Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezkontextové gramatiky s generovaním reverzov

Diplomová práca

2020
Bc. Alena Poláchová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Bezkontextové gramatiky s generovaním reverzov

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra informatiky
Školiteľ:	RNDr. Peter Kostolányi, PhD.

Bratislava, 2020
Bc. Alena Poláchová

61351357

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Alena Poláchová
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Bezkontextové gramatiky s generovaním reverzov
Reversal-Generating Context-Free Grammars

Anotácia: Práca sa bude venovať vybraným doposiaľ nepreskúmaným vlastnostiam bezkontextových gramatík s generovaním reverzov, najmä so zreteľom na možnosti efektívnej syntaktickej analýzy.

Cieľ: Preskúmať vybrané vlastnosti bezkontextových gramatík s generovaním reverzov. Zamerať sa predovšetkým na efektívne metódy syntaktickej analýzy.

Vedúci: RNDr. Peter Kostolányi, PhD.
Katedra: FMFI.KI - Katedra informatiky
Vedúci katedry: prof. RNDr. Martin Škoviera, PhD.

Dátum zadania: 23.11.2017

Dátum schválenia: 06.12.2017 prof. RNDr. Rastislav Kráľovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie:

Ďakujem svojmu školiteľovi RNDr. Petrovi Kostolányimu, PhD. za odborné vedenie, cenné rady a pripomienky ako aj za ochotu a trpezlivosť, s ktorou ma sprevádzal po celý čas písania diplomovej práce.

Abstrakt

Práca sa zaoberá modelom bezkontextových gramatík rozšíreným o generovanie špeciálnych terminálnych symbolov ®, kde každý takýto symbol v terminálnom slove otočí zvyšok slova za týmto symbolom. Už skôr bolo dokázané, že v prípade obmedzenia počtu generovaných reverzov na konštantu je takýto model ekvivalentný automatom s otáčacím zásobníkom. V tejto práci sa zaoberáme nedeterministickou schémou syntaktickej analýzy zdola nahor pre jazyky generované takýmito gramatikami. Prezentujeme obmenu nedeterministickej verzie štandardného algoritmu „posuň, redukuj“ pre tieto gramatiky, ktorú možno implementovať na zásobníkovom automate s otáčacím zásobníkom.

Kľúčové slová: bezkontextové gramatiky s generovaním reverzov, automaty s otáčacím zásobníkom, syntaktická analýza zdola nahor, algoritmus „posuň, redukuj, otoč“

Abstract

The thesis deals with a model of context-free grammars extended by a possibility of generating special terminal symbols ®, where each such symbol is interpreted as a reversal of the remaining input. It was already proved that this model is equivalent to flip-pushdown automata in case the number of generated reversals is limited by a constant. We describe a nondeterministic schema of bottom-up syntax analysis for languages generated by such grammars. In particular, we present an extension of the well-known nondeterministic shift-reduce algorithm for these grammars, which can be implemented using a flip-pushdown automaton.

Keywords: reversal-generating context-free grammar, flip-pushdown automaton, bottom-up syntax analysis, *shift-reduce-flip* algorithm

Obsah

Úvod

Jedným zo štandardných výsledkov klasickej teórie formálnych jazykov a automatov je ekvivalencia bezkontextových gramatík a zásobníkových automatov [4], pričom súvis medzi týmito dvoma modelmi nachádza praktické uplatnenie napríklad v oblasti syntaktickej analýzy. Snaha aplikovať známe postupy založené na bezkontextových gramatikách a zásobníkových automatoch aj na iné ako bezkontextové jazyky viedla viacerých autorov k skúmaniu rôznych rozšírení týchto modelov.

Sarkar [8] prišiel s modelom automatov s otáčacím zásobníkom a dokázal, že pri neobmedzenom počte otočení je rovnako silný ako Turingove stroje. Holzer a Kutrib [2, 3] o automatoch s otáčacím zásobníkom dokázali, že pri obmedzení počtu reverzov na konštantu sa so zvyšujúcim počtom povolených otočení zvyšuje sila automatu. Podobné tvrdenie o hierarchii vzhľadom na počet otočení dokázali aj o deterministickom variante automatov s otáčacím zásobníkom. Tiež vyslovili a dokázali tvrdenie, že nedeterministický variant je silnejší ako deterministický.

V článku [6] boli definované *bezkontextové gramatiky s generovaním reverzov*. Ide o rozšírenie bezkontextových gramatík, pri ktorom má gramatika možnosť vygenerovať niekoľko špeciálnych terminálnych symbolov reverzu. Symboly reverzu v terminálnom slove vyhodnotíme zľava doprava tak, že pre každý symbol reverzu otočíme zvyšok slova za týmto reverzom a samotný symbol pre reverz zmažeme. Výsledné slovo získané po vyhodnotení všetkých symbolov pre reverzy potom patrí do jazyka generovaného gramatikou. Keď obmedzíme počet vygenerovaných reverzov na konštantu $k \in \mathbb{N}$, sú gramatiky s generovaním práve k reverzov rovnako silné ako nedeterministické automaty s otáčacím zásobníkom a s k otočeniami zásobníka [6]. Gramatiky s generovaním reverzov umožnili dokázať niektoré nové výsledky o triede jazykov akceptovaných automatmi s otáčacím zásobníkom [6, 7].

Syntaktickou analýzou jazykov generovaných rôznymi rozšíreniami klasických bezkontextových gramatík sa zaoberá napríklad Kallmeyerovej monografia [5]. V tejto práci sa budeme zaoberať syntaktickou analýzou jazykov, ktoré sú generované gramatikami s generovaním konštantného počtu reverzov. Konkrétne sa budeme zaoberať syntaktickou analýzou zdola nahor. Navrhneme rozšírenie nedeterministickej verzie známeho algoritmu „posuň, redukuj“ (*shift-reduce*) [1] na prípad gramatík s generovaním konštantného počtu reverzov, pričom výsledný nedeterministický algoritmus na-

zveme „posuň, redukuj, otoč“. Pre každú vstupnú gramatiku bude tento nedeterministický algoritmus možné implementovať na (nedeterministickom) automate s otáčacím zásobníkom, ktorý vstupné slovo akceptuje práve vtedy, keď patrí do jazyka generovaného danou gramatikou. Determinizácie tohto algoritmu pre určité špeciálne triedy bezkontextových gramatík s generovaním reverzov (napríklad vhodné obdoby LR(0) alebo LR(k) gramatík) ponechávame ako otvorený problém pre ďalší výskum.

V prvej kapitole uvedieme definície pojmov, ktoré sa budú v práci používať. Zhrnieme známe výsledky o bezkontextových gramatikách s generovaním reverzov a automatoch s otáčacím zásobníkom. V druhej kapitole opíšeme samotný nedeterministický algoritmus „posuň, redukuj, otoč“. Následne definujeme normálne odvodenia v gramatikách s generovaním reverzov, ktoré tento algoritmus počas svojho behu konštruuje. V závere druhej kapitoly dokážeme správnosť nového nedeterministického algoritmu „posuň, redukuj, otoč“.

Označenia

V tejto práci budeme používať nasledovné označenia:

- $|w|$ – dĺžka slova w,
- $|w|_c$ – počet znakov c v slove w,
- w^R – reverz (zrkadlový obraz) slova w,
- ε – prázdne slovo,
- $\mathscr{L}(CF)$ – trieda všetkých bezkontextových jazykov,
- $\mathscr{L}(RE)$ – trieda všetkých rekurzívne vyčísliteľných jazykov,
- 2^X – množina všetkých podmnožín množiny X,
- 2^X_{fin} – množina všetkých konečných podmnožín množiny X,
- $X - Y$ – rozdiel množín, všetky prvky množiny X okrem prvkov množiny Y,
- $A \subseteq B$ – množina A je podmnožina množiny B,
- $A \subsetneq B$ – množina A je vlastná podmnožina množiny B.

Kapitola 1

Definície a známe výsledky

V tejto kapitole definujeme bezkontextové gramatiky s generovaním reverzov. Uvedieme známe výsledky z tejto oblasti. Motiváciou pre zavedenie bezkontextových gramatík s generovaním reverzov [6] bolo nájsť triedu gramatík, ktorá bude generovať rovnakú triedu jazykov, ako akceptujú automaty s otáčacím zásobníkom (angl. *Flip-Pushdown Automata*). Stručne uvedieme známe výsledky, na ktoré v druhej kapitole tejto práce nadviažeme. Definície základných pojmov z teórie formálnych jazykov možno nájsť v [4].

1.1 Gramatiky s generovaním reverzov

Bezkontextová gramatika s generovaním reverzov je bezkontextová gramatika rozšírená o špeciálny terminálny symbol ® a jeho následné vyhodnotenie. Pracuje nasledovne:

- Vygeneruje slovo podľa pravidiel danej bezkontextovej gramatiky, v ktorom sa môže nachádzať niekoľko terminálnych symbolov ®.
- Postupne zľava doprava vyhodnotí symboly ® tak, že otočí zvyšok slova za týmto symbolom a tento symbol ® odstráni zo slova.
- Takto vytvorené slovo patrí do jazyka generovaného touto gramatikou.

Teraz zadefinujeme bezkontextové gramatiky s generovaním reverzov formálne.

Definícia 1.1.1. Pätica $G = (N, T, P, \sigma, ®)$ je bezkontextová gramatika s generovaním reverzov ($RGCFG$, z angl. *reversal-generating context-free grammar*), kde (N, T, P, σ) je bezkontextová gramatika a $® \in T$ je špeciálny terminálny symbol.

Štvoricu (N, T, P, σ) interpretujeme rovnako ako v štandardných bezkontextových gramatikách, kde N je množina neterminálov, T je množina terminálov, predpokladáme $N \cup T = \emptyset$, $P \subseteq N \times (N \cup T)^*$ je množina pravidiel[1] a $\sigma \in N$ je počiatočný neterminál.

[1] Pravidlo štandardne uvádzame v tvare $\xi \to w$, kde $\xi \in N$ a $w \in (N \cup T)^*$.

Definícia 1.1.2. Nech $G = (N, T, P, \sigma, ®)$ je bezkontextová gramatika s generovaním reverzov a $G' = (N, T, P, \sigma)$ je štandardná bezkontextová gramatika. Krok odvodenia v gramatike G je taká binárna relácia $\Rightarrow_G$ na $(N \cup T)^*$, že pre $u, v \in (N \cup T)^*$ platí $u \Rightarrow_G v$ práve vtedy, keď platí $u \Rightarrow_{G'} v$ v gramatike G'. Ak je jasné, ktorej gramatiky sa krok odvodenia týka, píšeme namiesto $\Rightarrow_G$ iba $\Rightarrow$.

Definícia 1.1.3. Nech $G = (N, T, P, \sigma, ®)$ je bezkontextová gramatika s generovaním reverzov a $G' = (N, T, P, \sigma)$ je štandardná bezkontextová gramatika. Jazyk generovaný pôvodnou bezkontextovou gramatikou budeme označovať $L_{CF}(G) := L(G')$. Jazyk generovaný gramatikou G potom definujeme ako $L(G) = \{\varrho(w) \mid w \in L_{CF}(G)\}$, kde funkciu ϱ definujeme induktívne:

$$\varrho(w) = \begin{cases} w & w \in (T - \{®\})^*, \\ u\varrho(v^R) & w = u®v, \text{ kde } u \in (T - \{®\})^* \text{ a } v \in T^*. \end{cases}$$

Interpretáciu symbolov ® pomocou funkcie ϱ ilustrujeme na nasledujúcom príklade.

Príklad 1.1.4. Uvažujme slovo $w = w_1®w_2®w_3®w_4®w_5 \in L_{CF}(G)$ pre nejakú gramatiku $G = (N, T, P, \sigma, ®)$ s generovaním reverzov a slová $w_1, \ldots, w_5 \in (T - \{®\})^*$ neobsahujú symbol ®. Potom

$$\begin{aligned} \varrho(w) &= w_1\varrho(w_5^R®w_4^R®w_3^R®w_2^R) = \\ &= w_1w_5^R\varrho(w_2®w_3®w_4) = \\ &= w_1w_5^Rw_2\varrho(w_4^R®w_3^R) = \\ &= w_1w_5^Rw_2w_4^R\varrho(w_3) = w_1w_5^Rw_2w_4^Rw_3. \end{aligned}$$

Slovo $w_1w_5^Rw_2w_4^Rw_3$ patrí do jazyka $L(G)$.

Zovšeobecnením pozorovania z príkladu 1.1.4 dostávame nasledujúce tvrdenie ľahko dokázateľné matematickou indukciou. Tvrdenie je dokázané v článku [7].

Tvrdenie 1.1.5. *Nech $G = (N, T, P, \sigma, ®)$ je bezkontextová gramatika s generovaním reverzov. Potom pre jazyk generovaný gramatikou G platí*

$$\begin{aligned} L(G) = &\{w_1w_{2n}^Rw_2w_{2n-1}^R \ldots w_nw_{n+1}^R \mid n \in \mathbb{N}; w_1®w_2® \ldots ®w_{2n} \in L_{CF}(G)\} \cup \\ &\cup \{w_1w_{2n+1}^Rw_2w_{2n}^R \ldots w_nw_{n+2}^Rw_{n+1} \mid n \in \mathbb{N}; w_1®w_2® \ldots ®w_{2n+1} \in L_{CF}(G)\}. \end{aligned}$$

Definícia 1.1.6. Nech $G = (N, T, P, \sigma, ®)$ je bezkontextová gramatika s generovaním reverzov. Vetná forma v G je slovo $w \in (N \cup T)^*$ také, že platí $\sigma \Rightarrow^* w$.

V nasledujúcom opíšeme triedy gramatík, ktoré majú obmedzenie na počet vygenerovaných terminálnych symbolov ®. Nech $k \in \mathbb{N}$. Hovoríme, že gramatika generuje najviac k reverzov, ak v každej vetnej forme u platí, že $|u|_® \leq k$. Hovoríme, že gramatika generuje práve k reverzov, ak pre každú vetnú formu u platí $|u|_® \leq k$ a pre každé terminálne slovo $w \in L_{CF}(G) : |w|_® = k$.

Označenie 1.1.7. Symbolom $\mathscr{L}(RGCFG_k)$ označujeme triedu všetkých jazykov generovaných bezkontextovými gramatikami s generovaním najviac k reverzov.

Triedu všetkých jazykov generovaných $RGCFG$ s obmedzeným počtom generovaných reverzov na konštantu označujeme

$$\mathscr{L}(RGCFG_{fin}) = \bigcup_{k=0}^{\infty} \mathscr{L}(RGCFG_k).$$

Triedu jazykov generovaných bezkontextovými gramatikami s generovaním reverzov bez obmedzenia na počet generovaných reverzov označujeme $\mathscr{L}(RGCFG)$.

Pre neskoršie účely zavedieme normálny tvar pre gramatiky s generovaním reverzov (uvedený ako tvrdenie 2.6 článku [7]). V tomto normálnom tvare vieme pre každý neterminál v gramatike určiť, ktoré reverzy sa z neho vygenerujú v ľubovoľnom terminálnom slove. Navyše pre každý neterminál platí, že množina indexov reverzov, ktoré sa z tohto neterminálu vygenerujú, je v každom terminálnom slove rovnaká. Aby sme mohli splniť túto podmienku aj pre počiatočný neterminál, musí platiť, že gramatika generuje konštantný počet reverzov k pre nejaké $k \in \mathbb{N}$. Potom tento počiatočný neterminál vždy generuje všetky symboly reverzu v slove od prvého až po k-ty.

Definícia 1.1.8. Nech $G = (N, T, P, \sigma, ®)$ je bezkontextová gramatika s generovaním práve k reverzov, kde $k \in \mathbb{N}$. Gramatika G je v normálnom tvare so *znalosťou generovaných reverzov*, ak pre všetky $\xi \in \mathbb{N}$ platí jedna z dvoch nasledujúcich podmienok:

(i) Pre všetky $w \in T^*$: ak $\xi \Rightarrow^* w$, tak w neobsahuje žiadne symboly ®.

(ii) Existujú $i, j \in \{1, \ldots, k\}$ také, že pre každé $x, z, y \in T^*$ platí: ak $\sigma \Rightarrow^* x\xi y \Rightarrow^* xzy$, tak $|x|_® = i - 1$ a $|y|_® = k - j$.

Pre gramatiku G potom zavedieme funkciu $\phi_G : \mathbb{N} \to 2^{\{1,\ldots,k\}}$ takú, že pre $\xi \in N$ spĺňajúce (i) je $\phi_G(\xi) = \emptyset$ a pre $\xi \in N$ spĺňajúce (ii) s danými i, j je $\phi_G(\xi) = \{i, \ldots, j\}$. Ak je gramatika G zrejmá z kontextu, píšeme namiesto ϕ_G len ϕ.

Nech gramatika $G = (N, T, P, \sigma, ®)$ je v normálnom tvare z definície 1.1.8, s generovaním práve n reverzov. Nech $u_1®u_2$ je vetná forma pre $u_1, u_2 \in (N \cup T)^*$ v gramatike G. Potom pre tento symbol ® vieme jednoznačne určiť jeho index (poradie) v slove, ktoré sa z tejto vetnej formy vygeneruje. Nech z vetnej formy $u_1®u_2$ odvodíme ľubovoľné terminálne slovo $w_1®w_2$. Index reverzu je vďaka normálnemu tvaru medzi w_1 a w_2 vždy rovnaký. Ak $|w_1|_® = m - 1$, tak tento reverz je m-tý v poradí.

Rovnako aj pre každý ® na pravej strane nejakého pravidla vieme vďaka normálnemu tvaru gramatiky určiť, koľký reverz to bude v slove, ktoré sa z neho vygeneruje. Zoberme si pravidlo $\xi \to v_1®v_2$ pre nejaké $v_1, v_2 \in (N \cup T)^*$. Pre ľubovoľnú vetnú

formu $u_1\xi u_2$ pre $u_1, u_2 \in T^*$, v ktorej sa nachádza neterminál ξ z ľavej strany pravidla, odvodíme terminálne slovo, kde $w_1, w_2 \in T^*$:

$$u_1\xi u_2 \Rightarrow u_1 v_2 \circledR v_2 u_2 \Rightarrow^* u_1 w_1 \circledR w_2 u_2.$$

Potom symbol reverzu na pravej strane tohto pravidla je i-ty v poradí, kde $i := |u_1 w_1|_{\circledR}+1$. Nech $n := |w|_{\circledR}$ je počet reverzov v slove w, kde $w \in T^*$. *Stredný reverz v slove* w nazývame ten, ktorý je $\left\lceil \frac{n+1}{2} \right\rceil$-ty v poradí v tomto terminálnom slove. Pre stredný reverz v slove $w = w_1 \circledR w_2$ platí, že $|w_1|_{\circledR} = \left\lceil \frac{n+1}{2} \right\rceil - 1$. *Stredný reverz vetnej formy* je taký reverz vo vetnej forme $u_1 \circledR u_2$, pre ktorý platí, že pre každé terminálne slovo, ktoré sa odvodí z tejto vetnej formy $u_1 \circledR u_2 \Rightarrow^* w_1 \circledR w_2$, je tento reverz $\left\lceil \frac{n+1}{2} \right\rceil$-ty v poradí, teda $|w_1|_{\circledR} = \left\lceil \frac{n+1}{2} \right\rceil - 1$ pre $w_1, w_2 \in T^*$ a $u_1, u_2 \in (N \cup T)^*$. Hovoríme, že neterminál ξ generuje stredný reverz, ak $\left\lceil \frac{n+1}{2} \right\rceil \in \phi(\xi)$.

Poznámka 1.1.9. Z tvaru slov v tvrdení 1.1.5 možno vidieť, že stredný reverz v slove $w' \in L_{CF}(G)$ rozdeľuje toto slovo na dve časti. Symboly naľavo od stredného reverzu sa nachádzajú po aplikácii funkcie ϱ v slove $\varrho(w')$ v rovnakom poradí ako v slove w', zatiaľ čo symboly napravo od stredného reverzu sú v tomto slove $\varrho(w')$ reverzne.

Veta 1.1.10. *Pre každú bezkontextovú gramatiku G s generovaním práve n reverzov pre nejaké $n \in \mathbb{N}$ existuje ekvivalentná gramatika G' v normálnom tvare so znalosťou generovaných reverzov.*

Dôkaz. Je uvedený v článku [7]. □

1.2 Automaty s otáčacím zásobníkom

Tieto automaty vychádzajú z modelu konečného stavového automatu so zásobníkom a jednou vstupnou páskou. Automat s otáčacím zásobníkom sa môže (nedeterministicky) niekoľkokrát počas výpočtu rozhodnúť otočiť obsah zásobníka. Štúdium zásobníkových automatov s týmto rozšírením je zaujímavé z hľadiska akceptačnej sily automatov. Tejto téme sa venovali najmä Holzer a Kutrib [2, 3], ktorí prišli s výsledkom, že s počtom otočení sa zvyšuje sila automatu. Navyše, ak povolíme neobmedzený počet otočení zásobníka, sila automatu je rovnaká ako pre Turingove stroje, čomu sa venoval Sarkar [8] vo svojej práci. Tak, ako je štandardnému zásobníkovému automatu ekvivalentný model bezkontextových gramatík, tak je k zásobníkovým automatom s otáčacím zásobníkom ekvivalentný model pre bezkontextové gramatiky s generovaním reverzov opísaný v článku [6]. V tejto časti uvedieme známe výsledky o automatoch s otáčacím zásobníkom. Pre úplnosť uvádzame aj niektoré výsledky o deterministickom modeli, avšak ďalej v práci budeme pracovať iba s nedeterministickým variantom.

Definícia 1.2.1. Nedeterministický automat s otáčacím zásobníkom (NFPDA, z angl. *nondeterministic flip-pushdown automaton*) je osmica $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, kde K je neprázdna konečná množina stavov, Σ je vstupná abeceda, Γ je zásobníková abeceda, $\delta : K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{K \times \Gamma^*}_{fin}$ je štandardná prechodová funkcia nedeterministického zásobníkového automatu, $\Delta : K \to 2^K$ je otáčacia prechodová funkcia, $q_0 \in K$ je počiatočný stav, $Z_0 \in \Gamma$ je počiatočný symbol na zásobníku a $F \subseteq K$ je množina akceptačných stavov.

Definícia 1.2.2. Nech $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je nedeterministický automat s otáčacím zásobníkom. Konfigurácia automatu A je usporiadaná trojica

$$(q, w, s) \in K \times \Sigma^* \times \Gamma^*.$$

Symbol q je stav automatu, w je nedočítaná časť vstupného slova a s je slovo na zásobníku (s dnom zásobníka naľavo).

Definícia 1.2.3. Nech $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je nedeterministický automat s otáčacím zásobníkom. Krok výpočtu $\vdash_A$ je binárna relácia na konfiguráciách automatu A taká, že:

- Pre všetky $p, q \in K$, $a \in \Sigma \cup \{\varepsilon\}$, $u \in \Sigma^*$, $s, t \in \Gamma^*$ a $Z \in \Gamma$ platí $(p, au, sZ) \vdash_A (q, u, st)$, ak $(q, t) \in \delta(p, a, Z)$.
- Pre všetky $p, q \in K$, $u \in \Sigma^*$ a $s \in \Gamma^*$ platí $(p, u, Z_0 s) \vdash_A (q, u, Z_0 s^R)$, ak $q \in \Delta(p)$.
- Relácia neobsahuje žiadne iné dvojice.

Ak je zjavné, o ktorý automat sa jedná, namiesto $\vdash_A$ píšeme len $\vdash$.

Definícia 1.2.4. Nech $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je nedeterministický automat s otáčacím zásobníkom.

Jazyk akceptovaný automatom A s prázdnym zásobníkom je množina

$$N(A) = \{w \in \Sigma^* | \ \exists q \in K : (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}.$$

Jazyk akceptovaný automatom A akceptačným stavom

$$L(A) = \{w \in \Sigma^* | \ \exists q \in F \ \exists s \in \Gamma^* : (q_0, w, Z_0) \vdash^* (q, \varepsilon, s)\}.$$

Nech $A = (K, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ je nedeterministický automat s otáčacím zásobníkom. Nech $k \in \mathbb{N}$ je konštanta. Hovoríme, že automat A pracuje s najviac k otočeniami zásobníka, ak počas každého svojho výpočtu spraví najviac k krokov podľa prechodovej funkcie Δ. Hovoríme, že automat pracuje s práve k otočeniami zásobníka vtedy, ak pracuje s najviac k otočeniami zásobníka a každý jeho akceptačný výpočet obsahuje práve k otočení zásobníka podľa prechodovej funkcie Δ.

Označenie 1.2.5. Nech $k \in \mathbb{N}$ je konštanta. Triedu všetkých jazykov L takých, že $L = L(A)$ pre nejaký nedeterministický automat s otáčacím zásobníkom A pracujúci s najviac k otočeniami zásobníka, označujeme $\mathscr{L}(NFPDA_k)$.

Triedu jazykov, pre ktoré existuje automat s ohraničeným počtom otočení nejakou konštantou, označujeme

$$\mathscr{L}(NFPDA_{fin}) = \bigcup_{k=0}^{\infty} \mathscr{L}(NFPDA_k)$$

a triedu jazykov akceptovaných ľubovoľným NFPDA označujeme $\mathscr{L}(NFPDA)$.

Sila automatov s neobmedzeným počtom prevrátení zásobníka je ekvivalentná sile Turingových strojov, o čom hovorí nasledujúca veta.

Veta 1.2.6 (Sarkar [8])**.** *Platí* $\mathscr{L}(NFPDA) = \mathscr{L}(RE)$.

O sile automatov, ktoré pracujú s obmedzeným počtom prevrátení zásobníka, pojednáva nasledujúca *Holzerova-Kutribova veta o hierarchii*, ktorá hovorí, že pridaním otočenia sa zvýši výpočtová sila modelu.

Veta 1.2.7 (Holzer, Kutrib [2])**.** *Triedy jazykov* $\mathscr{L}(NFPDA_k)$ *tvoria nekonečnú hierarchiu vzhľadom na k:*

$$\mathscr{L}(CF) = \mathscr{L}(NFPDA_0) \subsetneq \mathscr{L}(NFPDA_1) \subsetneq \mathscr{L}(NFPDA_2) \subsetneq \ldots$$

Podobné výsledky platia aj pre deterministický variant automatu s otáčacím zásobníkom (Holzer, Kutrib [2]). Deterministický automat s otáčacím zásobníkom s obmedzeným počtom otočení zásobníka na konštantu definujeme podobne ako pre nedeterministický variant. Ďalej v práci však budeme pracovať len s nedeterministickým variantom.

Definícia 1.2.8. Deterministický automat s otáčacím zásobníkom (DFPDA) je (nedeterministický) automat s otáčacím zásobníkom $A = (K, \Sigma, \Gamma, \delta, \Delta, q_0, Z_0, F)$ taký, že platí:

(i) Pre všetky $q \in K, c \in \Sigma \cup \{\varepsilon\}$ a $Z \in \Gamma$ platí $|\delta(q, c, Z)| \leq 1$.

(ii) Pre všetky $q \in K$ a $Z \in \Gamma$ platí: ak $\delta(q, \varepsilon, Z) \neq \emptyset$, tak $\delta(q, c, Z) = \emptyset$ pre všetky $c \in \Sigma$.

(iii) Pre všetky $q \in K$ platí $|\Delta(q)| \leq 1$.

(iv) Pre všetky $q \in K$ platí: ak $\Delta(q) \neq \emptyset$, tak $\delta(q, c, Z) = \emptyset$ pre všetky $c \in \Sigma \cup \{\varepsilon\}$ a $Z \in \Gamma$.

Označenie 1.2.9. Nech $k \in \mathbb{N}$. Potom $\mathscr{L}(DFPDA_{\leq k})$ označujeme triedu všetkých jazykov L takých, že $L = L(A)$ pre nejaký deterministický zásobníkový automat A s otáčacím zásobníkom pracujúci s najviac k otočeniami zásobníka. Ďalej,

$$\mathscr{L}(DFPDA_{\leq fin}) = \bigcup_{k=0}^{\infty} \mathscr{L}(DFPDA_{\leq k}).$$

Označenie 1.2.10. Nech $k \in \mathbb{N}$. Potom $\mathscr{L}(DFPDA_{=k})$ označujeme triedu všetkých jazykov L takých, že $L = L(A)$ pre nejaký deterministický zásobníkový automat A s otáčacím zásobníkom pracujúci s práve k otočeniami zásobníka. Ďalej,

$$\mathscr{L}(DFPDA_{=fin}) = \bigcup_{k=0}^{\infty} \mathscr{L}(DFPDA_{=k}).$$

Triedu jazykov akceptovaných ľubovoľným DFPDA označujeme $\mathscr{L}(DFPDA)$.

Oproti nedeterministickým automatom s otáčacím zásobníkom, v deterministickom variante už neplatí rovnosť medzi triedou jazykov akceptovaných automatmi s najviac k a práve k otočeniami zásobníka.

Veta 1.2.11 (Holzer, Kutrib [3])**.** *Nech $k \geq 1$. Potom platí*

$$\mathscr{L}(DFPDA_{=k}) \subsetneq \mathscr{L}(DFPDA_{\leq k}).$$

Navyše platí $\mathscr{L}(DFPDA_{=fin}) \subsetneq \mathscr{L}(DFPDA_{\leq fin})$.

Holzerova-Kutribova veta o hierarchii platí aj pre deterministické automaty s otáčacím zásobníkom:

Veta 1.2.12 (Holzer, Kutrib [2])**.** *Triedy jazykov $\mathscr{L}(DFPDA_{\leq k})$ tvoria nekonečnú hierarchiu vzhľadom na k:*

$$\mathscr{L}(detCF) = \mathscr{L}(DFPDA_{\leq 0}) \subsetneq \mathscr{L}(DFPDA_{\leq 1}) \subsetneq \mathscr{L}(DFPDA_{\leq 2}) \subsetneq \ldots$$

Nasledujúca veta hovorí, že nedeterministický variant automatov s otáčacím zásobníkom je silnejší ako deterministický.

Veta 1.2.13 (Holzer, Kutrib [3])**.** *Nech $k \in \mathbb{N}$. Potom platí*

$$\mathscr{L}(DFPDA_{\leq k}) \subsetneq \mathscr{L}(NFPDA_k).$$

Navyše platí $\mathscr{L}(DFPDA_{\leq fin}) \subsetneq \mathscr{L}(NFPDA_{fin})$.

1.3 Porovnanie sily gramatík s automatmi

Nasledujúca veta hovorí, že pre nejakú konštantu k je trieda jazykov generovaná gramatikami s generovaním práve k reverzov rovnaká ako trieda jazykov, ktorú akceptujú nedeterministické automaty s otáčacím zásobníkom.

Veta 1.3.1. *Nech $k \in \mathbb{N}$. Potom $\mathscr{L}(NFPDA_k) = \mathscr{L}(RGCFG_k)$.*

Dôkaz. Je uvedený v článku [6]. □

Dôsledok 1.3.2. *Platí $\mathscr{L}(NFPDA_{fin}) = \mathscr{L}(RGCFG_{fin})$*

Ako dôsledok už prezentovaných výsledkov dostávame aj nasledujúcu vetu o hierarchii tried jazykov generovaných bezkontextovými gramatikami s generovaním reverzov.

Veta 1.3.3. *Platí pre všetky $k \in \mathbb{N}$: $\mathscr{L}(RGCFG_{k+1}) \supsetneq \mathscr{L}(RGCFG_k)$, teda trieda gramatík generujúca $k+1$ reverzov je silnejšia ako trieda gramatík s generovaním k reverzov.*

Dôkaz. Ide o bezprostredný dôsledok vety 1.3.1 a vety 1.2.7. □

Kapitola 2

Syntaktická analýza zdola nahor

V tejto kapitole sa zameriame na syntaktickú analýzu zdola nahor pre jazyky generované bezkontextovými gramatikami s generovaním reverzov. O štandardných bezkontextových gramatikách vieme, že sa dajú efektívne nedeterministicky parsovať na zásobníkových automatoch algoritmom „posuň, redukuj“ (*shift-reduce*) [1]. Do zásobníkového automatu pridáme operáciu „otoč“, ktorá bude predstavovať otočenie zásobníka.

Slovo $w = w_1 ® w_2 ® \dots ® w_n$ vygenerované bezkontextovou gramatikou s generovaním reverzov má po aplikácii funkcie ϱ tvar $\varrho(w) = w_1 w_{2k}^R w_2 w_{2k-1}^R \dots w_k w_{k+1}^R$ pre nepárny počet reverzov, kde $n = 2k$ a $w_1 w_{2k+1}^R w_2 w_{2k}^R \dots w_k w_{k+2}^R w_{k+1}$ pre párny počet reverzov, kde $n = 2k + 1$ v slove w (tvrdenie 1.1.5). V slove w môžeme identifikovať stredný reverz ®, ktorý rozdeľuje pôvodné vygenerované slovo na dve časti. Stredný reverz je k-ty v poradí od začiatku slova pre párny ($n = 2k$) a $(k + 1)$-vý pre nepárny ($n = 2k + 1$) celkový počet podslov w_i v slove w pre $i \in \{1 \dots n\}$.

Nech gramatika $G = (N, T, P, \sigma, ®)$ je v normálnom tvare so znalosťou generovaných reverzov (definícia 1.1.8). Potom existuje jednoznačné rozdelenie neterminálov z tejto gramatiky do dvoch množín N_A a N_B tak, že v N_A sú všetky neterminály, ktoré generujú stredný reverz ® a N_B obsahuje všetky ostatné neterminály z gramatiky G, ktoré stredný reverz negenerujú (môžu však generovať ostatné ®). Množiny N_A a N_B tvoria disjunktný rozklad množiny N. Následne vieme rovnako rozdeliť aj pravidlá z G do množín P_A a P_B podľa neterminálu, ktorý v danom pravidle prepisujeme. Pravidlo z P_A prepisuje vždy neterminál z N_A, teda generuje stredné ®. Pravidlo z P_B prepisuje neterminál z N_B a na pravej strane pravidla sa nenachádza stredný reverz ani neterminál generujúci stredný reverz. Formálne, $P_A = \{\xi \to x \in P \mid \xi \in N_A\}$ a $P_B = \{\xi \to x \in P \mid \xi \in N_B\}$. Platí $P_A \cup P_B = P$ a $P_A \cap P_B = \emptyset$. V nasledujúcom uvažujeme iba gramatiky v normálnom tvare so znalosťou generovaných reverzov a používame označenie zavedené vyššie.

2.1 Schéma „posuň, redukuj, otoč“

Opíšeme nedeterministický algoritmus „posuň, redukuj, otoč“. Do štandardného algoritmu „posuň, redukuj“ [1] pridáme operáciu „otoč“, ktorá bude reprezentovať práve jedno otočenie celého zásobníka. Uvažujme bezkontextovú gramatiku $G = (N, T, P, \sigma, ®)$ s generovaním reverzov. Slovo, ktoré gramatika použitím pravidiel vygeneruje, je slovo $w' \in L_{CF}(G)$. Toto slovo w' môže obsahovať niekoľko terminálnych symbolov ®. Pomocou funkcie ϱ vyhodnotíme všetky výskyty tohto symbolu reverzu v terminálnom slove w' tak, že dostaneme slovo $w := \varrho(w')$ patriace do jazyka $L(G)$.

Algoritmus na vstupe dostane nejaké slovo w a gramatiku G a jeho úlohou je rozhodnúť, či toto vstupné slovo w patrí do jazyka $L(G)$. Algoritmus od konca konštruuje nejaké odvodenie slova w' zo štandardnej bezkontextovej gramatiky $L_{CF}(G)$. Algoritmus sa nedeterministicky rozhoduje, kedy otočí zásobník, čo reprezentuje uhádnutie pozície jedného symbolu reverzu ® na vstupe. Operácia „posuň“ štandardne načíta jeden znak zo vstupu a vloží ho na vrch zásobníka. Operácia „redukuj podľa $\xi \to w$“ vždy pracuje na základe niektorého z pravidiel gramatiky, kde zo zásobníka vyberie slovo w a nahradí ho neterminálom ξ, ktorý vloží na vrch zásobníka. Ak sa jedná o pravidlo $\xi \to w$ zo skupiny P_B, teda neterminál negenerujúci stredný ®, tak operácia „redukuj podľa $\xi \to w$“ štandardne vytiahne celú pravú stranu pravidla w z vrchu zásobníka a na vrch vloží neterminál ξ z ľavej strany pravidla. Ak je zásobník otočený počas redukcie naopak ako na začiatku behu algoritmu, tak automat vytiahne reverz pravej strany pravidla w^R a vloží na vrch zásobníka neterminál ξ. Ak sa jedná o pravidlo $\xi \to w_1\xi_2w_2$ zo skupiny P_A, kde $\xi, \xi_2 \in N_A$ a neterminál generuje ® nepriamo (na pravej strane pravidla existuje neterminál ξ_2, ktorý toto stredné ® generuje neskôr v odvodení), tak je pravá strana pravidla, podľa ktorého redukujeme, rozdelená na vrchu a spodku zásobníka.

- Ak je zásobník otočený rovnako ako na začiatku, tak automat vyberie zo zásobníka podslovo $w_1\xi_2$ z pravej strany pravidla, otočí zásobník a vyberie druhú časť pravej strany pravidla zo zásobníka w_2^R.

- Ak je zásobník otočený naopak ako na začiatku algoritmu, tak automat vyberie zo zásobníka slovo $w_2^R\xi_2$, otočí zásobník a vyberie zvyšok pravej strany, teda w_1.

Špeciálne, ak sa jedná o pravidlo $\xi \to w_1®w_2$ zo skupiny P_A, kde neterminál generuje ® priamo[1], tak:

- Ak je zásobník otočený rovnako ako na začiatku, tak automat vyberie zo zásobníka slovo w_1, otočí zásobník a vyberie druhú časť pravej strany pravidla w_2^R ®.

[1] Symbol stredného reverzu sa nachádza na pravej strane pravidla.

- Ak je zásobník otočený naopak ako na začiatku algoritmu, tak automat vyberie zo zásobníka slovo w_2^R, otočí zásobník a vyberie zvyšok pravej strany, teda w_1Ⓡ.

Na konci každého kroku redukcie automat vloží na vrch zásobníka neterminál ξ z ľavej strany pravidla, podľa ktorého sa redukovalo. Takýto algoritmus možno zrejme implementovať na zásobníkovom automate s otáčacím zásobníkom.

Definujeme najprv operácie POSUŇ, OTOČ a REDUKUJ $(\xi \to w)$ pre každé pravidlo $\xi \to w \in P$. Nasledovné operácie predstavujú podprogramy, ktoré volá hlavný algoritmus. Zamietnutie výpočtu počas behu operácie znamená zamietnutie vstupu v algoritme.

Označenie 2.1.1. Otočenie zásobníka nadobúda dve hodnoty, a to ↑, ak je otočený ako na začiatku a ↓, ak je otočený opačne.

Operácia POSUŇ

1. Prečítaj znak zo vstupu a vlož tento znak na vrch zásobníka. V prípade neúspechu zamietni výpočet.

Operácia OTOČ

1. Vlož na vrch zásobníka symbol Ⓡ.
2. Otoč obsah zásobníka.

Operácia REDUKUJ $(\xi \to w)$
Operácia berie jeden argument, pravidlo $\xi \to w \in P$ z nejakej gramatiky $G = (N, T, P, \sigma, Ⓡ)$ v normálnom tvare so znalosťou generovaných reverzov.

1. Ak je $\xi \in N_A$ a $w = w_1Ⓡw_2$, kde tento Ⓡ je stredný, tak:
   (a) Ak je otočenie zásobníka ↑, tak vyber w_1, otoč zásobník, vyber $w_2^RⓇ$.
   (b) Ak je otočenie zásobníka ↓, tak vyber w_2^R, otoč zásobník, vyber w_1Ⓡ.

   V prípade neúspechu pri vyberaní zo zásobníka zamietni výpočet.
   Inak pokračuj krokom 4.
2. Ak je $\xi \in N_A$ a $w = w_1\xi_2w_2$, kde $\xi_2 \in N_A$, tak:
   (a) Ak je otočenie zásobníka ↑, tak vyber $w_1\xi_2$, otoč zásobník, vyber w_2^R.
   (b) Ak je otočenie zásobníka ↓, tak vyber $w_2^R\xi_2$, otoč zásobník, vyber w_1.

   V prípade neúspechu pri vyberaní zo zásobníka zamietni výpočet.
   Inak pokračuj krokom 4.

3. Ak je $\xi \in N_B$, tak:

   (a) Ak je otočenie zásobníka $\uparrow$, vyber w z vrchu zásobníka.

   (b) Inak vyber w^R.

   V prípade neúspechu pri vyberaní zo zásobníka zamietni výpočet.
   Inak pokračuj krokom 4.

4. Vlož neterminál ξ na vrch zásobníka.

Nedeterministický algoritmus „posuň, redukuj, otoč“
VSTUP: slovo w a gramatika $G = (N, T, P, \sigma, ®)$ v normálnom tvare so znalosťou generovaných reverzov,
VÝSTUP: ÁNO práve vtedy, keď $w \in L(G)$.

1. Na dno zásobníka vlož symbol $||$

2. Nedeterministicky vyber a vykonaj jednu z operácií:

   - POSUŇ
   - OTOČ
   - Pre nejaké nedeterministicky vybraté pravidlo $\xi \to x \in P_B$ vykonaj operáciu REDUKUJ $(\xi \to x)$.

3. Ak je na zásobníku práve slovo $||\sigma$ a ak je dočítané celé vstupné slovo, akceptuj, inak nedeterministicky pokračuj krokom 2 alebo 4.

4. Nedeterministicky vyber pravidlo $\xi \to x \in P_A$, kde $x = w_1®w_2$ a toto ® je stredné. Vykonaj operáciu REDUKUJ $(\xi \to x)$.

5. Ak je na zásobníku práve slovo $||\sigma$ a ak je dočítané celé vstupné slovo, akceptuj, inak pokračuj krokom 6.

6. Nedeterministicky vyber pravidlo $\xi \to x \in P_A$ a vykonaj operáciu REDUKUJ $(\xi \to x)$. Pokračuj krokom 5.

Poznámka 2.1.2. Algoritmus môže akceptovať slovo v kroku 3 len vtedy, keď toto slovo neobsahuje žiadny reverz, respektíve keď existuje slovo $w' \in L_{CF}(G)$ také, že $|w'|_® = 0$ a pre toto slovo zrejme platí $\varrho(w') = w' = w$. Keďže v algoritme uvažujeme gramatiku v normálnom tvare so znalosťou generovaných reverzov, tak všetky slová, ktoré patria do $L_{CF}(G)$, neobsahujú žiaden reverz. Potom tento algoritmus zrejme pracuje rovnako ako štandardný nedeterministický algoritmus „posuň, redukuj“.

Poznámka 2.1.3. Algoritmus sa dá upraviť tak, aby konštruoval strom odvodenia vstupného slova.

Poznámka 2.1.4. Pri vykonávaní algoritmu uvažujeme len také gramatiky, kde počiatočný neterminál σ nie je na pravej strane žiadneho pravidla. Tento predpoklad môžeme ľahko splniť tak, že pre vstupnú gramatiku $G = (N, T, P, \sigma, ®)$ vytvoríme novú gramatiku $G' = (N \cup \{\sigma'\}, T, P \cup \{\sigma' \to \sigma\}, \sigma', ®)$ s novým počiatočným neterminálom σ'. Zrejme táto gramatika generuje rovnakú množinu slov ako gramatika G. Tento predpoklad uľahčí kontrolu zásobníka na akceptačnú podmienku v kroku 3 a 5, kde v prípade falošného konca výpočtu by musel zásobník v najhoršom prípade vykonať navyše 2 otočenia (vybrať σ, nájsť symbol $||$, otočiť zásobník, nájsť symbol $||$ a v prípade neúspechu znova otočiť zásobník a vložiť σ späť na vrch zásobníka).

Príklad 2.1.5. Zoberme gramatiku $G = (N, T, P, \sigma, ®)$, kde
$N = \{\sigma, \alpha, \beta\}$, $T = \{\ a,\ b,\ \#,\ ®\ \}$,

$$\begin{aligned} P = \{&\sigma \to \alpha\beta, \\ &\alpha \to a\alpha b \mid ®, \\ &\beta \to a\beta b \mid ®\#\} \end{aligned}$$

a $L_{CF}(G) = \{a^n®b^na^m®\#b^m \mid m, n \in \mathbb{N}\}$. Táto gramatika zrejme generuje jazyk $L(G) = \{a^nb^m\#b^na^m \mid m, n \in \mathbb{N}\}$. Neterminály rozdelíme na také, ktoré generujú ® (označíme ich N_A) a ostatné (N_B). $N_A = \{\sigma, \beta\}$ a $N_B = \{\alpha\}$.

Použitím pravidiel z gramatiky odvodíme terminálne slovo $a^2®b^2a^3®\#b^3 \in L_{CF}(G)$.

$$\underline{\sigma} \Rightarrow \underline{\alpha}\beta \Rightarrow^* aa\underline{\alpha}bb\beta \Rightarrow aa®bb\underline{\beta} \Rightarrow^* a^2®b^2a^3\underline{\beta}b^3 \Rightarrow a^2®b^2a^3®\#b^3$$

Toto vygenerované slovo následne upravíme funkciou ϱ tak, aby sme dostali slovo z jazyka $L(G)$.

$$\varrho(a^2®b^2a^3®\#b^3) = a^2\varrho(b^3\#®a^3b^2) = a^2b^3\#\varrho(b^2a^3) = a^2b^3\#b^2a^3$$

Algoritmus „posuň, redukuj, otoč“ dostane na vstupe gramatiku G a slovo $a^2b^3\#b^2a^3$ z tejto gramatiky. V tabuľke 2.1 je znázornené parsovanie tohto slova.

	zásobník	**vstup**	**operácia**
↑	$\vdash \|\|$	$a^2b^3\#b^2a^3$	2 x POSUŇ
↑	$\vdash \|\|aa$	$b^3\#b^2a^3$	OTOČ
			(vlož Ⓡ na zásobník, otoč zásobník)
↓	$\vdash Ⓡaa\|\|$	$b^3\#b^2a^3$	3 x POSUŇ
↓	$\vdash Ⓡaa\|\|bbb$	$\#b^2a^3$	POSUŇ
↓	$\vdash Ⓡaa\|\|bbb\#$	b^2a^3	OTOČ
			(vlož Ⓡ na zásobník, otoč zásobník)
↑	$\vdash Ⓡ\#bbb\|\|aaⓇ$	b^2a^3	REDUKUJ (podľa pravidla $\alpha \to Ⓡ$ z P_B)
↑	$\vdash Ⓡ\#bbb\|\|aa\alpha$	b^2a^3	POSUŇ
↑	$\vdash Ⓡ\#bbb\|\|aa\alpha b$	ba^3	REDUKUJ (podľa pravidla $\alpha \to a\alpha b$ z P_B)
↑	$\vdash Ⓡ\#bbb\|\|a\alpha$	ba^3	POSUŇ
↑	$\vdash Ⓡ\#bbb\|\|a\alpha b$	a^3	REDUKUJ (podľa pravidla $\alpha \to a\alpha b$ z P_B)
↑	$\vdash Ⓡ\#bbb\|\|\alpha$	a^3	3 x POSUŇ
↑	$\vdash Ⓡ\#bbb\|\|\alpha aaa$		REDUKUJ (podľa pravidla $\beta \to Ⓡ\#$ z P_A)
			(vyber prázdne slovo, otoč, vyber $\#Ⓡ$, vlož β)
↓	$\vdash aaa\alpha\|\|bbb\beta$		REDUKUJ (podľa pravidla $\beta \to a\beta b$ z P_A)
			(vyber $b\beta$, otoč, vyber a, vlož β)
↑	$\vdash bb\|\|\alpha aa\beta$		REDUKUJ (podľa pravidla $\beta \to a\beta b$ z P_A)
			(vyber $a\beta$, otoč, vyber b, vlož β)
↓	$\vdash a\alpha\|\|b\beta$		REDUKUJ (podľa pravidla $\beta \to a\beta b$ z P_A)
			(vyber $b\beta$, otoč, vyber a, vlož β)
↑	$\vdash \|\|\alpha\beta$		REDUKUJ (podľa pravidla $\sigma \to \alpha\beta$ z P_A)
			(vyber $\alpha\beta$, otoč, vyber prázdne slovo, vlož σ)
↓	$\vdash \|\|\sigma$		akceptuj

Tabuľka 2.1: Parsovanie slova $a^2b^3\#b^2a^3$

Algoritmus pracuje v dvoch fázach. V prvej fáze používa operácie POSUŇ, OTOČ a REDUKUJ ($\xi \to w$) iba pre pravidlá $\xi \to w \in P_B$, ktoré negenerujú stredný reverz. V prvej fáze teda redukuje iba neterminál α z N_B. Algoritmus pracuje podľa krokov 1 až 3. Následne algoritmus prejde do druhej fázy, kde už používa iba operáciu REDUKUJ ($\xi \to w$) pre neterminály z N_A. Krok 4 algoritmus vykoná práve raz, a to pre pravidlo $\beta \to Ⓡ\#$, ktoré generuje stredné Ⓡ priamo na pravej strane tohto pravidla. V každom kroku redukcie algoritmus využije nejaké pravidlo z gramatiky. Podľa týchto použitých pravidiel možno zostrojiť strom odvodenia slova $a^2Ⓡb^2a^3Ⓡ\#b^3$.

2.2 Normálne odvodenia

Nedeterministický algoritmus „posuň, redukuj, otoč“ pri svojom výpočte na vstupe pozostávajúcom z gramatiky G a slova w od konca konštruuje nejaké odvodenie vstupného slova w v tejto gramatike G. V nasledujúcom opíšeme toto odvodenie, nazveme ho *normálne odvodenie*[2]. Neskôr o ňom dokážeme, že je rovnaké, ako konštruuje nedeterministický algoritmus „posuň, redukuj, otoč“.

Nech G je bezkontextová gramatika s generovaním práve n reverzov v normálnom tvare so znalosťou generovaných reverzov. Každé slovo $w \in L_{CF}(G)$ potom vieme zapísať v tvare $w = w_1 Ⓡ w_2 Ⓡ \ldots Ⓡ w_{n+1}$, kde pre každé z $n+1$ podslov platí $w_1, \ldots, w_{n+1} \in (T - \{Ⓡ\})^*$. V normálnom odvodení budú reverzy generované v poradí $A_1, \ldots, A_n$, kde:

$$A_k = \left\lceil \frac{n+1}{2} \right\rceil + (-1)^{n+k+1} \left\lfloor \frac{k}{2} \right\rfloor.$$

Pre párny počet reverzov je postupnosť v tvare $s, s-1, s+1, s-2, \ldots, n, 1$, kde s je index stredného reverzu.

Presnejšie pôjde o odvodenie nasledujúceho tvaru. V odvodení ako prvý (A_1) vygenerujeme symbol stredného reverzu tak, že prepisujeme vždy neterminál, ktorý generuje tento stredný reverz: $x \Rightarrow^* x_1 Ⓡ x_2$. Reverz medzi slovami x_1 a x_2 je stredný. Následne generujeme reverz s indexom $(s-1)$. Ak podslovo x_1 obsahuje neterminál ξ taký, že $(s-1) \in \phi(\xi)$, teda $x_1 = x_1' \xi x_1''$ pre nejaké $x_1', x_1'' \in (N \cup T)^*$, tak použijeme pravé krajné odvodenie na toto podslovo x_1. Opakujeme toto pravé krajné odvodenie na nové podslovo naľavo od stredného reverzu, kým toto podslovo obsahuje neterminál generujúci $(s-1)$-vý reverz. Keď je vygenerovaný $(s-1)$-vý reverz, ľavým krajným odvodením prepisujeme pravidlá v slove x_2, kým vygenerujeme reverz $s+1$. Takto striedame ľavé krajné a pravé krajné odvodenie, kým sú v slove ešte nejaké neterminály generujúce nejaký reverz. Ak už v slove nie sú žiadne neterminály generujúce reverz (ale existuje $\xi \in N$ také, že $\phi(\xi) = \emptyset$), tak najprv ľavým krajným odvodením prepíšeme všetky neterminály v podslove x_2 a následne pravým krajným odvodením všetky zvyšné neterminály v podslove x_1 naľavo od stredného reverzu.

Pre nepárny počet reverzov v slove opäť začneme vygenerovaním stredného reverzu. Následne pokračujeme ľavým krajným odvodením z podslova x_2 generovať $(s+1)$-vý reverz. Postupne striedame generovanie ďalších reverzov podľa postupnosti na pravej a ľavej strane od stredného reverzu. Pre nepárny počet reverzov je postupnosť v tvare $s, s+1, s-1, s+2, \ldots, n, 1$, kde s je index stredného reverzu.

[2] V skutočnosti ide o odvodenie nejakého terminálneho slova $w' \in L_{CF}(G)$ pre túto gramatiku G, pre ktoré platí, že $\varrho(w') = w$.

Definícia 2.2.1 (Normálne odvodenie)**.** Nech $G = (N, T, P, \sigma, ®)$ je *RGCFG* v normálnom tvare so znalosťou generovaných reverzov. Normálne odvodenie v G je také odvodenie, že všetky kroky $x \Rightarrow y$ tohto odvodenia spĺňajú nasledovné podmienky:

a) Ak x obsahuje neterminál ξ taký, že $\phi(\xi) \neq \emptyset$, zvoľme $i \in \{1, \ldots, n\}$ najmenšie také, že x obsahuje neterminál η taký, že $A_i \in \phi(\eta)$. Potom:

   i) Ak $i = 1$, tak y vznikne z x prepísaním neterminálu η.[3]

   ii) Ak $A_i < A_1$, tak y vznikne z x prepísaním najpravejšieho neterminálu, ktorý sa v x nachádza naľavo od stredného reverzu.[4]

   iii) Ak $A_i > A_1$, tak y vznikne z x prepísaním najľavejšieho neterminálu, ktorý sa v x nachádza napravo od stredného reverzu.

b) V opačnom prípade, ak x obsahuje nejaký neterminál v podslove napravo od stredného ®, tak y vznikne prepísaním najľavejšieho neterminálu napravo od stredného ®.

c) Inak, ak x obsahuje nejaký neterminál naľavo od stredného ®, tak y vznikne prepísaním najpravejšieho neterminálu naľavo od stredného ®.

Príklad 2.2.2. Zoberme gramatiku $G = (N, T, P, \sigma, ®)$ v normálnom tvare z definície 1.1.8, kde $N = \{\sigma, \gamma_{12}, \gamma_1, \gamma_2, \xi_2, \gamma_{36}, \gamma_{46}, \gamma_{45}\}$, $T = \{a_1, a_2, a_3, a_4, a_5, a_6, ®\}$ a

$$\begin{aligned} P = \{&\sigma \to \gamma_{12}\gamma_{36}, \\ &\gamma_{12} \to \gamma_1\gamma_2, \\ &\gamma_1 \to a_1®, \\ &\gamma_2 \to a_2\xi_2, \\ &\xi_2 \to ®, \\ &\gamma_{36} \to a_3®\gamma_{46}, \\ &\gamma_{46} \to \gamma_{45}®a_6, \\ &\gamma_{45} \to a_4®a_5\}. \end{aligned}$$

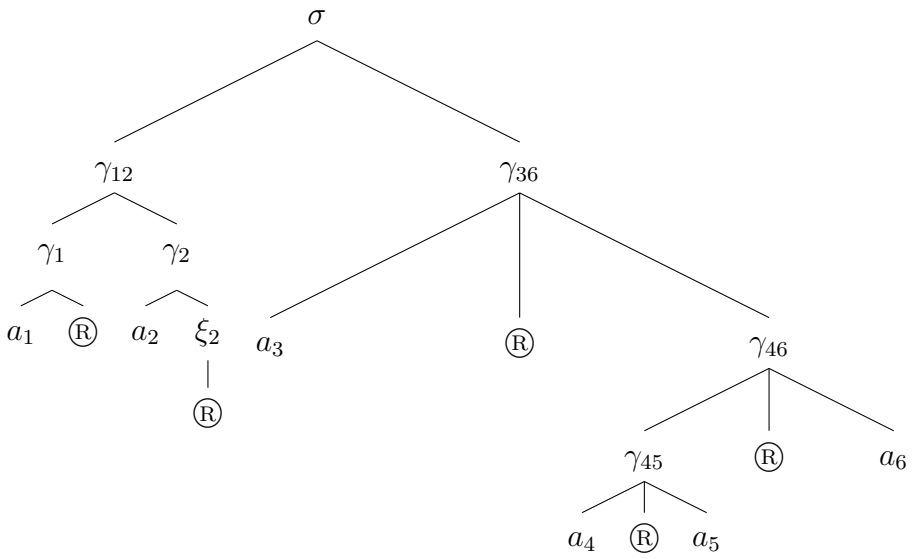
Vygenerujeme slovo $w = a_1®a_2®a_3®a_4®a_5®a_6$ z jazyka $L_{CF}(G)$ podľa pravidiel z tejto gramatiky.

[3] Z normálneho tvaru vyplýva, že takýto neterminál je v každej vetnej forme najviac jeden.

[4] Pre $i > 1$ platí, že stredný reverz vo vetnej forme x je už vygenerovaný a vieme ju zapísať v tvare $x = x_1®x_2$, kde tento reverz je stredný (vďaka normálnemu tvaru sa vždy z vetnej formy x spraví terminálne slovo, kde tento reverz je stredný).

$$
\begin{aligned}
\underline{\sigma} &\Rightarrow \gamma_{12}\underline{\gamma_{36}} \\
&\Rightarrow \gamma_{12}a_3\textcircled{R}\underline{\gamma_{46}} \\
&\Rightarrow \gamma_{12}a_3\textcircled{R}\underline{\gamma_{45}}\textcircled{R}a_6 \\
&\Rightarrow \underline{\gamma_{12}}a_3\textcircled{R}a_4\textcircled{R}a_5\textcircled{R}a_6 \\
&\Rightarrow \gamma_1\underline{\gamma_2}a_3\textcircled{R}a_4\textcircled{R}a_5\textcircled{R}a_6 \\
&\Rightarrow \gamma_1a_2\underline{\xi_2}a_3\textcircled{R}a_4\textcircled{R}a_5\textcircled{R}a_6 \\
&\Rightarrow \underline{\gamma_1}a_2\textcircled{R}a_3\textcircled{R}a_4\textcircled{R}a_5\textcircled{R}a_6 \\
&\Rightarrow a_1\textcircled{R}a_2\textcircled{R}a_3\textcircled{R}a_4\textcircled{R}a_5\textcircled{R}a_6
\end{aligned}
$$

Gramatika generuje 5 reverzov. Prvý reverz, ktorý gramatika vygeneruje, je podľa postupnosti $A_1 = 3$. V odvodení sa najprv prepíše neterminál σ a následne γ_{36}, ktoré generujú tento stredný reverz. Ďalší Ⓡ, ktorý sa bude podľa postupnosti generovať, je $A_2 = 4$. Podľa bodu (a, iii) z definície 2.2.1 sa prepisujú postupne neterminály γ_{46} a γ_{45}. Ako možno vidieť aj v tomto príklade, štvrtý reverz nebude vygenerovaný nutne ako druhý v poradí, hoci je v postupnosti A_2. Reálne poradie je určené tým, aké pravidlá sa v tej konkrétnej gramatike nachádzajú. Avšak poradie vyberania neterminálov, ktoré sa v každej vetnej forme prepisujú, ostáva rovnaké. Následne sa vygeneruje reverz $A_3 = 2$ postupným prepísaním neterminálov γ_{12}, γ_2 a ξ_2. Ďalší v poradí je $A_4 = 5$. Tento reverz bol už vygenerovaný v nejakom predošlom kroku, a preto hľadáme ďalší člen postupnosti. Posledný reverz je $A_5 = 1$, ktorý vznikne prepísaním neterminálu γ_1.



2.3 Správnosť algoritmu

Vyslovíme vetu o správnosti nedeterministického algoritmu „posuň, redukuj, otoč“, ktorú v nasledujúcej časti práce dokážeme.

Veta 2.3.1 (Správnosť algoritmu „posuň, redukuj, otoč“)**.** *Nech gramatika* $G = (N, T, P, \sigma, ®)$ *je v normálnom tvare so znalosťou generovaných reverzov. Ak algoritmus „posuň, redukuj, otoč“ z oddielu 2.1 dostane na vstupe gramatiku* G *a slovo* $w \in T^*$*, tak akceptuje svoj vstup práve vtedy, keď* $w \in L(G)$*.*

2.3.1 Odvodenie k akceptačnému výpočtu

Najprv opíšeme a sformulujeme invarianty, ktoré platia v každom kroku algoritmu. Na základe ich platnosti ukážeme, že pre každý akceptačný výpočet algoritmu na vstupe pozostávajúcom zo slova w a gramatiky G existuje odvodenie nejakého slova $w' \in L_{CF}(G)$ v gramatike G také, že $\varrho(w')$ je práve toto vstupné slovo w.

V texte nižšie budeme používať nasledovné označenie pre obsah zásobníka. Ak je otočenie zásobníka $\uparrow$, označme obsah zásobníka[5] nad zarážkou $\underline{x_\uparrow}$ a obsah zásobníka pod zarážkou $\overline{x_\uparrow}$. Celý obsah zásobníka je potom slovo $\overline{x_\uparrow}||\underline{x_\uparrow}$. Ak je otočenie zásobníka $\downarrow$, označme obsah zásobníka nad zarážkou $\underline{x_\downarrow}$ a obsah zásobníka pod zarážkou $\overline{x_\downarrow}$. V takom prípade je slovo $\overline{x_\downarrow}||\underline{x_\downarrow}$ celý obsah zásobníka.

Nech slovo na vstupe algoritmu je $w \in (T - \{®\})^*$. Algoritmus počas svojho behu rozdelí toto slovo na $k+1$ podslov určených k otočeniami zásobníka (použitím operácie OTOČ) tak, že $w = u_1 u_2 \dots u_{k+1}$, kde $u_i \in (T - \{®\})^*$ pre $i \in \{1, \dots, k+1\}$ a tieto podslová môžu byť aj prázdne. Nasledujúci príklad ilustruje takéto načítanie vstupu a jeho rozdelenie na podslová.

Príklad načítania vstupu: Algoritmus pri otočení zásobníka $\uparrow$ načíta nejaký prefix vstupného slova w, tento prefix označme u_1. Potom algoritmus použije operáciu OTOČ. Pri tejto operácii sa najprv na vrch zásobníka pridá symbol ® a potom sa otočí zásobník. Pred touto operáciou bol obsah zásobníka $||u_1$, po tejto operácii je to $®u_1^R||$. Následne algoritmus opäť načíta niekoľko symbolov zo vstupu, označme toto podslovo u_2 (toto podslovo môže byť aj prázdne). u_2 je vo vstupnom slove priamo nasledujúce za u_1. Na zásobníku sa nachádza slovo $®u_1^R||u_2$. Po ďalšom otočení zásobníka je slovo na zásobníku $®u_2^R||u_1®$. Ďalšie podslovo, ktoré algoritmus načíta zo vstupu, je u_3 atď.

Operácie POSUŇ a OTOČ spolu určujú, aký bude obsah zásobníka po načítaní jednotlivých symbolov zo vstupu. Algoritmus načítava slová s nepárnym indexom vtedy, keď

[5] Ak hovoríme, že obsahom zásobníka je slovo $a_1 \dots a_n$, tak posledný symbol tohto slova a_n je na vrchu zásobníka.

je otočenie zásobníka ↑. Slová s párnym indexom načítava, keď je otočenie zásobníka ↓. Tieto podslová slova w sú na zásobníku oddelené vždy jedným symbolom Ⓡ. Potom obsah zásobníka nad zarážkou pri otočení zásobníka ↑ bude nejaký prefix slova $u_1 Ⓡ u_3 Ⓡ \ldots$. Obsah zásobníka nad zarážkou pri otočení zásobníka ↓ bude nejaký prefix slova $u_2 Ⓡ u_4 Ⓡ \ldots$.

Operácia REDUKUJ $\xi \to a$ (podobne ako v štandardnom algoritme „posuň, redukuj“ [1]) zodpovedá použitiu tohto pravidla $\xi \to a$ v jednom kroku odvodenia vstupného slova[6] v gramatike. Pre pravidlá, ktoré negenerujú stredný reverz, je táto operácia identická ako v algoritme „posuň, redukuj“, ktorá z vrchu zásobníka vytiahne pravú stranu pravidla (respektíve jej reverz, ak je zásobník otočený naopak) a nahradí ju neterminálom na ľavej strane. Redukcia podľa pravidla, ktoré na pravej strane obsahuje stredný reverz, je o niečo komplikovanejšia. Pravá strana tohto pravidla je rozdelená na dve časti a nachádza sa na oboch koncoch zásobníka[7]. Algoritmus nahradí koniec slova $\underline{x_\uparrow}$ a začiatok slova $\overline{x_\uparrow}$ neterminálom ξ, čo opäť zodpovedá jednému kroku odvodenia v gramatike (analogicky pre otočenie zásobníka ↓). Podľa takto použitých pravidiel pre operáciu REDUKUJ vieme zostrojiť strom odvodenia pre vstupné slovo[8] ku každému akceptačnému výpočtu algoritmu.

V prípade akceptačného výpočtu algoritmu obsah zásobníka predstavuje nejakú vetnú formu alebo časť nejakej vetnej formy pre odvodenie vstupného slova v gramatike G. Nech $k+1$ je počet podslov, na ktoré sa vstupné slovo počas behu algoritmu rozdelí pomocou operácie OTOČ. Ak je počet podslov párny, $k+1 = 2m$, tak pri načítaní posledného podslova u_{2m} bude zásobník otočený ↓. Z obsahu zásobníka nad zarážkou $||$ bude odvoditeľné slovo $u_2 Ⓡ u_4 Ⓡ \ldots Ⓡ u_{2m}$. Z obsahu zásobníka pod zarážkou možno odvodiť slovo, ktoré pozostáva z podslov u s nepárnym indexom, teda $Ⓡ u^R_{2m-1} Ⓡ u^R_{2m-3} \ldots Ⓡ u^R_1$. Vetnú formu, ktorú predstavuje slovo na zásobníku pri otočení zásobníka ↓, vieme zapísať v tvare

$$\overline{x_\downarrow}^R \underline{x_\downarrow}^R \Rightarrow^* u_1 Ⓡ u_3 Ⓡ \ldots u_{2m-3} Ⓡ u_{2m-1} Ⓡ u^R_{2m} Ⓡ u^R_{2m-2} \ldots Ⓡ u^R_4 Ⓡ u^R_2.$$

Ak je počet podslov nepárny, $k+1 = 2m+1$, tak po načítaní posledného podslova u_{2m+1} bude zásobník otočený ↑. Z obsahu zásobníka nad zarážkou potom vieme odvodiť $u_1 Ⓡ u_3 Ⓡ \ldots u_{2m-1} Ⓡ u_{2m+1}$. Z obsahu zásobníka pod zarážkou vieme odvodiť slovo $Ⓡ u^R_{2m} Ⓡ u^R_{2m-2} \ldots Ⓡ u^R_2$. Vetná forma, ktorú reprezentuje slovo na zásobníku pri otočení

[6] V bezkontextovej gramatike s generovaním reverzov odvodenie existuje pre nejaké slovo w' z jazyka $L_{CF}(G)$, pre ktoré platí, že vyhodnotením symbolov Ⓡ v slove w' pomocou funkcie ϱ dostaneme vstupné slovo w pre tento algoritmus.

[7] Ak nedeterministický algoritmus uhádne toto odvodenie slova w', tak symboly naľavo od stredného reverzu v slove w' sa načítavajú na zásobník pri otočení zásobníka ↑ a symboly napravo od stredného reverzu v slove w' sa načítavajú na zásobník v reverznom poradí pri otočení zásobníka ↓.

[8] Strom odvodenia existuje pre slovo w', z ktorého dostaneme vstupné slovo interpretovaním symbolov Ⓡ pomocou funkcie ϱ.

zásobníka ↑, bude v tvare

$$\underline{x_{\uparrow}}\overline{x_{\uparrow}} \Rightarrow^* u_1 Ⓡ u_3 Ⓡ \ldots u_{2m-1} Ⓡ u_{2m+1} Ⓡ u_{2m}^R Ⓡ u_{2m-2}^R \ldots Ⓡ u_4^R Ⓡ u_2^R.$$

Nakoniec postupnou aplikáciou funkcie ϱ na terminálne slovo $u_1 Ⓡ u_3 Ⓡ \ldots Ⓡ u_4^R Ⓡ u_2^R$ z jazyka $L_{CF}(G)$ dostaneme slovo w, ktoré bolo na vstupe algoritmu:

$$\begin{aligned}\varrho(u_1 Ⓡ u_3 Ⓡ \ldots Ⓡ u_4^R Ⓡ u_2^R) &= u_1 \varrho(u_2 Ⓡ u_4 Ⓡ \ldots Ⓡ u_5^R Ⓡ u_3^R) = \\ &= u_1 u_2 \varrho(u_3 Ⓡ u_5 Ⓡ \ldots Ⓡ u_6^R Ⓡ u_4^R) = \\ &= \ldots \\ &= u_1 u_2 \ldots u_{k+1}\end{aligned}$$

Krok 4 v algoritme rozdeľuje beh algoritmu na dve časti. V prvej časti sa primárne načítava slovo zo vstupu a redukcie sa robia iba s pravidlami, ktoré negenerujú stredné Ⓡ. Na obsah zásobníka nad zarážkou a obsah zásobníka pod zarážkou sa môžeme pozerať ako na dve navzájom nezávislé odvodenia. To, čo ich spája, je počet otočení zásobníka, ktorý jednoznačne určuje, koľko symbolov Ⓡ sa aktuálne na zásobníku nachádza. V druhej časti algoritmu sa používa iba operácia REDUKUJ $(\xi \to w)$ pre pravidlá, ktoré generujú stredné Ⓡ (z množiny P_A). Pri každom použití tejto operácie sa zásobník otočí práve raz. Algoritmus už nepoužíva operáciu POSUŇ, a tak ide iba o postupnosť redukcií na vetnej forme, ktorú vieme zostrojiť z obsahu zásobníka nad a pod zarážkou.

Poznámka 2.3.2. Na úplnom spodku zásobníka je v prvej časti algoritmu (pred krokom 4) symbol || zarážky alebo symbol Ⓡ reverzu. Zarážka je tam pred prvým použitím operácie OTOČ, teda pri načítaní prvého podslova u_1. Následne každé použitie operácie OTOČ v kroku 2 pridá symbol Ⓡ na vrch a ihneď otočí zásobník, čo tento symbol dostane na spodok.

Teraz definujeme invarianty, ktoré platia v každom kroku nedeterministického algoritmu „posuň, redukuj, otoč“.

Invarianty pre algoritmus

Nech w je doposiaľ prečítaná časť vstupu. Nech k je počet otočení zásobníka doposiaľ vykonaných použitím operácie OTOČ. Tieto otočenia rozdeľujú prečítanú časť vstupného slova na $k+1$ podslov, označme ich $w = u_1 u_2 \ldots u'_{k+1}$. Prvých k podslov je už načítaných. Podslovo u'_{k+1} je doposiaľ prečítaná časť vstupu od posledného otočenia zásobníka (použitím operácie OTOČ).

I1 Hovoríme, že je splnený *invariant I1*, ak platí, že zásobník je otočený ↑ a pre obsah zásobníka platí

$$\underline{x_{\uparrow}} \Rightarrow_G^* x_{up}, \qquad \text{kde } x_{up} \text{ je } u_1 ® u_3 ® \ldots ® u'_{k+1},$$
$$\overline{x_{\uparrow}} \Rightarrow_G^* x_{down}, \qquad \text{kde } x_{down}^R \text{ je } u_2 ® u_4 ® \ldots ® u_k ®.$$

Pre slová x_{up} a x_{down} navyše platí $|x_{up}|_® = |x_{down}|_®$ a $\varrho(x_{up}x_{down}) = w$.

I2 Hovoríme, že je splnený *invariant I2*, ak platí, že zásobník je otočený ↓ a pre obsah zásobníka[9] platí

$$(\underline{x_{\downarrow}})^R \Rightarrow_G^* (x_{up})^R, \qquad \text{kde } x_{up} \text{ je } u_2 ® u_4 ® \ldots ® u'_{k+1},$$
$$(\overline{x_{\downarrow}})^R \Rightarrow_G^* (x_{down})^R, \qquad \text{kde } x_{down}^R \text{ je } u_1 ® u_3 ® \ldots ® u_k ®.$$

Pre slová x_{up} a x_{down} navyše platí $|x_{up}|_® + 1 = |x_{down}|_®$ a $\varrho(x_{down}^R x_{up}^R) = w$.

I3 Hovoríme, že je splnený *invariant I3*, ak platí, že zásobník je otočený ↑, obsah zásobníka vieme zapísať v tvare $\underline{x_{\uparrow}}\overline{x_{\uparrow}} \Rightarrow_G^* w'$ a $\underline{x_{\uparrow}} = x'\xi$, kde $\xi \in N_A$[10]. Pre odvodené slovo w' platí rovnosť $\varrho(w') = w$.

I4 Hovoríme, že je splnený *invariant I4*, ak platí, že zásobník je otočený ↓ a pre obsah zásobníka platí $\overline{x_{\downarrow}}^R \underline{x_{\downarrow}}^R \Rightarrow_G^* w'$ a $\underline{x_{\downarrow}} = x'\xi$, kde $\xi \in N_A$. Pre odvodené slovo w' platí rovnosť $\varrho(w') = w$.

Tvrdenie 2.3.3. *V každom kroku nedeterministického algoritmu „posuň, redukuj, otoč“ z oddielu 2.1 je platný niektorý z invariantov I1 až I4. Ak algoritmus ešte nevykonal krok 4, tak platí invariant I1 alebo invariant I2. Ak algoritmus už vykonal krok 4, tak platí invariant I3 alebo invariant I4.*

Dôkaz. Overíme, že invariant *I1* je splnený na začiatku algoritmu. Ukážeme, že ak platí v nejakom kroku jeden z invariantov *I1* alebo *I2* a algoritmus použije ľubovoľnú z operácií POSUŇ, OTOČ alebo REDUKUJ $(\xi \to w)$, tak bude opäť platiť niektorý z invariantov *I1* alebo *I2* (nie nutne ten istý). Špeciálne, ak vykonáme operáciu REDUKUJ $(\xi \to w)$ v kroku 4, tak algoritmus prejde do invariantu *I3* alebo *I4*. Nakoniec, ak platí jeden z invariantov *I3* alebo *I4* a algoritmus použije operáciu REDUKUJ $(\xi \to w)$, tak platí opäť jeden z invariantov *I3* alebo *I4*.

Na začiatku behu algoritmu je zásobník otočený ↑ a ešte nie je načítaný žiaden symbol zo vstupu. Algoritmus nepoužil operáciu OTOČ ani raz, a preto je $k := 0$.

[9] Pravá aj ľavá strana odvodenia (vetná forma) je reverzná v každom kroku odvodenia pri otočení zásobníka ↓, pretože nemeníme sadu pravidiel v gramatike, ale na zásobníku sú slová načítané v reverznom poradí.

[10] Symbol na vrchu zásobníka je neterminál generujúci stredný reverz.

Doposiaľ prečítaná časť vstupu je prázdne slovo $w = \varepsilon$, rovnako aj obsah zásobníka nad aj pod zarážkou sú prázdne. Počet symbolov reverzu je nad aj pod zarážkou rovnako 0. Funkcia ϱ na prázdnom slove vráti opäť prázdne slovo, čo je práve w. Platí invariant *I1*.

Platí invariant *I1*

POSUŇ Operácia POSUŇ zoberie znak c zo vstupu a dá ho na vrch zásobníka. Potom, ak z predpokladu platnosti invariantu *I1* platí $\underline{x_{\uparrow}} \Rightarrow^* u_1 Ⓡ u_3 Ⓡ \ldots Ⓡ u'_{k+1}$, tak platí aj $\underline{x_{\uparrow} c} \Rightarrow^* u_1 Ⓡ u_3 Ⓡ \ldots Ⓡ u'_{k+1} c$ pre rovnakú postupnosť použitých pravidiel. Počet symbolov Ⓡ na zásobníku sa nezmenil nad ani pod zarážkou. Keďže invariant v predpoklade spĺňal podmienku, že $\varrho(x_{up} x_{down})$ je slovo w, kde posledné podslovo je u'_{k+1}, tak pre nový obsah zásobníka po tejto operácii platí, že $\varrho(x_{up} c x_{down})$ je slovo wc, kde $u'_{k+1} c$ je posledné podslovo doposiaľ prečítané zo vstupu po poslednom otočení zásobníka. Po použití tejto operácie platí invariant *I1*.

OTOČ Nech k je počet doteraz použitých operácií OTOČ. Použitím tejto operácie sa zvýši počet otočení na $k + 1$. Operácia vloží na vrch zásobníka symbol Ⓡ a otočí obsah zásobníka. Z predpokladu platnosti invariantu *I1* je obsah zásobníka pred použitím operácie

$$\begin{aligned} \underline{x_{\uparrow}} &\Rightarrow^* A && \text{pre } A = u_1 Ⓡ u_3 Ⓡ \ldots Ⓡ u'_{k+1}, \\ \overline{x_{\uparrow}} &\Rightarrow^* B && \text{pre } B^R = u_2 Ⓡ u_4 Ⓡ \ldots u_k Ⓡ \end{aligned}$$

pre nejaké $A, B \in T^*$. Po použití operácie OTOČ bude obsah zásobníka

$$\begin{aligned} (\underline{x_{\downarrow}})^R &\Rightarrow^* (B^R)^R && \text{pre } B^R = u_2 Ⓡ u_4 Ⓡ \ldots u_k Ⓡ u'_{k+2}, \\ (\overline{x_{\downarrow}})^R &\Rightarrow^* (Ⓡ A^R)^R && \text{pre } A Ⓡ = u_1 Ⓡ u_3 Ⓡ \ldots Ⓡ u_{k+1} Ⓡ, \end{aligned}$$

kde označíme $u_{k+1} := u'_{k+1}$ a u'_{k+2} je nové slovo, ktoré označuje dočítanú časť vstupu za posledným načítaním symbolu Ⓡ na zásobník. Toto slovo u'_{k+2} je prázdne. Počet Ⓡ sa zvýšil o 1 pod zarážkou, keďže z predpokladu platnosti invariantu $|A|_Ⓡ = |B|_Ⓡ$, potom platí $|B^R|_Ⓡ + 1 = |Ⓡ A^R|_Ⓡ$. Z predpokladu platnosti invariantu *I1* platí, že $\varrho(x_{up} x_{down}) = \varrho(AB)$ je slovo w. Potom pre podmienku z invariantu *I2* platí, že $\varrho(x^R_{down} x^R_{up}) = \varrho((Ⓡ A^R)^R (B^R)^R) = \varrho(A Ⓡ B)$ je slovo w. Keďže počet symbolov Ⓡ je v slovách A a B rovnaký a slovo B začína symbolom Ⓡ, tak tento nový reverz otočí pomocou funkcie len prázdne slovo. Takto dostávame platnosť invariantu *I2*.

REDUKUJ Pre operáciu REDUKUJ budeme rozlišovať tri prípady podľa toho, v akom kroku algoritmu bola táto operácia použitá.

V druhom kroku algoritmu je pravidlo $\xi \to z \in P_B$, čo znamená, že negeneruje stredný reverz. Potom vieme slovo na vrchu zásobníka $\underline{x_\uparrow}$ zapísať v tvare $\underline{x_\uparrow} = yz$ pre nejaké $y, z \in (N \cup T)^*$. Operácia vyberie slovo z z vrchu zásobníka (ak toto nie je možné vykonať, automat sa zasekne a vstup neakceptuje). Z predpokladu platnosti invariantu *I1* existuje odvodenie pre obsah zásobníka nad zarážkou $yz \Rightarrow^* u_1 ® u_3 ® \ldots ® u'_{k+1}$. Nahradením pravej strany pravidla z neterminálom ξ na vrchu zásobníka dostávame nový obsah zásobníka $y\xi$, pre ktorý platí $y\xi \Rightarrow yz$ pomocou tohto redukovaného pravidla. Pre nový obsah zásobníka nad zarážkou platí $y\xi \Rightarrow^* u_1 ® u_3 ® \ldots ® u'_{k+1}$. Po použití tejto operácie ostáva platný invariant *I1*.

Pravidlo v štvrtom kroku algoritmu priamo generuje stredné ®, a preto vieme zapísať jeho pravú stranu $z = z_1 ® z_2$. Z predpokladu na použitie operácie platí, že slovo nad zarážkou vieme rozdeliť na podslová $\underline{x_\uparrow} = xz_1$ a slovo pod zarážkou na $\overline{x_\uparrow} = ® z_2 y$ (kde je toto ® stredné) pre nejaké $x, y, z_1, z_2 \in (N \cup T)^*$. Použitím operácie[11] dostávame nový obsah zásobníka, pre ktorý platí, že otočenie zásobníka je $\downarrow$ a vetná forma, ktorú predstavuje zásobník, je $\overline{x_\downarrow}^R \underline{x_\downarrow}^R = (x^R)^R (y^R \xi)^R = x\xi y$. Použitím pravidla, ktoré je v argumente tejto operácie, odvodíme $x\xi y \Rightarrow xz_1 ® z_2 y \Rightarrow^* w'$, kde vetná forma v strede je práve $\underline{x_\uparrow}\overline{x_\uparrow}$ z predpokladu, teda obsah zásobníka pred operáciou. Z predpokladu platnosti invariantu *I1* pre toto w' platí, že $\varrho(w')$ je slovo w. Na vrchu zásobníka pre $\underline{x_\downarrow} = y^R \xi$ sa nachádza neterminál ξ, pre ktorý z predpokladu operácie platí, že generuje stredný reverz. Po použití tejto operácie je v platnosti invariant *I4*.

V šiestom kroku algoritmu nie je platný tento invariant, takže táto kombinácia invariantu a operácie nemôže nastať.

Platí invariant *I2*

POSUŇ Podobne ako v predchádzajúcom prípade pre invariant *I1*, operácia vloží na vrch zásobníka symbol $c \in T$ zo vstupu. Pre obsah zásobníka nad zarážkou potom platí, že ak z predpokladu platnosti invariantu *I2* vieme odvodiť $(\underline{x_\downarrow})^R \Rightarrow^* (u_2 ® u_4 ® \ldots ® u'_{k+1})^R$, tak vieme odvodiť $(\underline{x_\downarrow} c)^R \Rightarrow^* (u_2 ® u_4 ® \ldots ® u'_{k+1} c)^R$ pre rovnakú postupnosť použitých pravidiel. Obsah zásobníka pod zarážkou ani počet symbolov ®, ktoré možno odvodiť z obsahu zásobníka nad a pod zarážkou, sa nezmenia. Zároveň, ak sme v predpoklade platnosti invariantu *I2* vedeli odvodiť slovo w použitím funkcie ϱ v tvare $u_1 u_2 \ldots u_k u'_{k+1}$, tak pridaním znaku c za slovo u'_{k+1} dostaneme $u_1 u_2 \ldots u_k u'_{k+1} c$, čo zodpovedá doposiaľ prečítanému vstupu wc. Platí invariant *I2*.

[11] Počas operácie sa z vrchu zásobníka odstráni podslovo z_1 a zvyšok slova sa nechá na zásobníku. Potom sa zásobník otočí. Nový obsah zásobníka pod zarážkou je $\overline{x_\downarrow} = x^R$ a nad zarážkou je $\underline{x_\uparrow} = y^R x_2^R ®$. Potom operácia z vrchu zásobníka vytiahne slovo $x_2^R ®$ a na vrch vloží neterminál ξ.

OTOČ Podobne ako v predchádzajúcom prípade pre invariant *I1*, operácia OTOČ zvýši počet otočení k, tým sa vyrovná počet reverzov na vrchu aj spodku zásobníka a do platnosti príde invariant *I1*. Rovnako možno tvrdenie aj dokázať.

REDUKUJ Znova rozdelíme použitie tohto pravidla podľa toho, v ktorom kroku algoritmu bolo použité.

V druhom kroku algoritmu vieme z predpokladu použitia operácie zapísať vrch zásobníka v tvare $\underline{x_\uparrow} = yz^R$ pre nejaké $y, z \in (N \cup T)^*$. Použitím operácie dostaneme na vrchu zásobníka slovo $y\xi$, z ktorého na jeden krok odvodenia v gramatike s použitím pravidla $\xi \to z$ vieme dostať $(y\xi)^R \Rightarrow (yz^R)^R$, respektíve $\xi y^R \Rightarrow zy^R$. Z nového obsahu zásobníka nad zarážkou po použití tejto operácie vieme odvodiť $(\underline{x_\uparrow})^R = (y\xi)^R \Rightarrow (yz^R)^R \Rightarrow^* (u_2 ® u_4 ® \ldots ® u'_{k+1})^R$, kde vychádzame z predpokladu platnosti invariantu *I2* na odvodenie tohto slova. Potom slovo odvoditeľné z vrchnej aj spodnej časti zásobníka ostáva rovnaké, rovnako aj počet symbolov ® v oboch častiach. Ostáva v platnosti invariant *I2*.

Vo štvrtom kroku algoritmus použil pravidlo v tvare $\xi \to z_1 ® z_2$, kde tento ® je stredný. Slovo nad zarážkou môžeme zapísať v tvare $\underline{x_\downarrow} = xz_2^R$ a slovo pod zarážkou v tvare $\overline{x_\downarrow} = ®z_1^R y$ pre nejaké $x, y, z_1, z_2 \in (N \cup T)^*$, čo je v súlade s predpokladom platnosti invariantu *I2*. Po použití operácie bude nový obsah zásobníka $\underline{x_\uparrow} = y^R\xi$ a $\overline{x_\uparrow} = x^R$. Potom nový obsah zásobníka predstavuje vetnú formu $\underline{x_\uparrow}\overline{x_\uparrow} = y^R\xi x^R$, z ktorej odvodením dostávame $y^R\xi x^R \Rightarrow y^R z_1 ® z_2 x^R \Rightarrow w'$, kde vetná forma v strede je $\overline{x_\downarrow}^R \underline{x_\downarrow}^R$ z predpokladu platnosti invariantu *I2*. Takto dostaneme odvodenie slova w', pre ktoré platí $\varrho(w') = w$. Nové otočenie zásobníka je ↑. Na vrchu zásobníka je $\xi \in N_A$. Potom platí invariant *I3*.

Platí invariant *I3*

REDUKUJ Pre tento invariant je relevantná operácia REDUKUJ iba v kroku 6. Pravidlo, ktoré použijeme, je v tvare $\xi \to z_1\gamma z_2$, kde $\xi, \gamma \in N_A$ sú neterminály generujúce stredný reverz a $z_1, z_2 \in (N_B \cup T)^*$. Z predpokladu, že môžeme použiť toto pravidlo, vieme rozdeliť slovo nad zarážkou $\underline{x_\uparrow} = xz_1\gamma$ a slovo pod zarážkou $\overline{x_\uparrow} = z_2 y$. Z predpokladu platnosti tohto invariantu vieme v gramatike odvodiť $\underline{x_\uparrow}\overline{x_\uparrow} \Rightarrow^* w'$, kde toto w' je v správnom tvare a vieme z neho pomocou funkcie ϱ odvodiť prečítanú časť vstupu w. Otočenie zásobníka je ↑. Po použití operácie bude na zásobníku nad zarážkou slovo $\underline{x_\downarrow} = y^R\xi$ a pod zarážkou slovo $\overline{x_\downarrow} = x^R$. Otočenie zásobníka je po použití operácie ↓ a na vrchu zásobníka je $\xi \in N_A$, ktorý generuje stredný reverz. Vetná forma, ktorú predstavuje nový obsah zásobníka, je $\overline{x_\downarrow}^R \underline{x_\downarrow}^R = (x^R)^R (y^R\xi)^R = x\xi y$. Jedným krokom odvodenia, kde použijeme pravidlo, podľa ktorého sme v tejto operácii redukovali, dostaneme $x\xi y \Rightarrow xz_1\gamma z_2 y$,

čo je $\underline{x_{\uparrow}}\overline{x_{\uparrow}}$ z predpokladu. Potom platí $\overline{x_{\downarrow}}^R\underline{x_{\downarrow}}^R \Rightarrow xz_1\gamma z_2 y \Rightarrow^* w'$ pre rovnaké w' ako v predpoklade platnosti invariantu *I3*. Po použití tejto operácie platí invariant *I4*.

Platí invariant *I4*

REDUKUJ Pre tento invariant je opäť možné použiť operáciu REDUKUJ iba v kroku 6. Nech z predpokladu použitia operácie je pravidlo v tvare $\xi \to z_1\gamma z_2$, kde $\xi, \gamma \in N_A$. Obsah zásobníka nad zarážkou vieme zapísať v tvare $\underline{x_{\downarrow}} = y^R z_2^R \gamma$ a pod zarážkou v tvare $\overline{x_{\downarrow}} = z_1^R x^R$. Z platnosti invariantu *I4* je otočenie zásobníka $\downarrow$ a zo slov na zásobníku vieme odvodiť $\overline{x_{\downarrow}}^R\underline{x_{\downarrow}}^R = (z_1^R x^R)^R(y^R z_2^R\gamma)^R = xz_1\gamma z_2 y \Rightarrow^* w'$, kde pre toto slovo platí, že $\varrho(w')$ je vstupné slovo w. Po použití operácie je otočenie zásobníka $\uparrow$. Obsah zásobníka nad zarážkou je $\underline{x_{\uparrow}} = x\xi$ a obsah zásobníka pod zarážkou je $\overline{x_{\uparrow}} = y$. Vetná forma, ktorú predstavuje takto upravený zásobník, je $\underline{x_{\uparrow}}\overline{x_{\uparrow}} = x\xi y$. Použitím pravidla $\xi \to z_1\gamma z_2$ z operácie na túto vetnú formu dostaneme $x\xi y \Rightarrow xz_1\gamma z_2 y$. S využitím odvodenia z predpokladu platnosti invariantu *I4* potom platí aj $\underline{x_{\uparrow}}\overline{x_{\uparrow}} \Rightarrow^* w'$, pre ktoré platia rovnaké podmienky, ako sú v predpoklade platnosti tohto invariantu *I4*. Na vrchu zásobníka je neterminál $\xi \in N_A$. Potom platí invariant *I3*.

Takto sme ukázali, že pre ľubovoľný výpočet v algoritme je splnený aspoň jeden z invariantov *I1* až *I4*. □

Tvrdenie 2.3.4. *Nech $G = (N, T, P, \sigma, ®)$ je bezkontextová gramatika s generovaním reverzov v normálnom tvare so znalosťou generovaných reverzov (definícia 1.1.8). Ku každému akceptačnému výpočtu podľa nedeterministického algoritmu „posuň, redukuj, otoč“ z oddielu 2.1 na vstupnom slove w a vstupnej gramatike G existuje odvodenie nejakého slova w' v tejto gramatike G, pre ktoré platí, že $\varrho(w')$ je vstupné slovo w.*

Dôkaz. Z tvrdenia 2.3.3 vieme, že algoritmus spĺňa jeden z invariantov *I1* až *I4* v každom kroku algoritmu. Preto môžeme predpokladať ich platnosť aj v každom kroku akceptačného výpočtu.

Ak algoritmus akceptuje výpočet v kroku 3, tak platí invariant *I1* alebo *I2*. Keďže algoritmus nepoužil operáciu REDUKUJ na žiadne pravidlo z množiny P_A, tak gramatika negeneruje stredné ®, a teda negeneruje žiadne symboly ®. Počas behu algoritmu nebola použitá žiadna operácia OTOČ. Algoritmus beží rovnako ako známy algoritmus „posuň, redukuj“ pre štandardné bezkontextové gramatiky.

Ak algoritmus akceptuje výpočet v kroku 5, tak spĺňa niektorý z invariantov *I3* alebo *I4*. Na zásobníku je podľa akceptačnej podmienky z kroku 5 iba neterminál σ. Potom je možné vygenerovať vstupné slovo w vo vstupnej gramatike G z tohto neterminálu podľa predpokladu platnosti niektorého invariantu. O neterminále σ navyše

z predpokladu normálneho tvaru pre gramatiku vieme, že generuje práve reverzy 1 až n, teda $\phi(\sigma) = \{1, \ldots, n\}$. Potom algoritmus akceptuje iba slová, ktoré počas behu algoritmu na zásobník pridali práve n symbolov reverzu pomocou operácie OTOČ, teda práve vtedy, keď počet otočení zásobníka k je rovné n, kde n je počet generovaných reverzov pre gramatiku G na vstupe algoritmu. □

2.3.2 Akceptačný výpočet k normálnemu odvodeniu

Nech gramatika $G = (N, T, P, \sigma, ®)$ je v normálnom tvare so znalosťou generovaných reverzov. Nech $\sigma \Rightarrow^* w'$ je normálne odvodenie slova w' v gramatike G také, že slovo $w := \varrho(w')$ patrí do jazyka $L(G)$. V normálnom odvodení z definície 2.2.1 platí, že reverzy sú generované v poradí od A_1-vého po A_n-tý, kde pre $k = 1, \ldots, n$ platí

$$A_k = \left\lceil \frac{n+1}{2} \right\rceil + (-1)^{n+k+1} \left\lfloor \frac{k}{2} \right\rfloor.$$

V nasledujúcom ukážeme, že pomocou nedeterministického algoritmu „posuň, redukuj, otoč“ z oddielu 2.1 možno na vstupe $w \in L(G)$ od konca simulovať takéto normálne odvodenie v gramatike G. Ku každému kroku odvodenia slova $w' \in L_{CF}(G)$ v gramatike G podľa pravidla $\xi \to x$ bude v tejto simulácii zodpovedať postupnosť operácií z algoritmu pozostávajúca z niekoľkých operácií POSUŇ, niekoľkých operácií OTOČ a práve jednej operácie REDUKUJ $\xi \to x$.

Neformálne opíšeme konštrukciu akceptačného výpočtu prislúchajúceho k danému odvodeniu. Nižšie ju opíšeme formálne.

Ak gramatika generuje stredný reverz, ktorý je A_1-vý v postupnosti (podľa prípadu (a, i) v normálnom odvodení), tak k jednému kroku odvodenia prislúcha práve jedna operácia REDUKUJ $\xi \to w$ podľa pravidla, ktoré sa pri odvodení použilo, kde toto pravidlo je z P_A. Táto situácia môže nastať v akceptačnom výpočte algoritmu „posuň, redukuj, otoč“ až v druhej časti, počas vykonávania krokov 4 až 6 v tomto algoritme.

Ak gramatika prepisuje pravidlo naľavo od stredného reverzu, tak je zásobník otočený ↑ a použitie operácie POSUŇ a REDUKUJ je rovnaké ako v štandardnom algoritme „posuň, redukuj“. Rovnako, ako v štandardnom algoritme, aj tu sa generuje pravé krajné odvodenie.

Ak gramatika prepisuje pravidlo napravo od stredného reverzu, tak k tomuto kroku odvodenia prislúcha otočenie zásobníka ↓. Symboly z vetnej formy sú na zásobníku uložené v reverznom poradí tak, že na vrchu zásobníka je nejaký sufix vetnej formy, kde posledný znak tohto sufixu je pri znaku zarážky ||. Na vrchu zásobníka pri otočení zásobníka ↓ sa tvorí ľavé krajné odvodenie (reverzne k pravému krajnému). Pre každé pravidlo $\xi \to x$, pre ktoré sa v tejto fáze vykonala operácia REDUKUJ $\xi \to x$, sa nachádza reverz jeho pravej strany x^R na vrchu zásobníka.

Pravé krajné odvodenie v podslove, handle, životaschopný prefix

Podobne, ako pre štandardné bezkontextové gramatiky, zavedieme označenie pre reláciu kroku pravého krajného a ľavého krajného odvodenia pre bezkontextové gramatiky s generovaním reverzov. Následne definujeme obdoby pojmov *handle*[12] a *životaschopný prefix* pre bezkontextové gramatiky s generovaním reverzov.

Označenie 2.3.5. Nech $G = (N, T, P, \sigma, ®)$ je *RGCFG* v normálnom tvare so znalosťou generovaných reverzov. Symbolom $\Rightarrow_{A_1}$ označujeme reláciu kroku odvodenia, pri ktorom sa prepíše stredový neterminál v gramatike G. Pre $u, v \in (N \cup T)^*$ teda platí $u \Rightarrow_{A_1} v$ práve vtedy, keď pre nejaké $u_1, u_2, x \in (N \cup T)^*$ a $\xi \in N_A$, pre ktoré $A_1 \in \phi(\xi)$, platí $u = u_1\xi u_2$, $v = u_1 x u_2$ a $\xi \to x \in P_A$.[13]

Symbolom $\Rightarrow_{RM}$ označujeme reláciu *kroku pravého krajného odvodenia* v gramatike G. Pre $u, v \in (N \cup T)^*$ platí $u \Rightarrow_{RM} v$ práve vtedy, keď pre nejaké $u_1, x \in (N \cup T)^*$, $u_2 \in T^*$ a $\xi \in N$ platí $u = u_1\xi u_2$, $v = u_1 x u_2$ a $\xi \to x \in P$.

Symbolom $\Rightarrow_{LM}$ označujeme reláciu *kroku ľavého krajného odvodenia* v gramatike G. Pre $u, v \in (N \cup T)^*$ platí $u \Rightarrow_{LM} v$ práve vtedy, keď pre nejaké $u_1 \in T^*$, $u_2, x \in (N \cup T)^*$ a $\xi \in N$ platí $u = u_1\xi u_2$ a $v = u_1 x u_2$.

Symoblom $\Rightarrow_{norm}$ označujeme jeden krok odvodenia podľa normálneho odvodenia v gramatike G z definície 2.2.1.

Definícia 2.3.6 (Handle)**.** Nech $v = v_1 x v_2$, kde $v_1, v_2, x \in (N \cup T)^*$, je vetná forma v gramatike $G = (N, T, P, \sigma, ®)$.

Podslovo x vetnej formy v je *stredová handle*, ak existuje $\xi \in N_A$ a pravidlo $\xi \to x \in P_A$ také, že $\sigma \Rightarrow^*_{norm} v_1\xi v_2 \Rightarrow_{A_1} v_1 x v_2$. V takom prípade budeme hovoriť, že x je stredová handle vo v pre toto odvodenie $\sigma \Rightarrow^*_{norm} v$.

Podslovo x vetnej formy v je *ľavá handle*, ak existuje $v_2' \in T^*$, $v_2'' \in (N \cup T)^*$, $\xi \in N_B$ a pravidlo $\xi \to x \in P_B$ také, že $v_2 = v_2' ® v_2''$, kde reverz medzi v_2' a v_2'' je stredný vo v a pre nejaké $y_1, \ldots, y_s \in (N \cup T)^*$ platí

$$\begin{aligned}\sigma \Rightarrow^*_{norm} v_1\xi v_2' ® y_1 \Rightarrow_{norm} v_1 x v_2' ® y_1 \Rightarrow_{norm} \\ \Rightarrow_{norm} v_1 x v_2' ® y_2 \Rightarrow_{norm} \ldots \Rightarrow_{norm} v_1 x v_2' ® y_s = v,\end{aligned}$$

$y_s = v_2''$ a $y_1 \Rightarrow_{LM} y_2 \Rightarrow_{LM} \ldots \Rightarrow_{LM} y_s$, kde $s \in \mathbb{N} - \{0\}$. V takom prípade tiež budeme hovoriť, že x je ľavá handle vo v pre toto odvodenie $\sigma \Rightarrow^*_{norm} v$.

Podslovo x vetnej formy v je *pravá handle*, ak existuje $v_1' \in (N \cup T)^*$, $v_1'' \in T^*$, $\xi \in N_B$ a pravidlo $\xi \to x \in P_B$ také, že $v_1 = v_1' ® v_1''$, kde reverz medzi v_1' a v_1'' je

[12] Slovenský ekvivalent slova *rukoväť* sa nezaužíval.

[13] V každej vetnej forme je práve jeden neterminál generujúci stredný reverz.

stredný vo v, a pre nejaké $y_1, \ldots, y_s \in (N \cup T)^*$ platí

$$\begin{aligned}\sigma \Rightarrow^*_{norm} y_1 Ⓡ v_1'' \xi v_2 \Rightarrow_{norm} y_1 Ⓡ v_1'' x v_2 \Rightarrow_{norm} \\ \Rightarrow_{norm} y_2 Ⓡ v_1'' x v_2 \Rightarrow_{norm} \ldots \Rightarrow_{norm} y_s Ⓡ v_1'' x v_2 = v,\end{aligned}$$

$y_s = v_1'$ a $y_1 \Rightarrow_{RM} y_2 \Rightarrow_{RM} \ldots \Rightarrow_{RM} y_s$, kde $s \in \mathbb{N} - \{0\}$. V takom prípade budeme hovoriť, že x je pravá handle vo v pre toto odvodenie $\sigma \Rightarrow^*_{norm} v$.

Neformálne, stredová handle označuje výskyt pravej strany pravidla vo vetnej forme, kde sa prepísal stredový neterminál. Ľavá handle označuje výskyt pravej strany pravidla, ktoré sa prepísalo ako posledné vo vetnej forme v ľavom podslove od stredného reverzu, pričom táto ľavá časť vetnej formy vznikla pravým krajným odvodením v podslove od stredného reverzu. Pravá handle označuje výskyt pravej strany pravidla, ktoré sa prepísalo ako posledné vo vetnej forme v pravom podslove od stredného reverzu, pričom v pravom podslove od stredného reverzu pre každú vetnú formu v tomto odvodení bolo použité ľavé krajné odvodenie.

Teraz zadefinujeme životaschopné prefixy v bezkontextových gramatikách s generovaním reverzov. Nepôjde o prefixy v pravom zmysle slova, ale o časti vetnej formy, z ktorých možno odvodením terminálneho slova a následne aplikovaním funkcie ϱ odvodiť prečítaný prefix vstupného slova. Tieto časti vetnej formy sú reprezentované vhodne interpretovaným obsahom zásobníka počas behu nedeterministického algoritmu. Hlavnou podmienkou životaschopnosti je možnosť pokračovať vo výpočte algoritmu až po redukciu na počiatočný neterminál σ. Formalizácia tejto podmienky pomocou handle bude podobná ako v štandardných bezkontextových gramatikách.

Definícia 2.3.7 (Životaschopný prefix)**.** Nech $G = (N, T, P, \sigma, Ⓡ)$ je v normálnom tvare so znalosťou generovaných reverzov.

i) Nech $\sigma \Rightarrow^*_{norm} u$ je pevne dané odvodenie slova $u = u_1 Ⓡ u_2$,

$$\begin{aligned}u_1 &\in \Big(T \cup \big\{\eta \in N \mid \phi(\eta) \subseteq \{1, \ldots, A_1 - 1\}\big\}\Big)^*, \\ u_2 &\in \Big(T \cup \big\{\eta \in N \mid \phi(\eta) \subseteq \{A_1 + 1, \ldots, n\}\big\}\Big)^*\end{aligned}$$

a reverz medzi u_1 a u_2 je stredný. Nech x_1 je prefix u_1, x_2 je sufix u_2 a nech pre toto odvodenie existuje aspoň jedna ľavá alebo pravá handle v u. Zároveň platí, že všetky symboly vo vetnej forme u medzi slovami x_1 a x_2 sú terminálne ($u = x_1 u' x_2$, kde $u' \in T^*$).

Dvojica $(\underline{x_1}, x_2)$ je *životaschopný prefix*[14] pre vetnú formu u a odvodenie $\sigma \Rightarrow^*_{norm} u$, ak x_1 siaha najviac po koniec ľavej handle pre odvodenie $\sigma \Rightarrow^*_{norm} u$ vo vetnej forme u (ak nejaká existuje), x_2 siaha najviac po začiatok pravej handle

[14] Nejde o prefix slova u v pravom zmysle.

pre odvodenie $\sigma \Rightarrow^*_{norm} u$ vo vetnej forme u (ak nejaká existuje) a počet reverzov v každom terminálnom slove vygenerovanom z x_1 je rovnaký ako pre x_2. Ak navyše x_1 siaha práve po koniec ľavej handle pre toto odvodenie v u, tak hovoríme o *úplnom životaschopnom prefixe*.

Dvojica $(x_1, \underline{x_2})$ je *životaschopný prefix* pre vetnú formu u a odvodenie $\sigma \Rightarrow^*_{norm} u$, ak x_1 siaha najviac po koniec ľavej handle pre odvodenie $\sigma \Rightarrow^*_{norm} u$ vo vetnej forme u (ak nejaká existuje), x_2 siaha najviac po začiatok pravej handle pre odvodenie $\sigma \Rightarrow^*_{norm} u$ vo vetnej forme u (ak nejaká existuje) a počet reverzov v každom terminálnom slove vygenerovanom z x_1 je o jedna väčší ako pre x_2. Ak navyše x_2 siaha práve po začiatok pravej handle pre toto odvodenie v u, tak hovoríme o *úplnom životaschopnom prefixe*.

ii) Nech $\sigma \Rightarrow^*_{A_1} u$ je pevne dané odvodenie slova u. Potom ľubovoľná dvojica $(\underline{x_1}, x_2)$ alebo $(x_1, \underline{x_2})$, kde $u = u_1u_2$, x_1 je prefix u_1, x_2 je sufix u_2 a existujú $u'_1, u''_1, u'_2, u''_2 \in (N \cup T)^*$ také, že $u_1 = u'_1u''_1$, $u_2 = u'_2u''_2$ a $u''_1u'_2$ je stredová handle pre odvodenie $\sigma \Rightarrow^*_{A_1} u$ vo vetnej forme u, je *životaschopný prefix* pre vetnú formu u a odvodenie $\sigma \Rightarrow^*_{A_1} u$. Zároveň platí, že všetky symboly vo vetnej forme u medzi slovami x_1 a x_2 sú terminálne ($u = x_1u'x_2$, kde $u' \in T^*$). Ak navyše $x_1 = u_1$ a $x_2 = u_2$, tak hovoríme o *úplnom životaschopnom prefixe*.

Podčiarknutie v životaschopnom prefixe intuitívne interpretujeme ako otočenie zásobníka. Ak je zásobník otočený ↑, tak nás budú zaujímať životaschopné prefixy, kde je podčiarknuté prvé slovo. V opačnom prípade nás budú zaujímať životaschopné prefixy s podčiarknutým druhým slovom.

V nasledujúcom opíšeme, ako bude vyzerať akceptačný výpočet nedeterministického algoritmu „posuň, redukuj, otoč“, ktorý zodpovedá danému normálnemu odvodeniu v gramatike. Každému kroku odvodenia podľa pravidla $\xi \to x$ v algoritme zodpovedá niekoľko použití operácie POSUŇ, niekoľko použití operácie OTOČ a práve jedno použitie operácie REDUKUJ $\xi \to x$.

Konštrukcia akceptačného výpočtu k normálnemu odvodeniu

Na začiatku algoritmu vykonáme inicializáciu algoritmu „posuň, otoč, redukuj“, ktorá spočíva vo vložení symbolu || na spodok zásobníka. Uvažujme normálne odvodenie $\sigma = u_0 \Rightarrow_{norm} u_1 \Rightarrow_{norm} \ldots \Rightarrow_{norm} u_n = w'$, kde $w' \in T^*$. Prechádzame toto odvodenie od konca. Pre každý krok tohto odvodenia $u_{i-1} \Rightarrow_{norm} u_i$ pre $i = n, \ldots, 1$, kde v tomto kroku bolo použité pravidlo $\xi \to x$, opakujeme:

0. Pre otočenie zásobníka ↑, nech obsah zásobníka nad zarážkou je slovo x_1 a pod zarážkou je slovo x_2. Pre otočenie zásobníka ↓, nech obsah zásobníka nad zarážkou je slovo x_2^R a pod zarážkou je slovo x_1^R.

1. Ak sa v tomto kroku prepisuje stredový neterminál a ak $(\underline{x_1}, x_2)$ alebo $(x_1, \underline{x_2})$ je úplný životaschopný prefix pre vetnú formu u_i a uvažované odvodenie, tak REDUKUJ $\xi \to x$, kde toto pravidlo je z P_A.
   Koniec spracovania tohto kroku odvodenia. Posuň sa na predchádzajúci krok odvodenia.

2. Ak $(\underline{x_1}, x_2)$ je úplný životaschopný prefix pre vetnú formu u_i a uvažované odvodenie a otočenie zásobníka je $\uparrow$ alebo $(x_1, \underline{x_2})$ je úplný životaschopný prefix pre vetnú formu u_i a uvažované odvodenie a otočenie zásobníka je $\downarrow$, tak REDUKUJ $\xi \to x$, kde toto pravidlo je z P_B.
   Koniec spracovania tohto kroku odvodenia. Posuň sa na predchádzajúci krok odvodenia.

3. Ak nemožno redukovať a vo vetnej forme u_i platí, že za prefixom x_1 je bezprostredne nasledujúci symbol ®, $(\underline{x_1}, x_2)$ je životaschopný prefix a otočenie zásobníka je $\uparrow$ alebo bezprostredne pred sufixom x_2 je vo vetnej forme u_i symbol ®, $(x_1, \underline{x_2})$ je životaschopný prefix a otočenie zásobníka je $\downarrow$, tak OTOČ.

4. Inak POSUŇ.

5. Opakuj celý postup pre ten istý krok odvodenia a nový životaschopný prefix.

Tvrdenie 2.3.8. *K normálnemu odvodeniu (definícia 2.2.1) slova w v bezkontextovej gramatike s generovaním reverzov G existuje akceptačný výpočet podľa nedeterministického algoritmu „posuň, redukuj, otoč“ na vstupe pozostávajúcom zo slova w a gramatiky G.*

Dôkaz. Nech gramatika G v normálnom tvare so znalosťou generovaných reverzov (definícia 1.1.8) generuje n reverzov.[15] Pre ľubovoľné slovo $w' \in L_{CF}(G)$ existuje v gramatike G normálne odvodenie $\sigma \Rightarrow^*_{norm} w'$ z definície 2.2.1. Vyhodnotením symbolov reverzu v tomto slove dostávame slovo $w := \varrho(w')$, ktoré patrí do jazyka $L(G)$ pre túto gramatiku. Podľa vyššie opísanej konštrukcie zostrojíme k odvodeniu $\sigma \Rightarrow^*_{norm} w'$ postupnosť krokov algoritmu pre toto odvodenie. Ukážeme, že pre každý krok tohto odvodenia vieme podľa tejto konštrukcie udržiavať na zásobníku životaschopný prefix pre toto odvodenie. Cieľom konštrukcie je zostrojiť akceptačný výpočet na vstupnom slove w a vstupnej gramatike G vtedy, keď $w \in L(G)$.

Na vstupe algoritmu je slovo w a gramatika G. Normálne odvodenie $\sigma = u_0 \Rightarrow_{norm} u_1 \Rightarrow_{norm} \ldots \Rightarrow_{norm} u_n = w'$, kde $w' \in T^*$, prechádzame od konca. Vetná forma, ktorá je na konci odvodenia (respektíve začiatku výpočtu), je terminálne slovo w', pre

[15] V tomto dôkaze predpokladáme $n \geq 1$. Pre 0 generovaných reverzov možno ľahko vidieť, že algoritmus pracuje rovnako ako štandardný algoritmus „posuň, redukuj“. Konštruuje sa v takom prípade pravé krajné odvodenie na vrchu zásobníka bez otočenia zásobníka.

ktoré platí, že $\varrho(w') = w$ je práve vstupné slovo pre algoritmus. Na začiatku výpočtu je otočenie zásobníka $\uparrow$ a životaschopný prefix, ktorý zodpovedá obsahu zásobníka, je $(\underline{\varepsilon}, \varepsilon)$ pre vetnú formu w' a toto odvodenie.[16] Pre každý krok odvodenia $u_{i-1} \Rightarrow_{norm} u_i$ predpokladáme, že $(\underline{x_1}, x_2)$ je životaschopný prefix pre vetnú formu u_i a toto odvodenie.[17] Ukážeme, že po vykonaní niektorého z krokov 1 až 4 podľa vyššie opísanej konštrukcie na zásobníku opäť vznikne životaschopný prefix pre nejakú vetnú formu (u_{i-1} alebo u_i) a toto odvodenie.

1. REDUKUJ $\xi \to x$ pre pravidlo z P_A

   $(\underline{x_1}, x_2)$ je úplný životaschopný prefix pre u_i, kde $u_i = x_1 x_2$. V tomto kroku ide o krok odvodenia stredového neterminálu $\Rightarrow_{A_1}$. Vetná forma u_i obsahuje stredovú handle x pre toto odvodenie a posledný krok tohto odvodenia možno zapísať v tvare

   $$u_i' \xi u_i'' \Rightarrow_{A_1} u_i' x u_i'',$$

   kde $u_{i-1} = u_i' \xi u_i''$ a $u_i = u_i' x u_i''$. Po tomto kroku môže nastať jeden z nasledujúcich prípadov:

   - Ak $i = 1$, tak $\xi = \sigma$ a dostali sme akceptačný výpočet.
   - Ak $i > 1$, tak existuje predchádzajúci krok $u_{i-2} \Rightarrow_{norm} u_{i-1}$ v tomto odvodení, kde bolo použité nejaké pravidlo podľa normálneho odvodenia. Keďže po redukcii je prefix $(u_i', \underline{\xi u_i''})$ a je to zároveň celá vetná forma a krok odvodenia $u_{i-2} \Rightarrow_{norm} u_{i-1}$ musí byť podľa A_1 (lebo aj tento bol podľa A_1), tak potom tento prefix musí byť úplný životaschopný prefix pre u_{i-1}.

2. REDUKUJ $\xi \to x$ pre pravidlo z P_B

   Z predpokladu, že u_i obsahuje ľavú handle, vieme zapísať vetnú formu v tvare

   $$u_i = u_i' \text{Ⓡ} u_i'',$$

   kde tento Ⓡ je stredný. Keďže životaschopný prefix $(\underline{x_1}, x_2)$ je úplný a obsahuje celú ľavú handle, tak

   $$u_i = v_1 x v_2 \text{Ⓡ} u_i'', \qquad \text{kde } v_1 \in (N \cup T)^*, v_2 \in T^*,$$
   $$x_1 = v_1 x$$

   a x_2 je sufix u_i'', ktorý siaha najviac po začiatok pravej handle (ak nejaká existuje).

[16] Zrejme sú splnené všetky podmienky pre platnosť tohto životaschopného prefixu pre prípad (i) alebo prípad (ii) z definície 2.3.7.

[17] Rovnako by sme konštruovali postupnosť krokov algoritmu pre životaschopný prefix $(x_1, \underline{x_2})$ vo vetnej forme u_i a toto odvodenie.

Aplikáciou predpokladu existencie ľavej handle[18] na krok odvodenia dostávame:

$$u_{i-1} \Rightarrow_{norm} u_i,$$
$$v_1 \xi v_2 \circledR u_i'' \Rightarrow_{norm} v_1 x v_2 \circledR u_i''.$$

Teraz potrebujeme overiť, či je dvojica $(\underline{v_1\xi}, x_2)$ životaschopný prefix. Môže nastať jeden z nasledujúcich prípadov:

- Nová vetná forma obsahuje novú ľavú handle.

  Nájdeme posledný predchádzajúci krok odvodenia v ľavom podslove od stredného reverzu:

  $$\sigma \Rightarrow^*_{norm} z \circledR y_1 \Rightarrow_{norm} v_1 \xi v_2 \circledR y_1 \Rightarrow_{norm}$$
  $$\Rightarrow_{norm} v_1 \xi v_2 \circledR y_2 \Rightarrow_{norm} \ldots \Rightarrow_{norm} v_1 \xi v_2 \circledR y_s = u_{i-1},$$

  $y_s = u_i''$ a $y_1 \Rightarrow_{LM} y_2 \Rightarrow_{LM} \ldots \Rightarrow_{LM} y_s$, kde $s \in \mathbb{N} - \{0\}$ pre nejaké $z, y_1, \ldots, y_s \in (N \cup T)^*$. Posledný krok odvodenia, kde bolo použité pravidlo v ľavej časti od stredného ®, je krok odvodenia $z \circledR y_1 \Rightarrow_{norm} v_1 \xi v_2 \circledR y_1$. Sporom dokážeme, že $v_1\xi$ je prefix nejakej ľavej handle v tomto odvodení:

  ak by bola pravá strana prepísaného pravidla v tomto kroku podslovo slova v_1, tak pre krok tohto odvodenia by platilo

  $$z_1 \xi_1 z_2 \xi v_2 \circledR y_1 \Rightarrow_{norm} z_1 x' z_2 \xi v_2 \circledR y_1$$

  pre nejaké $z_1, z_2 \in (N \cup T)^*$ a $\xi_1 \in N_B$. Čo je ale v spore s normálnym odvodením, kde sa vo vetnej forme $z_1\xi_1 z_2 \xi v_2 \circledR y_1$ prepíše podľa pravého krajného odvodenia najprv neterminál ξ.

  Z uvedenej argumentácie vieme, že $v_1\xi$ je prefix ľavej handle vo vetnej forme u_{i-1} pre toto odvodenie. Z predpokladu vieme, že x_2 siaha najviac po začiatok pravej handle pre toto odvodenie, ak taká existuje. Potom dvojica $(\underline{v_1\xi}, x_2)$ je životaschopným prefixom pre vetnú formu u_{i-1} a toto odvodenie.

- Nová vetná forma neobsahuje ľavú handle, ale obsahuje sufix pravej handle v slove x_2.

  Pôvodná vetná forma u_i obsahovala aj pravú handle a slovo x_2 sa nezmenilo. Pre slovo x_1 platí, že preň neexistuje ľavá handle, ale stále je prefixom slova u_i' a $v_1\xi$ je prefixom vetnej formy u_{i-1}. Dvojica $(\underline{v_1\xi}, x_2)$ je životaschopným prefixom pre vetnú formu u_{i-1} a toto odvodenie.

- Nová vetná forma neobsahuje ani ľavú ani pravú handle.

[18] Pre $s = 0$ z definície, keďže posledný krok sa vykonal v ľavom podslove x_1 od stredného reverzu.

Vetná forma neobsahovala pravú handle a x_2 je nejaký sufix slova u_i''. Ľavá handle neexistuje a x_1 je prefixom vetnej formy u_{i-1}. Z toho vyplýva, že predošlý krok odvodenia je podľa A_1[19] nasledovný $u_{i-2} \Rightarrow_{A1} u_{i-1}$. Potom sa vo vetnej forme u_{i-1} musí nachádzať stredová handle pre tento krok odvodenia. Dvojica ($\underline{v_1\xi}$, x_2) je životaschopným prefixom pre vetnú formu u_{i+1} a toto odvodenie tak, ako v prípade (ii) z definície o životaschopnom prefixe.

3. OTOČ

   Keďže vetná forma nebola redukovaná v niektorom z predošlých krokov, tak životaschopný prefix $(\underline{x_1}, x_2)$ vetnej formy u_i pre toto odvodenie nie je úplný. Vetnú formu vieme zapísať v tvare $u_i = x_1 ® u_i' x_2$ pre $u_i' \in T^*$. Keďže životaschopný prefix nie je úplný pre ľavú handle ani stredovú handle a vo vetnej forme nasleduje symbol ®, tak môžeme vykonať operáciu OTOČ, kde sa z dvojice $(\underline{x_1}, x_2)$ na zásobníku stane dvojica $(x_1®, \underline{x_2})$. Z predpokladu platí, že počet reverzov, ktoré možno vygenerovať zo slov x_1 a x_2, je rovnaký. Potom zo slova $x_1®$ možno vygenerovať o jeden symbol reverzu viac ako zo slova x_2. Takto dostávame nový životaschopný prefix $(x_1®, \underline{x_2})$ pre pôvodnú vetnú formu u_i v tomto odvodení.

4. POSUŇ (a zároveň životaschopný prefix $(\underline{x_1}, x_2)$ nie je úplný pre vetnú formu u_i a toto odvodenie a za slovom x_1 vo vetnej forme u_i nie je ďalší znak ®)

   Keďže životaschopný prefix nie je úplný a za slovom x_1 musí byť nejaký terminálny symbol iný ako symbol reverzu, tak túto vetnú formu možno zapísať v tvare $u_i = x_1 c u_i' x_2$, kde $c \in T, u_i' \in T^*$. Po vykonaní operácie dostaneme dvojicu $(\underline{x_1 c}, x_2)$. Keďže predtým slovo x_1 neobsahovalo celú ľavú handle, tak v tomto kroku sme nemohli prejsť za jej koniec. Počet reverzov, ktoré možno vygenerovať z týchto slov, sa nezmenil. Takto dostávame nový životaschopný prefix $(\underline{x_1 c}, x_2)$ pre pôvodnú vetnú formu u_i a toto odvodenie.

Na konci algoritmu, teda po spracovaní prvého kroku normálneho odvodenia, bude na zásobníku iba neterminál σ, čo je životaschopný prefix pre vetnú formu u_1 v tomto odvodení. Keďže samotný neterminál σ na zásobníku je akceptačná podmienka pre algoritmus, tak sme zostrojili akceptačný výpočet k tomuto normálnemu odvodeniu. □

[19] Za využitia predpokladu, že gramatika generuje aspoň jeden reverz.

2.3.3 Veta o správnosti algoritmu

Teraz máme všetky potrebné tvrdenia na to, aby sme mohli dokázať vetu 2.3.1 z úvodu tohto oddielu:

Veta 2.3.1 (Správnosť algoritmu „posuň, redukuj, otoč“)**.** *Nech gramatika* $G = (N, T, P, \sigma, ®)$ *je v normálnom tvare so znalosťou generovaných reverzov. Ak algoritmus „posuň, redukuj, otoč“ z oddielu 2.1 dostane na vstupe gramatiku* G *a slovo* $w \in T^*$*, tak akceptuje svoj vstup práve vtedy, keď* $w \in L(G)$*.*

Dôkaz. Platnosť tejto vety vyplýva ako dôsledok z tvrdenia 2.3.4, ktoré hovorí, že ku každému akceptačnému výpočtu podľa algoritmu existuje odvodenie slova w v gramatike G a tvrdenia 2.3.8, ktoré hovorí, že ku každému normálnemu odvodeniu slova w v gramatike G existuje akceptačný výpočet podľa nedeterministického algoritmu „posuň, redukuj, otoč“. □

Ľahko možno vidieť, že pre všetky slová w z jazyka $L(G)$ pre ľubovoľnú gramatiku $G = (N, T, P, \sigma, ®)$ existuje akceptačný výpočet podľa nedeterministického algoritmu „posuň, redukuj, otoč“ taký, v ktorom sa vykoná lineárny počet otočení zásobníka od dĺžky odvodenia slova pre nejaké slovo $w' \in L_{CF}(G)$, pre ktoré platí $\varrho(w') = w$.

Záver

V práci sme sa zaoberali nedeterministickým variantom syntaktickej analýzy zdola nahor pre triedu jazykov generovaných bezkontextovými gramatikami s generovaním reverzov. Opísali sme rozšírený nedeterministický algoritmus „posuň, redukuj, otoč“, kde nová operácia „otoč“ predstavuje otočenie zásobníka. Vstupom pre tento algoritmus môže byť ľubovoľná bezkontextová gramatika s generovaním konštantného počtu reverzov a v normálnom tvare so znalosťou generovaných reverzov. Dokázali sme správnosť tohto nového algoritmu a popísali sme *normálne odvodenia*, ktoré tento algoritmus spätne konštruuje. Pre potreby dôkazu sme pre gramatiky s generovaním reverzov zaviedli obdoby pojmov *handle* a *životaschopný prefix* známych z klasickej syntaktickej analýzy zdola nahor pre štandardné bezkontextové gramatiky.

V oblasti syntaktickej analýzy pre gramatiky s generovaním reverzov zostáva stále veľa otvorených problémov. Hlavným otvoreným problémom je preskúmanie možností determinizácie algoritmu „posuň, redukuj, otoč“ pre niektoré špeciálne triedy vstupných gramatík. Takouto triedou gramatík by mohla byť napríklad vhodne definovaná obdoba LR(0) gramatík alebo LR(k) gramatík pre bezkontextové gramatiky s generovaním reverzov. V práci sme sa venovali iba syntaktickej analýze zdola nahor. Ďalším pokračovaním v tejto oblasti by mohla byť deterministická syntaktická analýza zhora nadol.

Literatúra

[1] Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. Compilers: Principles, Techniques, and Tools (2nd Edition), pages 236–238. Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

[2] Markus Holzer and Martin Kutrib. Flip-pushdown automata: k + 1 pushdown reversals are better than k. In Lecture Notes in Computer Science, 03 2003.

[3] Markus Holzer and Martin Kutrib. Flip-pushdown automata: Nondeterminism is better than determinism. In Zoltán Ésik and Zoltán Fülöp, editors, Developments in Language Theory, pages 361–372, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[4] John E. Hopcroft and Jeff D. Ullman. Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Company, 1979.

[5] Laura Kallmeyer. Parsing beyond context-free grammars. Springer Science & Business Media, 2010.

[6] Peter Kostolányi. Two grammatical equivalents of flip-pushdown automata. Italiano G.F., Margaria-Steffen T., Pokorný J., Quisquater JJ., Wattenhofer R. (eds) SOFSEM 2015: Theory and Practice of Computer Science, Lecture Notes in Computer Science, vol 8939., 2015.

[7] Peter Kostolányi. A pumping lemma for flip-pushdown languages. RAIRO - Theoretical Informatics and Applications, 50, 10 2016.

[8] Palash Sarkar. Pushdown automaton with the ability to flip its stack. report no. 81. In Electronic Colloquium on Computational Complexity (ECCC), 2001.