězec (slovo) $xy \in \sum$*

Množina všech slov (řetězců) abecedy $\sum$, která obsahuje prázdné slovo je značena jako $\sum^*$, pak $\sum^+$ značí množinu bez prázdného slova (ϵ). Pokud máme řetězec (slovo) w, kde $w \in \sum^*$ pak $|w|$ je počet symbolů v tomto slově.

2.2 Jazyky

Formální jazyk je libovolná množina slov dané abecedy, tedy i libovolná podmnožina $\sum^*$. Pro libovolný jazyk nad abecedou $\sum$ tedy platí $L \subseteq \sum^*$.

Jazyky dělíme na konečné a nekonečné. Pokud obsahuje konečný počet řetězců je konečný, jinak je jazyk nekonečný. Na rozdíl od konečných jazyků nejde nekonečné jazyky popsat výčtem jejich slov. K popisu těchto jazyků je nutné využít jiné nástroje, jako jsou gramatiky nebo automaty, které dokážou popsat oba typy jazyků.

2.3 Gramatiky

Gramatika je složena z konečné množiny pravidel, které dokáží generovat jazyk, pro který je gramatika vytvořena. Aplikováním těchto pravidel postupně přepisujeme příslušné symboly a vytváříme tím slova daného jazyka. Základním modelem pro rozdělení gramatik podle jejich síly je tzv. Chomského hierarchie, kterou vytvořil Noam Chomsky a publikoval v [1].

2.3.1 Chomského hierarchie gramatik

Chomského hierarchie rozděluje gramatiky podle síly do čtyř skupin, gramatiky typu 0 jsou nejsilnější a gramatiky typu 3 jsou nejslabší. Gramatiky mají stejný základ, ale odlišný tvar pravidel. Pro pravidla tedy platí, že čím větší jsou na ně kladena omezení, tím menší vyjadřovací sílu má tato gramatika. U této hierarchie platí, že gramatiku vyššího typu můžeme vyjádřit nižším typem (například: gramatika typu 1 je vyjádřitelná typem 0).

- **typ 0** - Neomezené nebo frázové gramatiky, které nemají žádné omezení na tvar pravidel. Jazyky generované těmito gramatikami mohou být rozpoznány Turingovým strojem (model skládající se z konečného automatu, pravidel přechodové funkce a nekonečné pásky pro mezivýsledky) a někdy se pojmenovávají jako jazyky rekurzivně společné.
- **typ 1** - Kontextové gramatiky jsou podmnožinou typu 0 a generují kontextové jazyky, které jsou rozpoznány lineárním Turingovým strojem (lineární funkce omezující délku pásky).
- **typ 2** - Bezkontextové gramatiky jsou podmnožinou obou předchozích typů a generují bezkontextové jazyky, které jsou využívány při popisu syntaxe většiny programovacích jazyků. Tyto jazyky se dají rozpoznat zásobníkovým automatem.
- **typ 3** - Regulární gramatiky jsou nejnižší podmnožinou všech ostatních typů. Jazyky generované těmito gramatikami se nazývají regulární jazyky a lze je rozpoznat pomocí konečného automatu.

2.3.2 Důležité pojmy

Důležité pojmy pro definici gramatiky:

1. **Terminál** (terminální symbol) - jedná se o symbol z množiny symbolů jazyka, který je přijímán danou gramatikou. Tento symbol v procesu analýzy není nahrazován.
2. **Neterminál** (neterminální symbol) - symbol sloužící k rozvinutí pomocí pravidel gramatiky.
3. **Počáteční symbol** (startovací) - jedná se o speciální neterminální symbol, ze kterého jsou generovány všechny řetězce (slova) daného jazyka.
4. **Pravidlo** - jedná se o přepis, který specifikuje nahrazení neterminálů jiným řetězcem (terminálů a neterminálů).

2.3.3 Kontextové gramatiky

Kontextové gramatiky představil Noam Chomsky při snaze popsat přirozený jazyk, kde se určité slovo dá použít na základě okolního kontextu. Kontextové jazyky lze rozpoznat lineárním Turingovým strojem. Pravidla u těchto gramatik mají tvar: $\alpha A\beta \rightarrow \alpha\gamma\beta$, kde A je neterminální symbol, α a β jsou řetězce z neterminálů a terminálů tvořící kontext a γ je neprázdný řetězec terminálů a neterminálů.

Definice 2.3.1.
Kontextová gramatika je definována jako uspořádaná čtveřice $G = (N, T, P, S)$, kde

N je abeceda neterminálních symbolů

T je abeceda terminálních symbolů, kde $(N \cap T) \neq \emptyset$

P je konečná množina pravidel tvaru: $\alpha A\beta \rightarrow \alpha\gamma\beta$, kde $A \in N$, $\alpha, \beta \in (N \cup T)^*$, $\gamma \in (N \cup T)^+$

$S \in N$ je počáteční neterminální symbol

Derivační krok:
G je kontextová gramatika. Nechť $\alpha, \beta \in (N \cup T)^*$ a $p = (\alpha A\beta \rightarrow \alpha\gamma\beta) \in P$, potom se provádí krok z $\alpha A\beta$ do $\alpha\gamma\beta$ využitím pravidla p ($\alpha A\beta \Rightarrow \alpha\gamma\beta$).

2.3.4 Bezkontextové gramatiky

Bezkontextové gramatiky se používají pro definování syntaxe programovacích jazyků a generují tzv. bezkontextové jazyky. Jazyky jsou právě bezkontextové, pokud je lze akceptovat vhodným zásobníkovým automatem. Tyto gramatiky generují řetězce daného jazyka za použití konečné množiny gramatických pravidel. Pravidla jsou tvaru: $A \rightarrow x$, kde A je neterminální symbol a x je řetězec, který může být složený z terminálních a neterminálních symbolů. Jako speciální typ pravidla je využit tvz. ϵ-pravidlo ($A \rightarrow \epsilon$), které slouží pro odstranění daného neterminálu.

Definice 2.3.2.
Bezkontextová gramatika je definována jako uspořádaná čtveřice $G = (N, T, P, S)$, kde

N je abeceda neterminálních symbolů

T je abeceda terminálních symbolů

P je konečná množina pravidel tvaru: $A \rightarrow x$, kde $A \in N$, $x \in (N \cup T)^*$

$S \in N$ je počáteční neterminální symbol

Derivační krok:
G je bezkontextová gramatika. Nechť $u, v \in (N \cup T)^*$ a $p = (A \rightarrow x) \in P$, potom se provádí krok z uAv do uxv využitím pravidla p ($uAv \Rightarrow uxv$).

2.3.5 N-omezená stavová gramatika

N-omezené stavové gramatiky svojí sílou stojí mezi bezkontextovými a kontextovými gramatikami. Vyjadřovací síla těchto gramatik je ekvivalentní se sílou hlubokého zásobníkového automatu, což bylo dokázáno v [2].

Definice 2.3.3.
N-omezená stavová gramatika je definována jako uspořádaná šestice $G = (N, T, P, S, W, p_0)$, *kde*

N *je abeceda neterminálních symbolů*

T *je abeceda terminálních symbolů*

P *je konečná množina pravidel tvaru:* $(p, A) \rightarrow (q, v)$, *kde* $A \in N$, $p, q \in W$, $v \in (N \cup T)^+$

$S \in N$ *je startující neterminální symbol*

W *je konečná množina stavů*

$p_0 \in W$ *počáteční stav*

Kapitola 3

Automaty a jejich modifikace

Automaty se využívají jako teoretické výpočetní modely využívané pro studium formálních jazyků. V této kapitole si ukážeme postupný vývoj automatů až po hluboký zásobníkový automat, který je klíčový pro tuto práci. Nejčastěji slouží automaty ke kontrole, zda daný řetězec patří do jazyka, pro který je automat vytvořen.

3.1 Konečný automat

Konečný automat (angl. *finite state machine*) je důležitý pro konstrukci překladačů. Je využitý v lexikální analýze, která tvoří první fázi překladu. Tento automat funguje na principu konečného počtu stavů, mezi kterými se pohybuje. Přechod mezi jednotlivými stavy je řízen pomocí symbolu ze vstupní pásky. Symbol je přečten pomocí čtecí hlavy, která ukazuje na právě čtený symbol a po přečtení se posouvá na další. Na začátku je čtecí hlava umístěna nad nejlevějším symbolem a postupuje směrem doprava až na konec vstupního řetězce. Řízení začíná v předem specifikovaném počátečním stavu a správnost kontrolovaného řetězce je ověřena, pokud automat končí v jednom z koncových stavů po přečtení všech symbolů ze vstupní pásky. Konečné automaty jsou svou sílou ekvivalentní s regulárními jazyky. Tedy platí, že pro každý konečný automat M existuje regulární výraz r tedy: $L(M) = L(r)$.

Definice 3.1.1.
Konečný automat je definován jako uspořádaná pětice $M = (Q, \sum, R, s, F)$, *kde*

Q *je konečná neprázdná množina stavů*

$\sum$ *je množina vstupních symbolů (vstupní abeceda)*

R *je konečná množina pravidel tvaru:* $pa \rightarrow q$, $p, q \in Q$, $a \in \sum \cup \{\epsilon\}$

$s \in Q$ *je počáteční stav*

$F \subseteq Q$ *je množina všech koncových stavů*

Konfigurace: $\chi \in Q \sum *$
Přechod: *Nechť* pax *a* qx *jsou konfigurace konečného automatu* M, *kde* $p, q \in Q$, $a \in \sum \cup \{\epsilon\}$ *a* $x \in \sum^*$ *a zároveň* $r = pa \rightarrow q \in R$ *je pravidlo. Potom může konečný automat* M *provést přechod z* pax *do* qx *za použití* r *(*$pax \vdash qx$*).*
Přijímaný jazyk: *Nechť* M *je konečný automat, který přijímá jazyk* $L(M)$. *Jazyk* $L(M)$ *je pak definován:* $L(M) = \{w : w \in \sum *, sw \vdash^* f, f \in F)$.

3.2 Zásobníkový automat

Zásobníkový automat (angl. *pushdown automaton*) se využívá jako teoretický model syntaktického analyzátoru. Je podobný konečnému automatu, skládá se též ze vstupní pásky obsahující vstupní řetězec, konečného stavového řízení a čtecí hlavy. Zásobníkový automat obsahuje na rozdíl od konečného automatu navíc teoreticky nekonečný zásobník. Každý přechod je tedy závislý nejen na vstupním symbolu a stavu, ale i na symbolu z vrcholu zásobníku.

Automat provádí dvě základní operace při kontrole vstupního řetězce - vyjmutí a expanzi. Pokud se na vrcholu zásobníku nachází stejný symbol jako je právě čtený symbol z pásky, tak se provede vyjmutí toho symbolu ze zásobníku a čtecí hlava se posune na další symbol. Při výskytu neterminálního symbolu na vrcholu zásobníku se provede jeho expanze podle pravidla.

Zásobník v tomto automatu pracuje na principu „LIFO“, tedy kdo poslední přijde tak jako první odchází. Znamená to tedy, že symbol, který je jako poslední vložen do zásobníku je na jeho vrcholu. Tento zásobník tedy pracuje pouze se symbolem na vrcholu zásobníku.

Definice 3.2.1.
Zásobníkový automat je definován jako uspořádaná sedmice $M = (Q, \sum, \Gamma, R, s, S, F)$*, kde*

Q *je konečná neprázdná množina stavů*

$\sum$ *je množina vstupních symbolů (vstupní abeceda)*

Γ *je množina zásobníkových symbolů (zásobníková abeceda)*

R *je konečná množina pravidel tvaru:* ${}_Apa \rightarrow wq$*,* $A \in \Gamma$*,* $p, q \in Q$*,* $a \in \sum \cup \{\epsilon\}$*,* $w \in \Gamma^*$

$s \in Q$ *je počáteční stav*

$S \in \Gamma$ *je počáteční zásobníkový symbol*

$F \subseteq Q$ *je množina všech koncových stavů*

Konfigurace: $\chi \in \Gamma^* Q \sum^*$
Přechod: *Nechť* $xApay$ *a* $xwqy$ *jsou konfigurace zásobníkového automatu* M*, kde platí:* $x, w \in \Gamma^*$*,* $A \in \Gamma$*,* $p, q \in Q$*,* $a \in \sum \cup \{\epsilon\}$ *a* $y \in \sum *$*. Nechť* $r = Apa \rightarrow wq \in R$ *je pravidlo. Potom zásobníkový automat* M *může provést krok z* $xApay$ *do* $xwqy$ *použitím pravidla* r *(*$xApay \vdash xwqy$*).* ***Přijímaný jazyk:*** M *je zásobníkový automat, který přijímá jazyky:*

- *Přechodem* M *do koncového stavu:* $L(M)_f = \{w : w \in \sum^*, Ssw \vdash^* zf, z \in \Gamma^*, f \in F\}$
- *Vyprázdněním zásobníku:* $L(M)_\epsilon = \{w : w \in \sum^*, Ssw \vdash^* zf, z = \epsilon, f \in Q\}$
- *Přechodem* M *do koncového stavu a zároveň vyprázdněním zásobníku:* $L(M)_\epsilon = \{w : w \in \sum^*, Ssw \vdash^* zf, z = \epsilon, f \in Q\}$

3.3 Hluboký zásobníkový automat

Hluboký zásobníkový automat (angl. *deep pushdown automata*) vychází ze zásobníkového automatu, ale na rozdíl od něho má možnost pracovat nejen s vrcholem zásobníku, ale i s neterminály v určité hloubce v zásobníku. Poprvé byl představen v [2] a následně také publikován v [3] Alexandrem Medunou.

Automat pracuje na stejném principu vyjmutí a expanze jako zásobníkový automat. Operace vyjmutí se neliší. Pokud na vstupní pásce je stejný terminál jako na vrcholu zásobníku provede se vyjmutí. Na rozdíl od zásobníkového automatu, který provádí expanzi pouze neterminálu na vrcholu, je možnost provádět expanzi i neterminálů, které jsou umístěny níže v zásobníku. Expanzi lze tedy provést nad n nejvrchnějšími neterminály, kde n je maximální hloubka. Se zvyšující se hloubkou roste hierarchicky síla tohoto automatu. Síla tohoto automatu je srovnatelná se sílou n-omezených stavových gramatik.

Definice 3.3.1.
Hluboký zásobníkový automat je definován jako uspořádaná sedmice ${}_{deep}M = (Q, \sum, \Gamma, R, s, S, F)$ *viz. [4], kde*

deep je maximální hloubka, ve které může být provedena expanze neterminálu

Q *je konečná neprázdná množina stavů*

$\sum$ *je množina vstupních symbolů (vstupní abeceda)*

Γ *je množina zásobníkových symbolů (zásobníková abeceda), kde* $\sum \subset \Gamma$, $\$ \in (\Gamma - \sum)$, *symbol* $ *označuje dno zásobníku*

R *je konečná množina pravidel tvaru:* ${}_{d}pN \rightarrow wq$, $0 < d <= deep$ *(d značí hloubku prováděné expanze),* $p, q \in Q$, $N \in (\Gamma - \sum)$, $w \in \Gamma*$

$s \in Q$ *je počáteční stav*

$S \in \Gamma$ *je počáteční zásobníkový symbol*

$F \subseteq Q$ *je množina všech koncových stavů*

Konfigurace: $\chi \in Q \sum^* (\Gamma - \{\#\})^* \{\#\}$
Přechod: *Hluboký zásobníkový automat* ${}_{deep}M$ *provede přechod* $x_p \Rightarrow y$, *když* $x = (q, au, az)$, $y = (q, u, z)$, *kde* $q \in Q$, $a \in \sum$, $u \in \sum^*$, $z \in \Gamma^*$. ${}_{deep}M$ *provede expanzi, pokud* $x = (q, w, uAz)$, $y = (p, w, uvz)$ *a* $(mqA \rightarrow pv) \in R$, *kde* $q, p \in Q$, $w \in \sum^*$, $A \in \Gamma$, $u, v, z \in \Gamma^*$. *Tedy* $x_e \Rightarrow y$.
Přijímaný jazyk: ${}_{deep}M$ *je hluboký zásobníkový automat, který přijímá jazyk:* $L({}_{deep}M) = \{w : w \in \sum^*, (s, w, S\#) \Rightarrow^* (f, \epsilon, \#)$ *v* ${}_{deep}M$, *s* $f \in F\}$. *Tento jazyk je přijímán koncovým automatem a prázdným stavem. Automat dokáže přijmout jazyk i když se nenachází v koncovém stavu:* $E({}_{deep}M) = \{w : w \in \sum^*, (s, w, S\#) \Rightarrow^* (f, \epsilon, \#)$ *v* ${}_{deep}M$, *s* $f \in Q\}$

Kapitola 4

Syntaktická analýza

Syntaktická analýza neboli syntaktický analyzátor(angl. *parser*) slouží ke kontrole vstupního řetězce tokenů. Hlavním úkolem je tedy rozhodnout zda vstupní řetězec patří do daného jazyka. Syntaktický analyzátor tedy staví ze vstupních tokenů tzv. **derivační strom**, který se snaží vyplnit prostor mezi startujícím symbolem a vstupním řetězcem. Syntaktickou analýzu dělíme podle způsobu hledání derivace věty na syntaktickou analýzu pracující **shora dolů** a **zdola nahoru**.

4.1 Syntaktická analýza shora dolů

Tato metoda postupuje podle levé derivace od kořenu derivačního stromu směrem dolů k listům. Kořen je reprezentován startujícím neterminálním symbolem, který je v postupných derivačních krocích expandován až na terminální symboly. Ty lze následně porovnat se vstupním řetězcem. Metoda pracuje na principu aplikování gramatických pravidel, která

Obrázek 4.1: Syntaktická analýza shora dolů

obsahují na levé straně příslušný neterminál. Problém nastává pokud více pravidel má shodnou levou stranu. Vybere se jedno pravidlo a pokud dojde k neshodě terminálů je nutný návrat a použití jiného pravidla jinak se pokračuje dále. Tento způsob je však značně neefektivní a tím i nepoužitelný u programovacích jazyků. Problém lze vyřešit použitím prediktivní syntaktické analýzy pracující s *LL(k) gramatikami*. Prediktivní syntaktická analýza je níže modifikována o hluboký zásobníkový automat, proto si přiblížíme dvě základní metody konstrukce analyzátorů.

4.1.1 Syntaktická analýza rekurzivním sestupem

Jak již naznačuje název tato metoda využívá rekurzi. Každý neterminál je reprezentován samostatnou funkcí. Tyto funkce jsou rekurzivně volány podle výskytu neterminálů v pravidlech. Využití metody je vhodné u jednoduchých gramatik, protože u složitých gramatik s větším počtem neterminálů by výsledkem byl rozsáhlý méně přehledný kód. Nevýhodou této metody je nutnost změny celého kódu při změně v gramatice nebo použití jiné gramatiky. Z hlediska časové složitosti může metoda dosáhnout až exponenciální časové složitosti.

4.1.2 Prediktivní, tabulkou řízená syntaktická analýza

Syntaktický analyzátor u této metody se skládá ze zásobníku symbolů, vstupní pásky a prediktivní tabulky. Tato metoda je v této práci modifikována, proto si ji přiblížíme. Analyzátor na základě symbolu na vrcholu zásobníku a aktuálního symbolu ze vstupní pásky provádí dvě základní operace: vyjmutí a expanzi pravidla. Analýza končí buď vyprázdněním zásobníku a vstupní pásky nebo při syntaktické chybě, kdy pro danou kombinaci symbolů neexistuje záznam v tabulce. Vyjmutí je provedeno pokud na vrcholu zásobníku je shodný terminální symbol jako je aktuální symbol ze vstupní pásky. Pokud se na vrcholu zásobníku nachází neterminální symbol je podle tabulky zjištěno příslušné pravidlo a provedena expanze podle tohoto pravidla. Tato metoda využívá jednoduchý algoritmus 1, který je imunní vůči změnám gramatiky. Zde vložený algoritmus v pseudokódu dodržuje stejnou syntaxi, která byla nastolena v [3].

Algoritmus 1: Prediktivní tabulkou řízená syntaktická analýzy

```
1  pop($) ; // vložení startujícího symbolu na zásobník
2  nextToken(a) ; // funkce vracející další terminál
3  repeat
4      switch X do
5          case X = $
6              if a = $ then SUCCESS ;
7              else ERROR ;
8          case X ∈ T
9              if X = a then
10                 pop(X);
11                 nextToken(a);
12             else ERROR ;
13         case X ∈ N
14             if X → x ∈ tabulka[X, a] then
15                 pop(X);
16                 push(reversal(x)) ; // obrácení pravé strany pravidla a
                   vložení na zásobník
17             else ERROR ;
18     endsw
19 until SUCCESS or ERROR;
```

Výstupem toho algoritmu je posloupnost pravidel, které symbolizují nejlevější derivaci pro vstupní řetězec neboli tzv. levý rozbor.

4.2 Syntaktická analýza zdola nahoru

Pro úplnost zde představím i opačně pracující metodu. Tato metoda na rozdíl od metody shora dolů postupuje směrem od vstupního řetězce ke kořenu (startující symbol). Analýza začíná tedy od jednotlivých terminálů, která jsou postupně převáděny na neterminály až na startující neterminál. Využívá se především pro analýzu matematických výrazů. Je silnější než metoda shora dolů.

Obrázek 4.2: Syntaktická analýza zdola nahoru

Kapitola 5

Modifikace prediktivní syntaktické analýzy

V této kapitole představím možnost využití hlubokého zásobníkového automatu v tabulkou řízené prediktivní analýze. U níže vytvořeného algoritmu si popíšeme jeho funkci krok po kroku.

5.1 Tabulkou řízená prediktivní syntaktická analýza s hlubokým zásobníkovým automatem

Rozšíření této metody spočívá v nahrazení klasického zásobníkového automatu za hluboký zásobníkový automat a tím posílení celé této analýzy. Analyzátor provádí vyjmutí a expanze. Expanze neterminálu je však díky hlubokému zásobníkovému automatu odlišná. Jak již vychází z definice tohoto automatu expanze je prováděna od nejhlubších neterminálů v zásobníku.

K tomuto úkolu je využito pomocného zásobníku, který vybere jednotlivé symboly ze zásobníku až do požadované maximální hloubky nebo dna zásobníku. Je tedy nutné modifikovat základní algoritmus tabulkou řízené prediktivní analýzy tak, aby získal tyto neterminály a podle prediktivní tabulky získal ve správném pořadí čísla pravidel pro expanzi jednotlivých neterminálů. Analýza končí vyprázdněním zásobníku a vstupní pásky.

5.1.1 Modifikovaný algoritmus

Modifikovaný algoritmus vychází ze základního algoritmu pro tabulkou řízenou prediktivní syntaktickou analýzu. Pro využití hlubokého zásobníkového automatu je nutné zavést několik nových proměnných a funkcí. Na rozdíl od zásobníkového automatu může pracovat hluboký zásobníkový automat v různé hloubce zanoření. Je tedy nutné v algoritmu pracovat s hodnotou udávají hloubku.
Podrobný popis jednotlivých proměnných a funkcí využitých v algoritmu:

- X - vrchol zásobníku
- X_p - vrchol pomocného zásobníku
- **a** - token/terminál ze vstupní pásky
- **deep** - maximální hloubka pro expanzi

- **d** - aktuální hloubka
- **push, pop** - funkce pro vložení/vyjmutí ze zásobníku
- **push_p, pop_p** - funkce pro vložení/vyjmutí z pomocného zásobníku

Algoritmus 2 pracuje stejně jako základní algoritmus 1 v cyklu, který určí zda daný vstupní řetězec patří do jazyka nebo ne. Řetězec patří do jazyka pokud je vyprázdněn zásobník a jsou přečteny všechny symboly ze vstupní pásky. V případě, že nastane neplatná kombinace symbolu ze zásobníku a vstupní pásky tak kontrolovaný řetězec nepatří do jazyka.

Algoritmus 2: Modifikovaný algoritmus prediktivní syntaktické analýzy

```
1  repeat
2      switch X do
3          case X = $
4              if a = $ then SUCCESS ;
5              else ERROR ;
6          case X ϵ T
7              if X = a then
8                  pop(X); nextToken(a);
9              else ERROR ;
10         case X ϵ N
11             repeat // cyklus na vyhledání neterminálů
12                 switch X do
13                     case X = $
14                         END;
15                     case X ϵ T
16                         push_p(X); pop(X);
17                     case X ϵ N
18                         d = d + 1; push_p(X); pop(X);
19                 endsw
20             until d == deep or END;
21             repeat // cyklus na expanzi neterminálů
22                 switch Xp do
23                     case X = $
24                         END
25                     case X ϵ T
26                         push_p(X); pop(X);
27                     case X ϵ N
28                         if Xp → x ϵ LL-table[Xp, a, d] then
29                             push(reversal(x)); pop_p(Xp); d = d − 1;
30                         else ERROR ;
31                 endsw
32             until d == 0 or END or ERROR;
33     endsw
34 until SUCCESS or ERROR;
```

Pokud se na vrcholu zásobníku nachází terminál, který odpovídá aktuálnímu terminálu ze vstupní pásky tak je vyjmut. Při výskytu neterminálu na vrcholu zásobníku je nutné na rozdíl od základního algoritmu, který expandoval pouze tento neterminál, nalézt a následně expandovat i neterminály do příslušné hloubky v zásobníku. Expanze neterminálů se provádí od nejhlubšího neterminálu až po neterminál na vrcholu zásobníku. Je tedy nutná změna pořadí neterminálů.

K vyhledání a následné změně pořadí neterminálů se využívá pomocný zásobník, do kterého jsou postupně vkládány symboly ze zásobníku. Pokud algoritmus narazí na terminál tak jej odstraní ze zásobníku a vloží jej na pomocný zásobník. Totéž provádí i s neterminály, kde navíc při každé této operaci zvyšuje aktuální hloubku. Tento cyklus končí po dosažení požadované hloubky nebo vyprázdněním zásobníku. Po cyklu jsou všechny neterminály, které mají být expandovány, umístěny na pomocném zásobníku. Dříve nejhlouběji umístěný neterminál je nyní na vrcholu pomocného zásobníku.

V druhém cyklu je tedy provedena postupná expanze všech neterminálů z pomocného zásobníku zpět na zásobník. Pokud narazí na terminál tak jej zkopíruje zpět na zásobník, ale v případě neterminálu se provede expanze podle pravidla. Jaké pravidlo má být použito se určí pomocí prediktivní tabulky, ve které se pomocí expandovaného neterminálu, aktuálního terminálu ze vstupní pásky a hloubky daného neterminálu určí číslo pravidla nebo při chybné struktuře vstupního řetězce syntaktická chyba.

Ukázka dotazu do prediktivní tabulky:

$$pravidlo = tabulka[X_p, a, d] \tag{5.1}$$

Pravá strana vybraného pravidla je obrácena a vložena na zásobník. Expandovaný neterminál je odstraněn z pomocného zásobníku a je snížena aktuální hloubka. Cyklus takto expanduje všechny neterminály na pomocném zásobníku a končí vyprázdněním pomocného zásobníku.

5.1.2 Modifikovaná LL gramatika

Výše uvedený algoritmus využívá LL gramatiky, u který na rozdíl od běžných LL gramatik obsahují jednotlivá pravidla hloubku, která je klíčová pro použití pravidla. Hodnota určuje, ve které hloubce je možné pravidlo použít pro expanzi. Hloubka je znázorněna indexem na začátku každého pravidla, kde hloubka 1 označuje neterminál na vrcholu zásobníku.

Ukázka pravidel pro jazyk $L = \{wcw | w \in \{a, b\}^*\}$:

1. ${}_1S \rightarrow XcY$
2. ${}_1X \rightarrow aX$
3. ${}_1X \rightarrow bX$
4. ${}_2Y \rightarrow aY$
5. ${}_2Y \rightarrow bY$
6. ${}_1X \rightarrow \epsilon$
7. ${}_1X \rightarrow \epsilon$

Pro vybrání pravidla je tedy nutné znát nejen terminál ze vstupní pásky a neterminál ze zásobníku, ale i hloubku, ve které se tento neterminál nachází. Kvůli přidané hloubce, jakož to dalším parametru pro vybrání pravidla je potřeba rozšířit LL tabulku o další rozměr. Výsledná tabulka je tak trojrozměrná. Pro zobrazení tabulky ve dvourozměrném prostoru je hodnota hloubky zobrazena v první buňce tabulky (levý horní roh tabulky).

Ukázka LL tabulky pro jazyk $L = \{wcw|w \in \{a, b\}^*\}$:

Hloubka 1:

1	a	b	c	$
X	2	3	6	
Y				
S	1	1	1	
$				☺

Hloubka 2:

2	a	b	c	$
X				
Y	4	5	7	
S				
$				

Tato gramatika pracuje do hloubky dvě. To znamená, že provádí expanzi až dvou nejvrchnějších neterminálů na zásobníku.

Kapitola 6

Demonstrace funkce modifikovaného algoritmu

Pro demonstraci funkce modifikovaného algoritmu si v této kapitole ukážeme přijmutí dvou kontextových jazyků, které by nebylo možné přijímat základním algoritmem se zásobníkovým automatem. Pro každý jazyk je zvolený řetězec. Přijetí tohoto řetězce je znázorněno v jednotlivých krocích, které zobrazují stavy zásobníků, vyhledání a expanzi pravidel, vyjmutí terminálních symbolů a přesun symbolů mezi zásobníky. V tomto zobrazení jednotlivých kroků analýzy vstupního řetězce označuje symbol „$“ dno zásobníku nebo konec vstupní pásky. Dotazy do LL tabulky jsou na rozdíl od klasické analýzy rozšířeny o třetí údaj, který reprezentuje aktuální hloubku expandovaného neterminálu ze zásobníku. Pro oba jazyky byly vytvořeny příslušné gramatiky pracující s danou hloubkou.

6.1 Jazyk $L = \{a^n b^n c^n | n \geq 1\}$

Tento jazyk není bezkontextový, toto tvrzení je dokázáno pomocí důkazu. Slova tohoto jazyka se skládají ze symbolů „a, b, c“, které mají dané přesné pořadí. Všechny znaky se musí vyskytovat ve stejném počtu, tedy pokud slovo začíná třemi symboly „a“ musí dále obsahovat tři symboly „b“ a následně i „c“. Hodnota n tedy musí být větší než jedna.

6.1.1 Důkaz

Lemma:
Jazyk $L = \{a^n b^n c^n | n \geq 1\}$ není bezkontextový.
Důkaz:
Vyházíme z obecného lemma o vkládání pro bezkontextové jazyky. Zvolíme tedy $t = a^n b^n c^n$, kde n je přirozené a libovolné číslo. Při využití obecného tvaru rozdělení $t = uv^i xy^i z$, kde $|vxy| \leq n$ a $vy \neq \epsilon$ mohou nastat pro umístění vy tyto případy:

1. Slovo vy bude umístěno mezi symboly a, pak pro $i = 0$ se změní počet a vůči b
2. Slovo vy bude umístěno na hranici mezi symboly a, b, pak pro $i = 0$ se změní počet a, b vůči c
3. Slovo vy bude umístěno mezi symboly b, pak pro $i = 0$ se změní počet b vůči c
4. Slovo vy bude umístěno na hranici mezi symboly b, c, pak pro $i = 0$ se změní počet b, c vůči a

5. Slovo vy bude umístěno mezi symboly a, pak pro $i = 0$ se změní počet c vůči a

Výsledek:
Ve všech těchto možnostech nastalo $uv^ixy^iz \notin L$, a proto tento jazyk není bezkontextový.

6.1.2 Gramatika

Gramatika pro tento jazyk provádí také expanzi maximálně prvních dvou neterminálů na zásobníku. Obsahuje 3 přepisovací pravidla a 2 tzv. vymazávací pravidla neboli ϵ-pravidla.

Pravidla:

1. ${}_1S \rightarrow AB$
2. ${}_1A \rightarrow aAb$
3. ${}_2B \rightarrow Bc$
4. ${}_1A \rightarrow \epsilon$
5. ${}_2B \rightarrow \epsilon$

LL tabulka:

1	a	b	c	$
A	2	4		
B				
S	1			
$				☺

2	a	b	c	$
A				
B	3	5		
S				
$				

6.1.3 Přijmutí jazyka

Ukázka přijmutí řetězce „aabbcc“, který patří do toho jazyka:

pomocný zásobník	zásobník	vstupní páska	LL tabulka	akce/pravidlo
$	$$S$	$aabbcc$$		
$$S$	$	$aabbcc$$	$[S, a, 1] \Rightarrow$	1: ${}_1S \rightarrow AB$
$	$$BA$	$aabbcc$$		
$$AB$	$	$aabbcc$$	$[B, a, 2] \Rightarrow$	3: ${}_2B \rightarrow Bc$
$$A$	$$cB$	$aabbcc$$	$[A, a, 1] \Rightarrow$	2: ${}_1A \rightarrow aAb$
$	$$cBbAa$	$aabbcc$$		vyjmutí „a“
$	$$cBbA$	$abbcc$$		
$$AbB$	$$c$	$abbcc$$	$[B, a, 2] \Rightarrow$	3: ${}_2B \rightarrow Bc$
$$Ab$	$$ccB$	$abbcc$$	$[A, a, 1] \Rightarrow$	2: ${}_1A \rightarrow aAb$
$	$$ccBbbAa$	$abbcc$$		vyjmutí „a“
$	$$ccBbbA$	$bbcc$$		
$$AbbB$	$$cc$	$bbcc$$	$[B, b, 2] \Rightarrow$	5: ${}_2B \rightarrow \epsilon$
$$Abb$	$$cc$	$bbcc$$	$[A, b, 1] \Rightarrow$	4: ${}_1A \rightarrow \epsilon$
$	$$ccbb$	$bbcc$$		vyjmutí „b“
$	$$ccb$	$bcc$$		vyjmutí „b“
$	$$cc$	$cc$$		vyjmutí „c“
$	$$c$	$c$$		vyjmutí „c“
$	$	$$		ÚSPĚCH

Zvolený řetězec patří do tohoto jazyka ($aabbcc \in \{a^nb^nc^n | n \geq 1\}$). Celkem pro přijetí bylo provedeno 7 expanzí podle pravidel a 6 vyjmutí terminálů z vrcholu zásobníku.

6.2 Jazyk $L = \{wcw|w \in \{a,b\}^*\}$

Na základě důkazu níže, není tento jazyk bezkontextový. Jedná se tedy o dvě shodná slova („w“) skládají se ze symbolů „a, b“ v libovolném pořadí. Tyto slova jsou oddělena symbolem „c“, který označuje právě střed všech slov patřících do tohoto jazyka.

6.2.1 Důkaz

Lemma: Jazyk $L = \{wcw|w \in \{a,b\}^*\}$ není bezkontextový.
Důkaz:
Vycházíme z obecného lemma o vkládání pro bezkontextové jazyky. Zvolíme tedy $t = a^n b^n c a^n b^n$, kde n je přirozené a libovolné číslo, dále tedy platí $t \in L$ a $|t| > n$. Při využití obecného tvaru rozdělení $t = uv^i xy^i z$, kde $|vxy| \leq n$ a $vy \neq \epsilon$ mohou nastat pro tvar vy tyto případy:

1. Pokud vy bude v první polovině, při zvolení $i = 0$ a odebraním některého ze symbolů a nebo b z této poloviny slova narušíme počet symbolů v obou polovinách. Symbol c přestane zobrazovat polovinu a tím tedy nebude patři do jazyka L.

2. V případě, že vy se bude vyskytovat v přes střed slova existují dvě možnosti:

   - Pokud je c součástí vy, pak pro $i = 0$ dostáváme slovo bez c, které tedy nepatří do jazyka L.
   - Platí tedy při $i = 0$ odebereme-li nějaké b z první poloviny slova nebo nějaké a z druhé poloviny tak slovo nebude patřit do jazyka L, ale nikdy nesmí nastat stav kdy odebereme buď a nebo b zároveň z obou částí slova, protože $|vwx| \leq n|$ a slovo obsahuje ještě $n + 1$ dalších symbolů.

3. Pokud vy bude ve druhé polovině slova a zvolíme-li $i = 0$ bude druhá část slova kratší a tedy slovo nebude patřit do L

Výsledek: Ve všech případech nastalo $uv^i xy^i z \notin L$ z toho tedy plyne, že tento jazyk není bezkontextový.

6.2.2 Gramatika

Gramatika pro tento jazyk provádí expanzi maximálně prvních dvou neterminálů na zásobníku. Obsahuje 5 přepisovacích pravidel a 2 tzv. vymazávací pravidla neboli ϵ-pravidla.

Pravidla:

1. ${}_1S \rightarrow XcY$
2. ${}_1X \rightarrow aX$
3. ${}_1X \rightarrow bX$
4. ${}_2Y \rightarrow aY$
5. ${}_2Y \rightarrow bY$
6. ${}_1X \rightarrow \epsilon$
7. ${}_2Y \rightarrow \epsilon$

LL tabulka:

1	a	b	c	$
X	2	3	6	
Y				
S	1	1	1	
$				☺

2	a	b	c	$
X				
Y	4	5	7	
S				
$				

6.2.3 Přijmutí jazyka

Ukázka přijmutí řetězce „abbacabba“, který patří do toho jazyka:

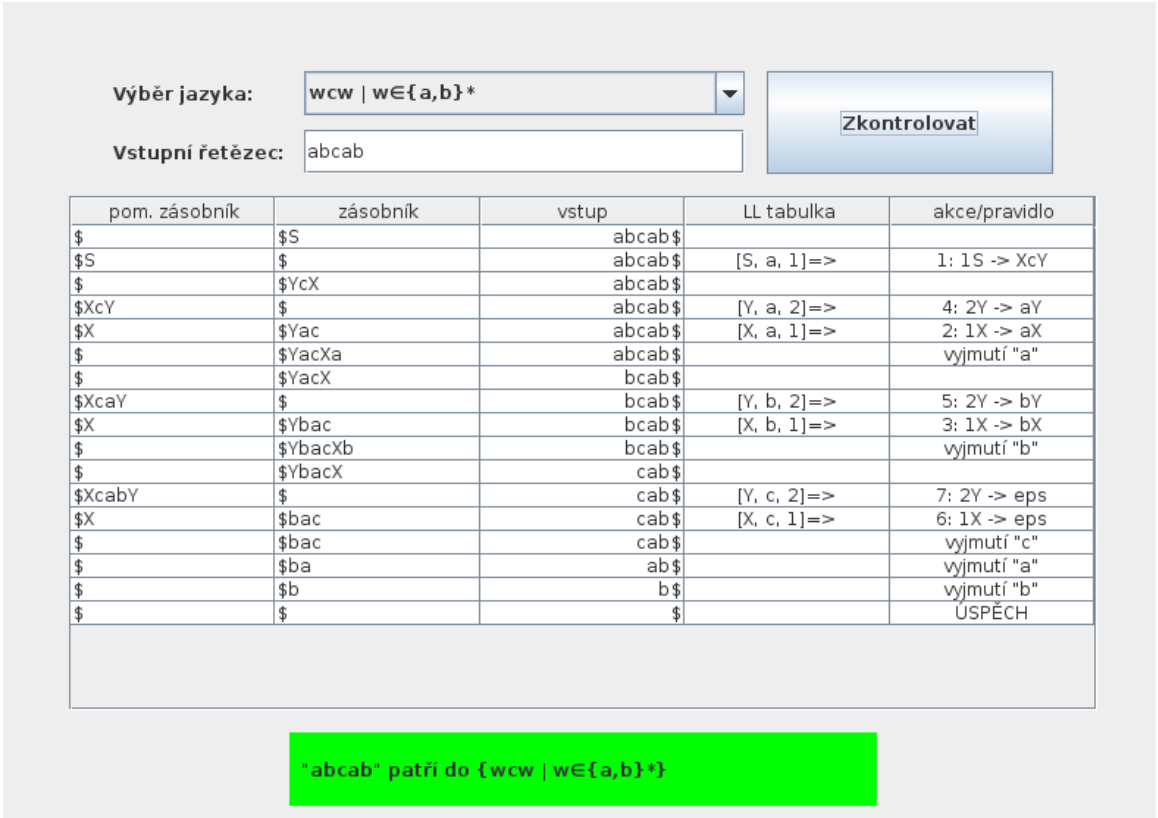
pomocný zásobník	zásobník	vstupní páska	LL tabulka	akce/pravidlo
$\$$	$\$S$	$abbacabba\$$		
$\$S$	$\$$	$abbacabba\$$	$[S, a, 1] \Rightarrow$	1: ${}_1S \rightarrow XcY$
$\$$	$\$YcX$	$abbacabba\$$		
$\$XcY$	$\$$	$abbacabba\$$	$[Y, a, 2] \Rightarrow$	4: ${}_2Y \rightarrow aY$
$\$Xc$	$\$Ya$	$abbacabba\$$	$[X, a, 1] \Rightarrow$	2: ${}_1X \rightarrow aX$
$\$$	$\$YacXa$	$abbacabba\$$		vyjmutí „a“
$\$$	$\$YacX$	$bbacabba\$$		
$\$XcaY$	$\$$	$bbacabba\$$	$[Y, b, 2] \Rightarrow$	5: ${}_2Y \rightarrow bY$
$\$Xca$	$\$Yb$	$bbacabba\$$	$[X, b, 1] \Rightarrow$	3: ${}_1X \rightarrow bX$
$\$$	$\$YbacXb$	$bbacabba\$$		vyjmutí „b“
$\$$	$\$YbacX$	$bacabba\$$		
$\$XcabY$	$\$$	$bacabba\$$	$[Y, b, 2] \Rightarrow$	5: ${}_2Y \rightarrow bY$
$\$Xcab$	$\$Yb$	$bacabba\$$	$[X, b, 1] \Rightarrow$	3: ${}_1X \rightarrow bX$
$\$$	$\$YbbacXb$	$bacabba\$$		vyjmutí „b“
$\$$	$\$YbbacX$	$acabba\$$		
$\$XcabbY$	$\$$	$acabba\$$	$[Y, a, 2] \Rightarrow$	4: ${}_2Y \rightarrow aY$
$\$Xcabb$	$\$Ya$	$acabba\$$	$[X, a, 1] \Rightarrow$	2: ${}_1X \rightarrow aX$
$\$$	$\$YabbacXa$	$acabba\$$		vyjmutí „a“
$\$$	$\$YabbacX$	$cabba\$$		
$\$XcabbaY$	$\$$	$cabba\$$	$[Y, c, 2] \Rightarrow$	7: ${}_2Y \rightarrow \epsilon$
$\$Xcabba$	$\$$	$cabba\$$	$[X, c, 1] \Rightarrow$	6: ${}_1X \rightarrow \epsilon$
$\$$	$\$abbac$	$cabba\$$		vyjmutí „c“
$\$$	$\$abba$	$abba\$$		vyjmutí „a“
$\$$	$\$abb$	$bba\$$		vyjmutí „b“
$\$$	$\$ab$	$ba\$$		vyjmutí „b“
$\$$	$\$a$	$a\$$		vyjmutí „a“
$\$$	$\$$	$\$$		ÚSPĚCH

Zvolený řetězec patří do tohoto jazyka ($abbacabba \in \{wcw | w \in \{a, b\}^*\}$). Celkem pro přijetí bylo provedeno 11 expanzí podle pravidel a 9 vyjmutí terminálů z vrcholu zásobníku.

Kapitola 7

Implementace

Pro ukázku a testování funkčnosti navrženého algoritmu byla vytvořena aplikace s uživatelským rozhraním v programovacím jazyce Java. Na tvorbu jednoduchého uživatelského rozhraní byla využita knihovna Swing. Aplikace je navržena jako jednoduchý nástroj pro demonstraci funkce algoritmu. Umožňuje výběr mezi jazyky, které byli představeny v kapitole 6. Dále je možnost přidat libovolný další jazyk respektive jeho gramatiku, která splňuje podmínky modifikované LL gramatiky (viz 5.1.2). Ukázka vzhledu aplikace je vyobrazena na obrázcích 7.1 a 7.2. Aplikace tedy zkontroluje zda zadaný řetězec patří do jazyka a zobrazí jednotlivé kroky kontroly. U každého kroku jsou zobrazeny stavy zásobníků, symboly na vstupní pásce a použitá pravidla nebo provedené akce.



pom. zásobník	zásobník	vstup	LL tabulka	akce/pravidlo
$	$S	abcab$		
$S	$	abcab$	[S, a, 1]=>	1: 1S -> XcY
$	$YcX	abcab$		
$XcY	$	abcab$	[Y, a, 2]=>	4: 2Y -> aY
$X	$Yac	abcab$	[X, a, 1]=>	2: 1X -> aX
$	$YacXa	abcab$		vyjmutí "a"
$	$YacX	bcab$		
$XcaY	$	bcab$	[Y, b, 2]=>	5: 2Y -> bY
$X	$Ybac	bcab$	[X, b, 1]=>	3: 1X -> bX
$	$YbacXb	bcab$		vyjmutí "b"
$	$YbacX	cab$		
$XcabY	$	cab$	[Y, c, 2]=>	7: 2Y -> eps
$X	$bac	cab$	[X, c, 1]=>	6: 1X -> eps
$	$bac	cab$		vyjmutí "c"
$	$ba	ab$		vyjmutí "a"
$	$b	b$		vyjmutí "b"
$	$	$		ÚSPĚCH

Obrázek 7.1: Ukázka přijmutí řetězce (slova)

7.1 Struktura a popis aplikace

Aplikace se skládá z několika tříd, kde každá reprezentuje určitý funkční celek. Po spuštění kontroly je načtena vybraná gramatika i s pravidly a provede se kontrola vstupního řetězce pomocí analyzátoru, který implementuje funkci modifikovaného algoritmu. Třídy se dělí na třídy reprezentující grafické uživatelské rozhraní a na třídy reprezentující samotný analyzátor.

7.1.1 Třídy reprezentující uživatelské rozhraní

GUI a MainLabel

Třídy zobrazují okno aplikace a starají se o interakci s uživatelem. Po spuštění aplikace jsou ve třídě `GUI` nastaveny parametry zobrazovaného okna. Třída `MainLabel` zobrazuje jednotlivé funkční a informační elementy v okně. Po spuštění kontroly uživatelem pak volá metody pro načtení vybrané gramatiky, nastavení zadaného řetězce a následné zahájení kontroly. Po proběhnutí této kontroly, zobrazí v tabulce jednotlivé kroky kontroly a následně i rozhodnutí zda zadaný řetězec patří do vybraného jazyka nebo ne. Pokud řetězec nepatří do jazyka je v tabulce jako poslední zobrazen krok, ve kterém se vyskytla chyba.

7.1.2 Třídy reprezentující analyzátor

Parser

Jedná se o srdce celé aplikace, kde je prováděna kontrola vstupního řetězce pomocí implementovaného algoritmu prediktivní syntaktické analýzy s hlubokým zásobníkem. Jsou zde vytvořeny všechny potřebné struktury pro analýzu. Třída obsahuje dva zásobníky, kde zásobník `mainStack` reprezentuje hlavní zásobník pracující do hloubky a zásobník `tmpStack` je pomocný zásobník. Dále je zde uchována gramatika jazyka, který byl vybrán v podobě objektu třídy reprezentující gramatiku.

Jednotlivé kroky analýzy jsou uloženy jako řádky tabulky v listu (`outputRows`). Modifikovaný algoritmus tabulkou řízené prediktivní syntaktické s hlubokým zásobníkem je implementovaný v metodě (`predictiveParsing()`), která provede kontrolu vstupního řetězce. Další metody slouží pro nastavování výše zmíněných struktur.

InputTape

Jedná se o reprezentaci vstupní pásky, která obsahuje vstupní řetězec. Pomocí metody `getNextToken()` čte postupně zleva doprava symboly z pásky a předává je k analýze.

Grammar

Tato třída reprezentuje modifikovanou LL gramatiku jazyka. Uchovává jednotlivá pravidla, maximální hloubku pro expanzi neterminálů na zásobníku a LL tabulku. Gramatiky jsou načítány ze souboru, který má přesně daný formát (viz 7.2). Pro načtení gramatiky je vytvořena metoda `loadGrammar()`, která nejprve načte z tohoto souboru důležité informace o gramatice, její pravidla a LL tabulku. Dále tato třída obsahuje metody pro práci s pravidly, dotazy do LL tabulky a metodu pro získání pravé strany pravidla při expanzi podle čísla pravidla.

LLtable

V této třídě se nachází LL tabulka a metody pro práci s ní. Tabulka je uložena ve třírozměrném poli, kde třetí rozměr reprezentuje hodnotu hloubky. Načítání tabulky je provedeno stejně jako u pravidel ze souboru s gramatikou po jednotlivých řádcích. Základní funkce (metoda `getIdRule()`) tabulky je tedy zjistit zda pro kombinaci neterminálu ze zásobníku, terminálu ze vstupní pásky a aktuální hloubky existuje pravidlo a pokud ano, tak vrátit index tohoto pravidla.

Rule

Třída reprezentující jednotlivá pravidla gramatiky. Uchovává index pravidla, který je důležitý při výběru pravidla pomocí LL tabulky, dále hloubku pro expanzi pravidla a klasicky levou a pravou stranu pravidla.

7.2 Struktura souboru s gramatikou jazyka

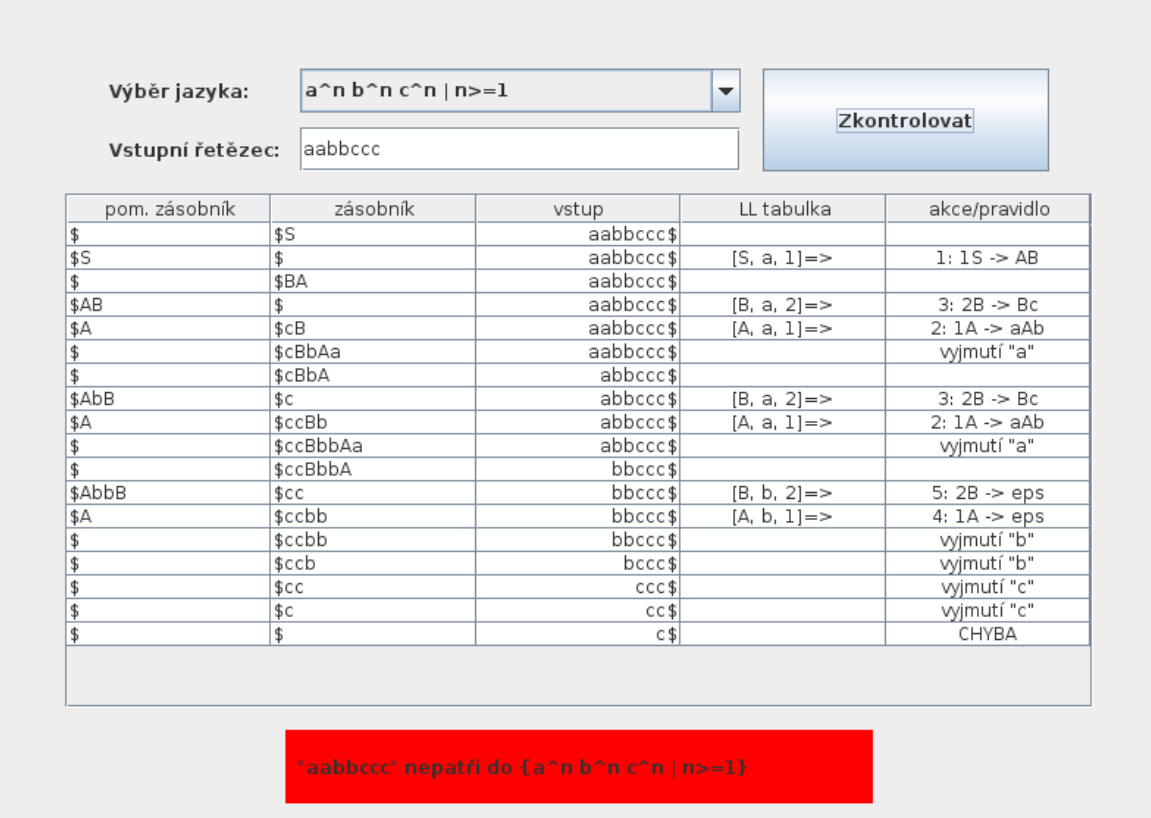
Jednotlivé soubory jsou pojmenovány podle jazyka, pro který je gramatika vytvořena. Na prvním řádku je název jazyka, který bude zobrazován v aplikaci. Na druhém řádku jsou zobrazeny tři hodnoty, kde první udává maximální hloubku expanze a následující hodnoty udávají počet řádků a sloupců LL tabulky. Následují řádky s pravidly (začínající prefixem *R:*) a po nich struktura LL tabulky (prefix *LL:*). Jednotlivá pravidla jsou zapsána pomocí čtyř hodnot udávajících index pravidla, hloubku, levou a pravou stranu pravidla.

Ukázka souboru s gramatikou

```
1  {wcw|w ∈ {a, b}∗}
2  2 5 5
3  R: 1 1 S XcY
4  R: 2 1 X aX
5  R: 3 1 X bX
6  R: 4 2 Y aY
7  R: 5 2 Y bY
8  R: 6 1 X eps
9  R: 7 2 Y eps
10 LL: 1 a b c $
11 LL: X 2 3 6 0
12 LL: Y 0 0 0 0
13 LL: S 1 1 1 0
14 LL: $ 0 0 0 -1
15 LL: 2 a b c $
16 LL: X 0 0 0 0
17 LL: Y 4 5 7 0
18 LL: S 0 0 0 0
19 LL: $ 0 0 0 0
```

7.3 Spuštění a ovládání aplikace

Aplikaci lze spustit z přiloženého a již vytvořeného JAR archivu nebo pomocí nástroje ANT pro kompilaci a spuštění (metoda `run`). Aplikace vyžaduje přiložené soubory s jednotlivými gramatikami. Po spuštění je tedy nutné zvolit jazyk respektive gramatiku a zadat vstupní řetězec. Po potvrzení bude tato gramatika načtena z přiloženého souboru a provedena kontrola vstupního řetězce (slova). Po kontrole je zobrazen výsledek a je možné zadat nový řetězec nebo vybrat jiný jazyk.



pom. zásobník	zásobník	vstup	LL tabulka	akce/pravidlo
$	$S	aabbccc$		
$S	$	aabbccc$	[S, a, 1]=>	1: 1S -> AB
$	$BA	aabbccc$		
$AB	$	aabbccc$	[B, a, 2]=>	3: 2B -> Bc
$A	$cB	aabbccc$	[A, a, 1]=>	2: 1A -> aAb
$	$cBbAa	aabbccc$		vyjmutí "a"
$	$cBbA	abbccc$		
$AbB	$c	abbccc$	[B, a, 2]=>	3: 2B -> Bc
$A	$ccBb	abbccc$	[A, a, 1]=>	2: 1A -> aAb
$	$ccBbbAa	abbccc$		vyjmutí "a"
$	$ccBbbA	bbccc$		
$AbbB	$cc	bbccc$	[B, b, 2]=>	5: 2B -> eps
$A	$ccbb	bbccc$	[A, b, 1]=>	4: 1A -> eps
$	$ccbb	bbccc$		vyjmutí "b"
$	$ccb	bccc$		vyjmutí "b"
$	$cc	ccc$		vyjmutí "c"
$	$c	cc$		vyjmutí "c"
$	$	c$		CHYBA

Obrázek 7.2: Ukázka nepřijmutí řetězce (slova)

Kapitola 8

Závěr

Tato práce se zabývala možným využitím hlubokého zásobníkového automatu v syntaktické analýze. Cílem bylo nahradit stávající zásobníkový automat v algoritmu pro prediktivní syntaktickou analýzu za hluboký zásobníkový automat, a tím získat větší vyjadřovací sílu tohoto algoritmu. Síla modifikovaného algoritmu s hlubokým zásobníkovým automatem roste se zvyšující se využívanou hloubkou pro expanzi neterminálních symbolů a je tedy ekvivalentní k n-omezeným stavovým gramatikám.

Při návrhu modifikovaného algoritmu bylo potřeba vyřešit několik problémů, které vznikly využitím hlubokého zásobníkového automatu. Jedním z těchto problémů byla tzv. expanze neterminálu v hloubce zásobníku a dodržení správného pořadí od nejhlouběji umístěného neterminálu po neterminál na vrcholu zásobníku. Tedy expanze neterminálního symbolu podle vybraného pravidla, nenacházejícího se na vrcholu zásobníku. Algoritmus byl modifikován tak, aby neterminální symboly našel v hloubce i když budou obklopeny ostatními symboly (terminální symboly) a provedl jejich expanzi. S tímto problémem souvisí i nutnost znát aktuální hloubku expandovaného neterminálního symbolu, protože chybná hodnota hloubky by mohla vést k chybně provedené syntaktické analýze.

Řešením je využití pomocného zásobníku k nalezení neterminálu v maximální možné hloubce, která je specifikována použitou gramatikou. Symboly jsou přesouvány postupně ze zásobníku do pomocného zásobníku. Přesouvání symbolů do pomocného zásobníku končí buď dosažením požadované hloubky (počet přesunutých neterminálních symbolů) nebo vyprázdněním zásobníku. Díky této operaci bude na vrcholu pomocného zásobníku dříve nejhlouběji umístěný neterminální symbol. Po té může být provedena postupná expanze neterminálů z pomocného zásobníku zpět na výchozí zásobník.

Vznikl na pochopení jednoduchý a přehledný algoritmus, který reprezentuje možné využitím hlubokého zásobníkového automatu při syntaktické analýze.

Při využití hlubokého zásobníkového automatu bylo nezbytné rozšířit obyčejné LL gramatiky o možnost pracovat s hodnotou hloubky zanoření v zásobníku. Nová gramatika obsahuje pravidla s údajem o hloubce. Hodnota u každého pravidla určuje hloubku, ve které může být využito k expanzi. Na základě takto modifikovaných pravidel vznikla i LL tabulka, která na rozdíl od obyčejné LL tabulky obsahuje navíc třetí rozměr reprezentující hloubku.

Obtížnost tvorby těchto gramatik roste s náročností jazyka a se zvyšující se maximální využívanou hloubkou. V navazující práci by bylo zajímavé se zamyslet nad možností automatické konstrukce LL tabulky na základě modifikovaných pravidel s hloubkou.

Pro demonstraci funkce modifikovaného algoritmu byla vytvořena jednoduchá aplikace, která pomocí tohoto algoritmu provádí prediktivní syntaktickou analýzu pro daný vstupní řetězec. Aplikace umožňuje výběr mezi dvěma jazyky (jejich gramatikami) a zároveň umožňuje i vložení další gramatiky při dodržení konvencí (viz sekce 7.2). Aplikace obsahuje jednoduché uživatelské rozhraní pro ovládání a zobrazování výsledků analýzy, je proto vhodná jako podpora při výuce.

Po dohodě s vedoucím práce nebyla analyzována složitost a náročnost nově vytvořeného algoritmu. Tato analýza by mohla být předmětem další navazující práce.

Literatura

[1] Chomsky, N.: Three models for the description of language. *IRE Transactions on Information Theory*, ročník 2, č. 3, 1956: s. 113–124.
URL http://dblp.uni-trier.de/db/journals/tit/tit2n.html#Chomsky56

[2] Meduna, A.: Deep Pushdown Automata. *Acta Informatica*, ročník 2006, č. 98, 2006: s. 114–124, ISSN 0001-5903.
URL http://www.fit.vutbr.cz/research/view_pub.php.en?id=8000

[3] Meduna, A.: *Formal Languages and Computation*. Taylor and Francis, Taylor & Francis Informa plc, 2014, ISBN 978-1-4665-1345-7, 315 s.
URL http://www.fit.vutbr.cz/research/view_pub.php.en?id=10524

[4] Meduna, A.; Zemek, P.: *Regulated Grammars and Automata*. Springer US, 2014, ISBN 978-1-4939-0368-9, 694 s.
URL http://www.fit.vutbr.cz/research/view_pub.php?id=10498

Dodatek A

Obsah CD

- **Adresář** `aplikace` obsahuje spustitelnou již zkompilovanou aplikaci do archivu `JAR` a soubory s gramatikami, které jsou potřebné pro analýzu.
- **Adresář** `latex_src` obsahuje zdrojové soubory textové části pro LaTeX.
- **Soubor** `BP_Zdenek_Krestan_2015.pdf` je elektronická verze textové části bakalářské práce.
- **Adresář** `source` obsahuje zdrojové soubory aplikace a skript pro kompilaci aplikace.
- **Soubor** `README.txt` obsahuje stručný popis obsahu a návod na použití aplikace.