3.4.15 Greibachové normální forma (tvar). Řekneme, že CF gramatika $\mathcal{G}$ je v *Greibachové normální formě* (v *Greibachové normálním tvaru*), jestliže všechna pravidla mají tvar:

$$A \to a\alpha, \quad \text{kde } a \in \Sigma \text{ a } \alpha \in N^\star.$$

To znamená, že pravá strana každého pravidla začíná jedním terminálem a po něm následuje několik (třeba i žádný) neterminálů.

Uvědomte si, že pro gramatiku v Greibachové normální formě derivace slova délky n má n kroků. Také je zřejmé, že gramatika v Greibachové normálním tvaru vygeneruje pouze slova délky aspoň 1.

3.4.16 Věta. Ke každému bezkontextovému jazyku L existuje CF gramatika v Greibachové normálním tvaru $\mathcal{G}$ taková, že

$$L(\mathcal{G}) = L - \{\varepsilon\}.$$

3.4.17 Postup nalezení gramatiky v Greibachové normálním tvaru. Máme nevypouštěcí CF gramatiku $\mathcal{G}$.

- Očíslujeme její neterminály, tj. $N = \{A_1, A_2, \ldots, A_n\}$.
- Postupným spojováním pravidel a odstraňováním levé rekurze získáme pravidla, kde pravá strana začíná terminálem nebo jsou tvaru $A_i \to A_j\alpha$ pro $i < j$.
- Postupným spojováním pravidel (od n do 1) dosáhneme toho, aby pravidla s levou stranou A_i měla tvar $A_i \to a\alpha$, kde $a \in \Sigma$.
- Zajistíme, aby pravidla obsahující nové neterminály (vzniklé při odstraňování levé rekurz

3.5 Analýza shora

V jedné z minulých přednášek jsme si ukázali algoritmus CYK, který pro danou gramatiku v Chomského normálním tvaru a pro dané slovo rozhodne, zda je slovo gramatikou generováno. Velmi často je potřeba vědět nejen, zda je slovo vygenerováno, ale znát i posloupnost pravidel gramatiky, která byla použita v levé derivaci daného terminálního slova. A právě k tomuto úkolu se často používá tak zvaná analýza shora. Pro některé CF gramatiky je tato analýza jednoduchá, pro jiné je složitější – používá „backtracking“.

3.5.1 Analýza shora. Již víme, viz. věta 3.2.23, že ke každé CF gramatice $\mathcal{G}$ můžeme sestrojit zásobníkový automat A takový, že jazyk přijímaný A prázdným zásobníkem je jazyk $L(\mathcal{G})$. Přitom zásobníkový automat

- má-li na vrcholu zásobníku neterminál A nahradí ho na zásobníku pravou stranou α pravidla $A \to \alpha$;
- má-li na vrcholu zásobníku terminál a a tento terminál se shoduje s čteným vstupem slova, odstraníme a z vrcholu zásobníku a přejdeme na vstupu na další písmeno;
- má-li na vrcholu zásobníku terminál a a tento terminál se neshoduje s čteným symbolem slova, automat se neúspěšně zastaví.

Ukážeme si postup na příkladě.

3.5.2 Příklad. Je dána CF gramatika $\mathcal{G}$ pravidly:

$$S \to aSa, S \to bSb, S \to c.$$

Tato gramatika generuje jazyk $L = \{wcw^R \mid w \in \{a,b\}^\star\}$. Očíslujme jednotlivá pravidla

1. $S \to aSa$
2. $S \to bSb$
3. $S \to c$

Derivace slova $bacab \in L(\mathcal{G})$ je

$$S \Rightarrow bSb \Rightarrow baSab \Rightarrow bacab,$$

kde jsme nejprve použili pravidlo 2, pak pravidlo 1 a na konec pravidlo 3.

V odpovídajícím zásobníkovém automatu postupujeme takto:

$$(q, bacab, S) \vdash (q, bacab, bSb) \vdash (q, acab, Sb) \vdash (q, acab, aSab) \vdash (q, cab, Sab) \vdash$$

$$\vdash (q, cab, cab) \vdash (q, ab, ab) \vdash (q, b, b) \vdash (q, \varepsilon, \varepsilon).$$

Jednotlivé kroky byly:

1. přepis neterminálu na vrcholu zásobníku podle některého pravidla gramatiky,
2. v případě, že na vrcholu zásobníku byl terminální symbol shodující se s právě čteným symbolem na vstupu, tzv. „krácení“, tj. přečtení symbolu na vstupu a odstranění vrcholu zásobníku.

Výsledkem analýzy je pak posloupnost čísel pravidel v pořadí v jakém byla použita. Tedy v našem případě je to posloupnost 2, 1, 3.

K tomu, abychom správně utvořili přijímající výpočet zásobníkového automatu, potřebujeme znát výsledek a to danou posloupnost čísel pravidel. Je-li jasné ze vstupního slova (písmene, které bude první čtelno), že existuje pouze jediné pravidlo, je analýza snadná.

3.5.3 Jednoduché $LL(1)$ gramatiky. Bezkontextová gramatika $\mathcal{G}$ se nazývá *jednoduchá* $LL(1)$ gramatika, jestliže pro každé její pravidlo platí:

1. Pravá strana každého pravidla začíná terminálním symbolem.
2. Jestliže dvě pravidla mají stejnou levou stranu, pak se liší terminálním symbolem, kterým začíná pravá strana pravidla.

Pro jednoduché $LL(1)$ gramatiky je lehké provádět analýzu shora. Máme-li na vstupu slovo $w = a_1a_2\ldots a_k$, pak jako první má smysl použít pouze pravidlo $S \to \alpha$, pro které α začíná terminálem a_1 a takové pravidlo je nejvýše jedno. Po zkrácení všech terminálů, které se shodovaly (alespoň a_1 bylo možné zkrátit) se může stát:

1. slovo přijmeme (vyprázdnili jsme zásobník a současně přečetli celé slovo),
2. na vrcholu zásobníku byl jiný terminální symbol než byl čten na vstupu nebo jsme přečetli celé slovo a nevyprázdnili zásobník — v tomto případě automat neúspěšně skončí,
3. na vrcholu zásobníku je neterminální symbol A a na vstupu je symbol a_i.

V případě 3. z definice jednoduché $LL(1)$ gramatiky víme, že existuje nejvýše jedno pravidlo $A \to \beta$, kde a_i je první symbol β. Tedy opět máme nejvýše jednu možnost, jak postupovat, abychom se dostali slovo w vygenerovali.

Právě popsaný postup formalizuje pojem $LL(1)$ *analyzátoru*.

3.5.4 Příklad. Je dána bezkontextová gramatika $\mathcal{G}$ z příkladu 3.5.2, tj.

1. $S \to aSa$
2. $S \to bSb$
3. $S \to c$

Její $LL(1)$ analyzátor je následující tabulka:

	a	b	c	ε
S	$aSa, 1$	$bSb, 2$	$c, 3$	chyba
a	krátit	chyba	chyba	chyba
b	chyba	krátit	chyba	chyba
c	chyba	chyba	krátit	chyba
ε	chyba	chyba	chyba	přijmout

3.5.5 $LL(1)$ **analyzátor** je v podstatě tabulka, kde řádky odpovídají terminálům, neterminálům a prázdnému slovu, sloupce odpovídají terminálům a prázdnému slovu.

V tabulce je:

- V řádku označeném neterminálem X a sloupci označeném terminálem a pravá strana pravidla $X \to a\alpha \in P$ a číslo tohoto pravidla, v případě, že takové pravidlo existuje.
- V řádku označeném terminálem a a sloupci označeném stejným terminálem a je krátit (tj. z čteného slova odstraníme terminál a).
- V řádku označeném ε a sloupci označeném ε je přijmout (výpočet úspěšně skončil a posloupnost pravidel udává levou derivaci vstupního slova).
- Ve všech ostatních případech je v tabulce chyba (výpočet skončil neúspěšně, nezvolili jsme správné pořadí pravidel nebo slovo není gramatikou generováno).

3.5.6 $LL(1)$ **gramatiky.** V některých případech můžeme $LL(1)$ analyzátor vytvořit i pro gramatiky, které nejsou jednoduché $LL(1)$ gramatiky. Stačí, aby platilo, že v případě, kdy čteme na vstupu terminál a a v levé derivaci máme přepsat neterminál X, tak existuje nejvýše jedno pravidlo, které může vést k vygenerování daného slova.

Označme $\mathbf{Fi}(\alpha)$ množinu všech terminálních symbolů a na něž začíná některé terminální slovo generované z α. V případě, že z α se dá vygenerovat prázdné slovo, ε také patří do $\mathbf{Fi}(\alpha)$.

$LL(1)$ gramatika je zhruba řečeno gramatika splňující:

1. Jestliže v gramatice $\mathcal{G}$ existují dvě pravidla $A \to \alpha$ a $A \to \beta$ se stejnou levou stranou, pak pro každé dvě levé derivace $S \Rightarrow^\star uA\mu$ a $S \Rightarrow^\star vA\chi$, $u, v \in \Sigma^\star$, $\mu, \chi \in (N \cup \Sigma)^\star$ platí $\mathbf{Fi}(\alpha\mu) \cap \mathbf{Fi}(\beta\chi) = \emptyset$. (To znamená, že podle vstupního symbolu jednoznačně poznáme, zda použít první nebo druhé pravidlo.)
2. Jestliže navíc $\mathbf{Fi}(A)$ obsahuje ε, pak $\mathbf{Fi}(\mu)$ je také disjunktní s $\mathbf{Fi}(\alpha\mu)$ (a tedy i $\mathbf{Fi}(\beta\chi)$).

e) také měla požadovaný tvar.

- Jestliže nyní máme terminální symbol uvnitř pravé strany některého pravidla, zavedeme za něj nový neterminál.

3.5.7 Úlohy algoritmicky řešitelné pro bezkontextové gramatiky Pro bezkontextové gramatiky jsou řešitelné následující problémy:

a) Otázka zda daná bezkontextová gramatika generuje neprázdný jazyk.

Zde stačí provést první krok algoritmu redukce gramatiky. Jazyk generovaný gramatikou je neprázdný právě tehdy, když množina neterminálů, ze kterých lze vygenerovat terminální slovo, obsahuje startovací symbol.

b Pro danou bezkontextovou gramatiku $\mathcal{G}$ a slovo w zjistit, zda $w \in L(\mathcal{G})$.

Zde stačí na příklad převést gramatiku do Chomského normálního tvaru a algoritmus CYK rozhodne, zda slovo je generováno nebo ne.

V další uvidíme, že pro řadu dalších otázek, např. zda dvě bezkontextové gramatiky generují alespoň jedno společné slovo, algoritmus, který by je vyřešil neexistuje.

3.5.8 Kontextové gramatiky, kontextové jazyky Víme, že jazyk $L = \{a^n b^n c^n \mid n \geq 1\}$ není bezkontextový. Existuje však pro něj kontextová gramatika, která jej generuje (gramatiku si ukážeme na přednášce).

I kontextové jazyky mají svůj nástroj v podobě „zobecněného“ automatu, který přijímá právě je. Jedná se o specielní případ Turingova stroje. Pojem Turingova stroje si přiblížíme v následující sekci.