VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

LINEÁRNÍ GRAMATICKÉ SYSTÉMY A JEJICH APLIKACE V SYNTAKTICKÉ ANALÝZE

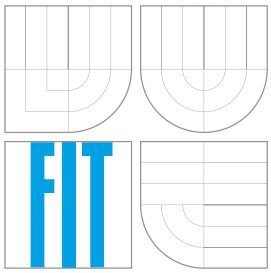
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ LICHOTA

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

LINEÁRNÍ GRAMATICKÉ SYSTÉMY A JEJICH APLIKACE V SYNTAKTICKÉ ANALÝZE

LINEAR GRAMMAR SYSTEMS AND THEIR APPLICATION IN PARSING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LUKÁŠ LICHOTA
AUTHOR

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

BRNO 2014

Abstrakt

Táto práce se zabývá modifikací klasických gramatických systémů na bázi bezkontextových gramatík a zavádí nové, modifikované lineární gramatické systémy na bázi gramatík lineárních. Nejprve budou teoreticky popsané a v další časti budou prakticky implementovány k syntaktické analýze.

Abstract

This work deals with the modification of classical grammar systems with context-free base and defines new, modified linear grammar systems with linear grammar base. At first they will be teoreticly described and in the next part they will be implemented for parsing.

Klíčová slova

gramatické systémy, lineární gramatické systémy, syntaktická analýza

Keywords

grammar systems, linear grammar systems, parsing

Citace

Lukáš Lichota: Lineární gramatické systémy a jejich aplikace v syntaktické analýze, bakalářská práce, Brno, FIT VUT v Brně, 2014

Lineární gramatické systémy a jejich aplikace v syntaktické analýze

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana prof. RNDr. Alexandra Medunu, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Lukáš Lichota
19. května 2014

Poděkování

Tímto bych rád poděkoval prof. RNDr. Alexandrovi Medunovi, CSc. za výbornou odbornou pomoc a ochotu pomáhat počas celé tvorby bakalářske práce.

© Lukáš Lichota, 2014.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **3**
- 1.1 Cieľ práce . . . 3
- 1.2 Štruktúra dokumentu . . . 3

2 Základné definície a pojmy **5**
- 2.1 Definície základných pojmov . . . 5
- 2.2 Operácie nad jazykmi . . . 6
- 2.3 Základné typy gramatík . . . 8

3 Gramatické systémy **12**
- 3.1 CD gramatické systémy . . . 12
- 3.2 PC gramatické systémy . . . 16

4 Modifikácia gramatických systémov na báze lineárnych gramatík **20**
- 4.1 Modifikácia CD gramatických systémov . . . 20
  - 4.1.1 Generatívna sila CD lineárnych gramatických systémov . . . 21
- 4.2 Modifikácia PC gramatických systémov . . . 22
  - 4.2.1 Generatívna sila PC lineárnych gramatických systémov . . . 23
  - 4.2.2 Generatívna sila PC lineárnych gramatických systémov v porovnaní s klasickými PC lineárnymi systémami . . . 24

5 Implementácia lineárnych gramatických systémov **26**
- 5.1 Štruktúra aplikácie . . . 26
- 5.2 Reprezentácia dát . . . 28
  - 5.2.1 Reprezentácia CD lineárneho gramatického systému . . . 28
- 5.3 Implementácia metód CD lineárneho gramatického systému . . . 28
  - 5.3.1 Kontrola zadania validného CD lineárneho gramatického systému . . . 28
  - 5.3.2 Kontrola aplikácie pravidiel na reťazec . . . 29
  - 5.3.3 Implementácia hlavného cyklu CD lineárneho gramatického systému 30
- 5.4 Implementácia metód PC lineárneho gramatického systému . . . 30
  - 5.4.1 Kontrola zadania validného PC lineárneho gramatického systému . . . 31
  - 5.4.2 Kontrola aplikácie pravidiel na reťazec . . . 31
  - 5.4.3 Implementácia hlavného cyklu PC lineárneho gramatického systému 31
- 5.5 Linear Grammar System Creator . . . 32

6 Záver **35**

A Obsah CD **37**

B Inštalácia **38**

C Manuál k programu **39**

Kapitola 1

Úvod

1.1 Cieľ práce

V súčasnej dobe sa v informatike stretávame čím ďalej tým viac s komplexnými problémami a úlohami, na vyriešení ktorých pracuje viacero procesorov, ktoré spolupracujú dohodnutým spôsobom. Objavujú sa tak pojmy ako distribúcia, kooperácia, paralelizmus, komunikácia, či synchronizácia. Nakoľko vo väčšine týchto problémov zohráva úlohu teoretická informatika, či už konkrétnejšie formálne jazyky, vyvstáva výzva pre vytvorenie systému gramatík, resp. automatov, ktoré spolupracujú pre generovanie, resp. rozpoznávanie jazyka [2]. Na scéne teoretickej informatiky sa preto obavujú gramatické systémy, ktoré túto výzvu premenili na realitu v podobe CD a PC gramatických systémov, ktorých základným prvkom sa stali bezkontextové gramatiky.

Táto práca mení klasické poňatie gramatických systémov a transformuje ho na gramatické systémy, ktoré nie sú nutne založené len na báze bezkontextových gramatík, ale definuje nové, lineárne gramatické systémy založené na báze lineárnych gramatík. Pri prvotnej myšlienke by sa mohlo zdať, že generatívna sila takýchto gramatických systémov bude príliš oslabená oproti klasickým gramatickým systémom a vyššia ako pri bežných lineárnych gramatikách, čo však táto práca vyvracia. Vyvodené závery a myšlienkové postupy budú stále demonštrované na príkladoch.

Okrem teoretickej časti práca obsahuje aj popis problémov a algoritmov potrebných pre transformáciu teoretického modelu do programovej podoby. Ukazuje, že použitím vhodných heuristík je možné výrazne skrátiť syntaktickú analýzu postavenú na lineárnych gramatických systémoch a demonštruje ich činnosť v implementovanom programe. Program obsahuje grafické užívateľské rozhranie, kde je možné jednoducho vytvoriť CD alebo PC lineárny gramatický systém a skúmať, ako generuje hľadané reťazce, nakoľko je program schopný ukázať celý proces generovania hľadaného reťazca.

1.2 Štruktúra dokumentu

Najprv zadefinujeme základné pojmy a definície, budeme postupovať od najzákladnejších a najjednoduchších k zložitejším a prejdeme všetok teoretický základ potrebný pre zadefinovanie gramatických systémov, ktoré budú definované v ďalšej kapitole.

Ďalej zadefinujeme modifikované – CD a PC lineárne gramatické systémy a predstavíme ich vlastnosti, hlavne generatívnu silu a porovnáme ju s generatívnou silou klasických gramatických systémov na báze bezkontextových gramatík.

Nakoniec bude popísaná implementácia zavedených CD a PC lineárnych gramatických systémov, významné algoritmy a problémy a bude popísaný program Linear Grammar System Creator, ktorý je výsledkom implementačnej časti a slúži pre demonštráciu poznatkov získaných v teoretickej časti.

Kapitola 2

Základné definície a pojmy

2.1 Definície základných pojmov

Definícia abecedy

Abeceda je konečná neprázdna množina elementov, ktoré nazývame symboly (viď [5]).

Definícia reťazca

Nech Σ je abeceda. Potom:

- ε je reťazec nad abecedou Σ.
- Pokiaľ x je reťazec nad abecedou Σ a $a \in \Sigma$, potom xa je reťazec nad abecedou Σ.

Poznámka: Symbol ε značí *prázdny reťazec*, teda taký, ktorý neobsahuje žiadny symbol. Symbolom Σ^* budeme značiť množinu všetkých reťazcov nad abecedou Σ.
(viď [5])

Definícia dĺžky reťazca

Nech x je reťazec nad abecedou Σ. Dĺžka reťaza x, $|x|$, je definovaná nasledovne:

- Pokiaľ $x = \varepsilon$, potom $|x| = 0$.
- Pokiaľ $x = a_1 \dots a_n$, potom $|x| = n$, pre $n \geq 1$ a $a_i \in \Sigma$ pre všetky $i = 1, \dots, n$.

(viď [5])

Definícia konkatenácie reťazcov

Nech x a y sú dva reťazce nad abecedou Σ. Konkatenácia reťazcov x a y je reťazec xy.
(viď [5])

Definícia mocniny reťazca

Nech x je reťazec nad abecedou Σ. Pre $i \geq 0$, $i-$tá mocnina reťaca x, x^i, je definovaná nasledovne:

- $x_0 = \varepsilon$

- pre $i \geq 1 : x^i = x^{i-1}$

(viď [5])

Definícia reverzácie reťazca

Nech x je reťazec nad abecedou Σ. Reverzácia reťaza x, $reversal(x)$, je definovaná nasledovne:

- Pokiaľ $x = \varepsilon$, potom $reversal(x) = \varepsilon$.
- Pokiaĺ $x = a_1, \ldots, a_n$ potom $reversal(a_1, \ldots, a_n) = a_n, dotsc, a_1$ pre $n \geq 1$ a $a_i \in \Sigma$ pre všetky $i = 1, \ldots, n$.

(viď [5])

Definícia prefixu reťazca

Nech x a y sú dva reťazce nad abecedou Σ. Potom x je prefixom y, pokiaľ existuje reťazec z nad abecedou Σ, pričom platí $xz = y$.
(viď [5])

Definícia sufixu reťazca

Nech x a y sú dva reťazce nad abecedou Σ. Potom x je prefixom y, pokiaľ existuje reťazec z nad abecedou Σ, pričom platí $zx = y$.
(viď [5])

Definícia podreťazca

Nech x a y sú dva reťazce nad abecedou Σ. Potom x je podreťazec y, pokiaľ existujú reťazce z, z' nad abecedou Σ, pričom platí $zxz' = y$.
(viď [5])

Definícia formálneho jazyka

Nech je daná abeceda Σ a nech Σ^* značí množinu všetkých reťazcov nad abecedou Σ. Každá podmnožina $L \subseteq \Sigma^*$ je jazyk nad abecedou Σ.
(viď [4])

Definícia konečného a nakonečného jazyka

Jazyk L je konečný, pokiaľ L obsahuje konečný počet reťazcov, inak je nekonečný.
(viď [4])

2.2 Operácie nad jazykmi

Definícia zjednotenia dvoch jazykov

Nech L_1 a L_2 sú dva jazyky nad abecedou Σ. Zjednotenie jazykov L_1 a L_2, $L_1 \cup L_2$ je definované nasledovne:

$$L_1 \cup L_2 = \{x : x \in L_1 \vee x \in L_2\}$$

(viď [5])

Definícia prieniku dvoch jazykov

Nech L_1 a L_2 sú dva jazyky nad abecedou Σ. Prienik jazykov L_1 a L_2, $L_1 \cap L_2$ je definovaný nasledovne:

$$L_1 \cap L_2 = \{x : x \in L_1 \wedge x \in L_2\}$$

(viď [5])

Definícia rozdielu dvoch jazykov

Nech L_1 a L_2 sú dva jazyky nad abecedou Σ. Rozdiel jazykov L_1 a L_2, $L_1 - L_2$ je definovaný nasledovne:

$$L_1 - L_2 = \{x : x \in L_1 \wedge x \notin L_2\}$$

(viď [5])

Definícia konkatenácie dvoch jazykov

Nech L_1 a L_2 sú dva jazyky nad abecedou Σ. Konkatenácia jazykov L_1 a L_2, $L_1 L_2$ je definovaná nasledovne:

$$L_1 L_2 = \{xy : x \in L_1 \wedge y \in L_2\}$$

(viď [5])

Definícia doplnku jazyka

Nech L je jazyk nad abecedou Σ. Doplnok jazyka L, $\overline{L}$ je definovaný nasledovne:

$$\overline{L} = \Sigma^* - L$$

(viď [5])

Definícia mocniny jazyka

Nech L je jazyk nad abecedou Σ. Pre $i \geq 0$, i-tá mocnina jazyka L, L^i je definovaná nasledovne:

- $L^0 = \{\varepsilon\}$
- Pre $i \geq 1 : L^i = LL^{i-1}$

(viď [5])

Definícia iterácie jazyka

Nech L je jazyk nad abecedou Σ. Iterácia jazyka L, L^* je definovaná nasledovne:

$$L^* = \bigcup_{n=0}^{\infty} L^n$$

(viď [5])

Definícia pozitívnej iterácie jazyka

Nech L je jazyk nad abecedou Σ. Pozitívna iterácia jazyka L, L^+ je definovaná nasledovne:

$$L^+ = \bigcup_{n=1}^{\infty} L^n$$

(viď [5])

2.3 Základné typy gramatík

Definícia neobmedzenej gramatiky

Neobmedzená gramatika G je štvorica $G = (N, T, P, S)$, kde:

- N je konečná množina neterminálnych symbolov.
- T je konečná množina terminálnych symbolov, pričom $N \cap T = \varnothing$.
- P je konečná množina pravidiel tvaru $x \to y$, kde $x \in (N \cap T)^* N (N \cap T)^*$ a $y \in (N \cap T)^*$.
- S je počiatočný neterminálny symbol.

Poznámka: Množinu všetkých jazykov, ktoré sú generované nejakou neobmedzenou gramatikou, nazveme *triedou rekurzívne vyčísliteľných jazykov*, alebo taktiež *triedou jazykov typu 0.*
(viď [4])

Definícia priamej derivácie pri neobmedzenej gramatike

Nech $G = (N, T, P, S)$ je neobmedzená gramatika, nech $u, v \in (N \cap T)^*$ a $p = x \to y \in P$ je pravidlo. Potom hovoríme, že uxv priamo derivuje uyv podľa pravidla p a zapisujeme $uxv \Rightarrow uyv$ $[p]$, alebo skrátene $uxv \Rightarrow uyv$.
(viď [4])

Definícia sekvencie derivácií pri neobmedzenej gramatike

Nech $G = (N, T, P, S)$ je neobmedzená gramatika.

- Nech $u \in (N \cup T)^*$. Potom hovoríme, že u derivuje v 0-krokoch u a zapisujeme $u \Rightarrow^0 u$ $[\varepsilon]$, alebo skrátene $u \Rightarrow^0 u$.
- Nech $u_0, u_1, \ldots, u_n \in (N \cup T)^*$, nech pre všetky $i = 1, \ldots, n$ platí $u_{i-1} \Rightarrow u_i$ $[p_i]$. Potom hovoríme, že u_0 derivuje v n-krokoch u_n a zapisujeme $u_0 \Rightarrow^n u_n$ $[p_1 p_2 \ldots p_n]$, alebo skrátene $u_0 \Rightarrow^n u_n$.
- Nech $u \Rightarrow^n v$ $[\pi]$ pre nejaké $n \geq 1$; $u, v \in (N \cup T)^*$. Potom hovoríme, že u netriviálne derivuje v a zapisujeme $u \Rightarrow^+ v$ $[\pi]$, alebo skrátene $u \Rightarrow^+ v$.
- Nech $u \Rightarrow^n v$ $[\pi]$ pre nejaké $n \geq 0$; $u, v \in (N \cup T)^*$. Potom hovoríme, že u derivuje v a zapisujeme $u \Rightarrow^* v$ $[\pi]$, alebo skrátene $u \Rightarrow^* v$.

(viď [4])

Definícia vetnej formy pri neobmedzenej gramatike

Nech $G = (N, T, P, S)$ je neobmedzená gramatika. Hovoríme, že $u \in (N \cap T)^*$ je vetná forma v neobmedzenej gramatike G práve vtedy, keď $S \Rightarrow^* u$.
(viď [4])

Definícia jazyka generovaného neobmedzenou gramatikou

Nech $G = (N, T, P, S)$ je neobmedzená gramatika. Jazyk generovaný neobmedzenou gramatikou G, $L(G)$, je definovaný nasledovne:

$$L(G) = \{w : w \in T^* \wedge S \Rightarrow^* w\}$$

(viď [4])

Definícia kontextovej gramatiky

Kontextová gramatika G je štvorica $G = (N, T, P, S)$, kde:

- N je konečná množina neterminálnych symbolov.
- T je konečná množina terminálnych symbolov, pričom $N \cap T = \varnothing$.
- P je konečná množina pravidiel tvaru $x \to y$, kde $x \in (N \cap T)^* N (N \cap T)^*$ a $y \in (N \cap T)^*$, pričom $|x| \leq |y|$.
- S je počiatočný neterminálny symbol.

Poznámka: Množinu všetkých jazykov, ktoré sú generované nejakou kontextovou gramatikou, nazveme *triedou kontextových jazykov*, alebo taktiež *triedou jazykov typu 1.*
(viď [4])

Definícia priamej derivácie, sekvencie derivácií a jazyka generovaného kontextovou gramatikou

Definície priamej derivácie, sekvencie derivácií a jazyka generovaného kontextovou gramatikou sú totožné s definíciami uvedenými pri neobmedzenej gramatike.
(viď [4])

Definícia bezkontextovej gramatiky

Bezkontextová gramatika G je štvorica $G = (N, T, P, S)$, kde:

- N je konečná množina neterminálnych symbolov.
- T je konečná množina terminálnych symbolov, pričom $N \cap T = \varnothing$.
- P je konečná množina pravidiel tvaru $A \to x$, kde $A \in N$ a $x \in (N \cup T)^*$
- S je počiatočný neterminálny symbol.

Poznámka: Množinu všetkých jazykov, ktoré sú generované nejakou bezkontextovou gramatikou, nazveme *triedou bezkontextových jazykov*, alebo taktiež *triedou jazykov typu 2.*
(viď [5])

Definícia priamej derivácie pri bezkontextovej gramatike

Nech $G = (N, T, P, S)$ je bezkontextová gramatika, nech $u, v \in (N \cup T)^*$ a $p = A \to x \in P$ je pravidlo. Potom hovoríme, že uAv priamo derivuje uxv podľa pravidla p a zapisujeme $uAv \Rightarrow uxv$ $[p]$, alebo skrátene $uAv \Rightarrow uxv$.
(viď [5])

Definícia najľavejšej derivácie pri bezkontextovej gramatike

Nech $G = (N, T, P, S)$ je bezkontextová gramatika, $u \in T^*$ a $v \in (N \cup T)^*$ a $p = A \to x \in P$ je pravido. Potom hovoríme, že uAv priamo derivuje v najľavejšej derivácii uxv podľa pravidla p, a zapisujeme $uAv \Rightarrow_{lm} uxv$ $[p]$, alebo skrátene $uAv \Rightarrow_{lm} uxv$.
(viď [5])

Definícia najpravejšej derivácie pri bezkontextovej gramatike

Nech $G = (N, T, P, S)$ je bezkontextová gramatika, $u \in (N \cup T)^*$ a $v \in T^*$ a $p = A \to x \in P$ je pravido. Potom hovoríme, že uAv priamo derivuje v najpravejšej derivácii uxv podľa pravidla p, a zapisujeme $uAv \Rightarrow_{rm} uxv$ $[p]$, alebo skrátene $uAv \Rightarrow_{rm} uxv$.
(viď [5])

Definícia sekvencie derivácií pri bezkontextovej gramatike

Nech $G = (N, T, P, S)$ je bezkontextová gramatika.

- Nech $u \in (N \cup T)^*$. Potom hovoríme, že u derivuje v 0-krokoch u a zapisujeme $u \Rightarrow^0 u$ $[\varepsilon]$, alebo skrátene $u \Rightarrow^0 u$.
- Nech $u_0, u_1, \ldots, u_n \in (N \cup T)^*$, nech pre všetky $i = 1, \ldots, n$ platí $u_{i-1} \Rightarrow u_i$ $[p_i]$. Potom hovoríme, že u_0 derivuje v n-krokoch u_n a zapisujeme $u_0 \Rightarrow^n u_n$ $[p_1 p_2 \ldots p_n]$, alebo skrátene $u_0 \Rightarrow^n u_n$.
- Nech $u \Rightarrow^n v$ $[\pi]$ pre nejaké $n \geq 1$; $u, v \in (N \cup T)^*$. Potom hovoríme, že u netriviálne derivuje v a zapisujeme $u \Rightarrow^+ v$ $[\pi]$, alebo skrátene $u \Rightarrow^+ v$.
- Nech $u \Rightarrow^n v$ $[\pi]$ pre nejaké $n \geq 0$; $u, v \in (N \cup T)^*$. Potom hovoríme, že u derivuje v a zapisujeme $u \Rightarrow^* v$ $[\pi]$, alebo skrátene $u \Rightarrow^* v$.

(viď [5])

Definícia jazyka generovaného bezkontextovou gramatikou

Nech $G = (N, T, P, S)$ je bezkontextová gramatika. Jazyk generovaný bezkontextovou gramatikou G, $L(G)$, je definovaný nasledovne:

$$L(G) = \{w : w \in T^* \wedge S \Rightarrow^* w\}$$

(viď [5])

Ľavá lineárna gramatika

Ľavá lineárna gramatika G je štvorica $G = (N, T, P, S)$, kde:

- N je konečná množina neterminálnych symbolov.
- T je konečná množina terminálnych symbolov, pričom $N \cap T = \varnothing$.
- P je konečná množina pravidiel tvaru $A \to Bx$ alebo $A \to y$, kde $A, B \in N$ a $x, y \in T^*$
- S je počiatočný neterminálny symbol.

(viď [5])

Pravá lineárna gramatika

Pravá lineárna gramatika G je štvorica $G = (N, T, P, S)$, kde:

- N je konečná množina neterminálnych symbolov.
- T je konečná množina terminálnych symbolov, pričom $N \cap T = \varnothing$.
- P je konečná množina pravidiel tvaru $A \to xB$ alebo $A \to y$, kde $A, B \in N$ a $x, y \in T^*$
- S je počiatočný neterminálny symbol.

Poznámka: Množinu všetkých jazykov, ktoré sú generované nejakou pravou lineárnou gramatikou, nazveme *triedou regulárnych jazykov*, alebo taktiež *triedou jazykov typu 3*.
(viď [5])

Lineárna gramatika

Lineárna gramatika G je štvorica $G = (N, T, P, S)$, kde:

- N je konečná množina neterminálnych symbolov.
- T je konečná množina terminálnych symbolov, pričom $N \cap T = \varnothing$.
- P je konečná množina pravidiel tvaru $A \to x$, kde $A \in N$ a $x \in T^*(N \cup \{\varepsilon\})T^*$
- S je počiatočný neterminálny symbol.

(viď [5])

Definícia pramej derivácie, sekvencie derivácií a jazyka generovaného pravou, ľavou alebo všeobecnou lineárnou gramatikou

Definície pramej derivácie, sekvencie derivácií a jazyka generovaného pravou, ľavou alebo všeobecnou lineárnou gramatikou sú totožné s definíciami uvedenými pri bezkontextovej gramatike.
(viď [5])

Kapitola 3

Gramatické systémy

Gramatické systémy sú systémy založené na komunikácií všeobecne n-tice gramatík, ktoré spolu komunikujú. V klasickom poňatí gramatických systémov ide o gramatiky bezkontextové. Podľa spôsobu komunikácie medzi gramatikami potom rozdeľujeme gramatické systémy na dva základné typy gramatických systémov: sekvenčne pracujúce CD gramatické systémy a paralelne pracujúce PC gramatické systémy. V tejto kapitole budú uvedené formálne definície týchto gramatických systémov, čo je predpokladom pre ich modifikáciu v kapitole 4.

3.1 CD gramatické systémy

CD (cooperating distributed) gramatické systémy pracujú sekvenčne. Skladajú sa z *komponentov* – gramatík. V jednom momente v takomto systéme pracuje len jedna gramatika. V závislosti od *módu* CD gramatického systému sa môže po vykonaní všeobecne k-krokov odovzdať riadenie ďalšej gramatike, ktorá pokračuje v derivovaní reťazca. Riadenie systému spočíva vo výbere komponentu v závislosti na móde CD gramatického systému. CD gramatický systém buď vygeneruje reťazec zložený z terminálnych symbolov, alebo nastane špeciálna ukončovacia podmienka, t.j. stav, kedy nemôže gramatika, resp. všetky gramatiky (záleží od použitého módu) vykonať ďalší krok a ukončí sa činnosť celého gramatického systému (viac viď [2]).

Definícia CD gramatického systému

CD gramatický systém stupňa n, $n \geq 1$, je definovaný nasledovne:

$$\Gamma = (N, T, S, P_1, \ldots, P_n, \text{kde})$$

- N je konečná množina neterminálnych symbolov.
- T je konečná množina terminálnych symbolov, pričom $N \cap T = \varnothing$.
- S je počiatočný neterminálny symbol.
- P_i je konečná množina pravidiel tvaru $A \to x$, kde $A \in N$ a $x \in (N \cup T)^*$ pre všetky $i = 1, \ldots, n$.

(viď [6])

Ukončovací derivační mód

Pokiaľ i-tá gramatika obsahuje pravidlo potrebné pre ďalší derivačný krok, prevedie tento krok, inak odovzdá riadenie nasledujúcej gramatike. Ak žiadna gramatika nemá potrebné pravidlo, nastáva špeciálny stav, kedy sa ukončí činnosť celého CD gramatického systému. Definícia:

$$x_i \Rightarrow^t y, \text{ ak}$$

- $x \Rightarrow^* y$ v $G_i = (N, T, P, S)$ a zároveň
- $y \not\Rightarrow z$ pre všetky $z \in (N \cup T)^*$

(viď [6])

k-kroký derivačný mód

Každá gramatika prevedie presne daný počet derivačných krokov k. Následne odovzdá riadenie nasledujúcej gramatike. Ak gramatika nemôže previesť až k krokov, nastáva špeciálny stav, kedy sa ukončí činnosť celého CD gramatického systému. Definícia:

$$x_i \Rightarrow^{=k} y, \text{ ak}$$

- $x \Rightarrow^k y$ v G_i

(viď [6])

Nanajvýš k-kroký derivačný mód

Každá gramatika prevedie n krokov, pričom $n \leq k$ a následne odovzdá riadenie nasledujúcej gramatike. Ak žiadna gramatika nemá potrebné pravidlo, nastáva špeciálny stav, kedy sa ukončí činnosť celého CD gramatického systému. Definícia:

$$x_i \Rightarrow^{\leq k} y, \text{ ak}$$

- $x \Rightarrow^n y$ v G_i pre nejaké $n \leq k$

(viď [6])

Prinajmenšom k-kroký derivačný mód

Každá gramatika prevedie n krokov, pričom $n \geq k$ a následne odovzdá riadenie nasledujúcej gramatike. Ak gramatika nemôže previesť až k krokov, nastáva špeciálny stav, kedy sa ukončí činnosť celého CD gramatického systému. Definícia:

$$x_i \Rightarrow^{\geq k} y, \text{ ak}$$

- $x \Rightarrow^n y$ v G_i pre nejaké $n \geq k$

(viď [6])

Množina derivačných módov

Pre potreby definície o jazyku generovanom CD gramatickým systémom označíme D množinu všetkých derivačných módov nasledovne:

$$D = \{*, t\} \cup \{\leq k, = k, \geq k : k = 1, 2, \dots\}$$

(viď [6])

Jazyk generovaný CD gramatickým systémom

Definícia jazyka generovaného CD gramatickým systémom nie je taká jednoduchá ako pri gramatikách. K definícii potrebujeme okrem množiny derivačných krokov zaviesť aj množinu možných derivácií j-tého komponentu, ktorá nám výslednú definíciu uľahčí. Definícia:

$$F(G_j, u, f) = \{v : u_j \Rightarrow^f v\}, \text{ kde } j \in \{1, \ldots, n\}, f \in D, u \in (N \cup T)^*$$

Potom jazyk generovaný CD gramatickým systémom (s ohľadom na určitý mód) má nasledujúcu definíciu:

$$L_f(\Gamma) = \{w \in T^* : \text{ kde sú } v_0, v_1, \ldots, v_m \text{ také, že}$$

$$v_i \in F(G_j, u_{i-1}, f),\ i = 1, \ldots, m,\ j_i \in \{1, \ldots, n\},$$

$$v_o = S,\ v_m = w,\ \text{pre nejaké } m \geq 1\}$$

Príklad: Majme nasledujúci CD gramatický systém:

$$\Gamma = (\{S, A, A', B, B'\}, \{a, b, c\}, S, P_1, P_2), \text{ kde:}$$

$$P_1 = \{S \to S, S \to AB, A' \to A, B' \to B\}$$

$$P_2 = \{A \to aA'b, B \to cB', A \to ab, B \to c\}$$

Dostávame tri rôzne jazyky v závislosti na použitom móde.

$$L_f(\Gamma)_{f \in \{=1, \geq 1, *, t\} \cup \{\leq k : k \geq 1\}} = \{a^n b^n c^m : m, n \geq 1\}$$

$$L_{=2}(\Gamma) = L_{\geq 2}(\Gamma) = \{a^n b^n c^n : n \geq 1\}$$

$$L_{=k}(\Gamma)_{k \geq 3} = L_{\geq 3}(\Gamma) = \varnothing$$

Vidíme, že sme dostali aj jazyk $\{a^n b^n c^n : n \geq 1\}$, ktorý nie je bezkontextový. CD gramatické systémy majú teda vyššiu silu ako bezkontextové gramatiky. (viď [6])

Označenie tried jazykov generovaných CD gramatickými systémami

Definujme nasledovné označenie:

$$CD_x^y(f), \text{ kde:}$$

- f je derivačný mód.
- y môže nadobúdať hodnotu ε a potom sú povolené ε-pravidlá alebo ak y zadaný nie je, ε-pravidlá povolené nie sú.
- x môže nadobúdať hodnotu $n,\ n \geq 1$, a teda určitý konečný počet komponentov, alebo hodnotu ∞, kde počet komponentov nie je obmedzený.

(viď [6])

Hybridné CD gramatické systémy

Pri bližšom skúmaní CD gramatických systémov nám napadne, že zadaný mód je rovnaký pre všetky komponenty, čo CD gramatický systém obmedzuje. Hybridný CD gramatický systém je teda CD gramatický systém, pri ktorom má každý komponent vlastný derivačný mód. Definícia:

$$\Gamma = (N, T, S, (P_1, f_1), \ldots, (P_n, f_n)), \text{ kde:}$$

- N, T, S, P_i sú definované rovnako ako v prípade klasického CD gramatického systému.
- f_i je mód i-tého komponentu, $f_i \in D$ pre všetky $i \in \{1, \ldots, n\}$

(viď [6])

Jazyk generovaný hybridným CD gramatickým systémom

Jazyk generovaný CD gramatickým systémom je definovaný nasledovne:

$$L_f(\Gamma) = \{w \in T^* : \text{ kde sú } v_0, v_1, \ldots, v_m \text{ také, že}$$

$$v_i \in F(G_{j_i}, u_{i-1}, f_{j_i}),\ i = 1, \ldots, m,\ j_i \in \{1, \ldots, n\},$$

$$v_o = S,\ v_m = w,\ \text{pre nejaké } m \geq 1\}$$

(viď [6])

Označenie tried jazykov generovaných hybridnými CD gramatickými systémami

Definujme nasledujúce označenie:

$$XCD^y_{x,v}(f) \text{ kde:}$$

- x, y, f sú definované rovnako ako v prípade CD gramatických systémov
- v môže nadobúdať hodnotu $m,\ m \geq 1$, potom každé pravidlo P_i obsahuje nanajvýš m pravidiel, alebo v nepíšeme, resp. dosadíme zaňho hodnotu ∞, čím značíme povolený ľubovoľný počet pravidiel.
- X nadobúda buď hodnotu H, čím značíme, že ide o hybridný CD gramatický systém (potom nepíšeme (f), nakoľko hybridný CD gramatický systém nemá globálny derivačný mód pre všetky komponenty), alebo X nadobúda hodnotu D, čím značíme že ide o deterministický CD gramatický systém (pre všetky $P_i, A \to u, A \to w \in P_i$ platí $u = w$), alebo za X nedosadíme nič, čím dávame najavo, že ide o obyčajný CD gramatický systém.

(viď [6])

Generatívna sila CD gramatických systémov

Ako sme ukázali na príklade v podkapitole 3.1, CD gramatické systémy majú vyššiu generatívnu silu ako bezkontextové gramatiky. V problematike generatívnej sily CD gramatických systémov je ešte veľa otvorených otázok. Zaujímavé teorémy:

- $\mathcal{L}(CF) = CD_1^y \subset CD_2^y \subseteq CD_r^y \subseteq CD_\infty^y \subseteq \mathcal{L}M$, pre všetky $f \in \{= k, \geq k : k \geq 2\}, r \geq 3$
- $CD_r^y(\geq k) \subseteq CD_r^y(\geq k+1)$
- $\mathcal{L}(CF) = CD_1^y(t) = CD_2^y(t) \subset CD_3^y(t) = CD_\infty^y(t) = \mathcal{L}(ET0L)$
- $\mathcal{L}(CF) = HCD_1 \subset HCD_2 \subseteq HCD_3 \subseteq HCD_4 = HCD_\infty = \mathcal{L}(M, ac)$
- $\mathcal{L}(ET0L) \subset HCD_4$
- $CD_\infty(=) \subset HCD_3$

(viac viď [2][6])

3.2 PC gramatické systémy

PC (paralell communicating) gramatické systémy pracujú paralelne. Skladajú sa z *komponentov* – gramatík. V jednom momente v takomto systéme pracujú všetky gramatiky zároveň. Komponent sa teda skladá z pravidiel a štartovacieho symbolu. Okrem klasických derivačných krokov sú v PC gramatických systémoch aj prioritné, *komunikačné kroky*. Komunikačné kroky, ako z názvu vyplýva, slúžia pre komunikáciu medzi komponentami. Ide o poskytnutie vygenerovaného reťazca komponentu, ktorý vygeneruje komunikačný symbol. Pokiaľ sa takýto symbol objaví, vykoná sa komunikačný krok a pokračuje sa v derivovaní. Ak sa niektorý komponent dostane do stavu, kedy nemá potrebné pravidlo pre ďalší derivačný krok, nastane špeciálna podmienka a ukončí sa činnosť celého PC gramatického systému. Ďalšou špeciálnou situáciou je *deadlock*, keď je v komunikačných symboloch vytvorená cyklická závislosť, čo taktiež vedie k ukončeniu činnosti gramatického systému. V praxi sa PC gramatické systémy uplatňujú v niektorých simuláciách z oblasti operačných systémov (viac viď [2]).

Definícia PC gramatického systému

PC gramatický systém stupňa n, $n \geq 1$, je definovaný nasledovne:

$$\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n)), \text{ kde}$$

- N je konečná množina neterminálnych symbolov.
- K je konečná množina komunikačných symbolov, $K = \{Q_1, \ldots, Q_n\}$
- T je konečná množina terminálnych symbolov, pričom N, K, T sú párovo disjunktné.
- S_i je počiatočný neterminálny symbol i-tého komponentu pre všetky $i = 1 \ldots, n$.
- P_i je konečná množina pravidiel tvaru $A \to x$, kde $A \in N$ a $x \in (N \cup T \cup K)^*$ pre všetky $i = 1, \ldots, n$.

(viď [7])

Derivačný krok PC gramatického systému

Derivačný krok u PC gramatického systému je prevádzaný nasledovne:
Ak platí pre všetky $1 \leq i \leq n$

- buď $x_i \Rightarrow y_i$ in $G_i = (N \cup K, T, P_i, S_i)$
- alebo $x_i = y_i \in T^*$

potom:

$$(x_i, \ldots, x_n) \Rightarrow_g (y_1, \ldots, y_n)$$

(viď [7])

Komunikačný krok PC gramatického systému

Komunikačný krok u PC gramatického systému je zložitejší ako derivačný a má pred ním prioritu. Postup prevádzania je nasledovný:

1. nastaviť $z_i = x_i$ pre všetky $i = 1, \ldots, n$.
2. ak pre každé $i = 1, \ldots, n$ platí, že $alph(x_i) \cap K \neq \varnothing$ a zároveň pre každé Q_j v x_i platí, že $alph(x_j) \cap K = \varnothing$, potom pre všetky Q_j v x_i previesť nasledujúci krok.
3. nastaviť $z_j = S_j$, nahradiť Q_j s x_j v x_i a tento novovzniknutý reťazec nastaviť do z_i.
4. preveď komunikačný krok $(x_1, \ldots, x_n) \Rightarrow_c (y_1, \ldots, y_n)$ s $y_i = z_i$ pre všetky $i = 1, \ldots, n$.

(viď [7])

Jazyk generovaný PC gramatickým systémom

Predtým než definujeme jazyk generovaný PC gramatickým systémom, je potrebné zaviesť pojem *priamej derivácie*:
Ak buď

$$(x_i, \ldots, x_n) \Rightarrow_g (y_1, \ldots, y_n)$$

alebo

$$(x_i, \ldots, x_n) \Rightarrow_c (y_1, \ldots, y_n)$$

potom

$$(x_i, \ldots, x_n) \Rightarrow (y_1, \ldots, y_n)$$

Definícia jazyka generovaného PC gramatickým systémom:

$$L(\Gamma) = \{x \in T^* : (S_1, S_2, \ldots, S_n) \Rightarrow^* (x, \alpha_2, \ldots, \alpha_n)$$

$$\alpha_i \in (N \cup T \cup K)^*, \text{ pre všetky } i = 2, \ldots, n\}$$

Príklad: Majme nasledujúci PC gramatický systém:

$$\Gamma = (\{S_1, S_1', S_2, S_3\}, K, \{a, b, c\}, (S_1, P_1), (S_3, P_3), (S_3, P_3)), \text{ kde:}$$

$$P_1 = \{S_1 \to abc, S_1 \to a^2b^2c^2, S_1 \to aS_1', S_1' \to a^3Q_2, S_2 \to b^2Q_3, S_3 \to c\}$$

$$P_2 = \{S_2 \to bS_2\}$$
$$P_2 = \{S_3 \to cS_3\}$$

$$(S_1, S_2, S_3) \Rightarrow (aS_1', b_S 2, cS_3) \Rightarrow^* (a^n S_1', b^n S_2, c^n S_3) \Rightarrow (a^{n+3} Q_2, b^{n+1} S_2, c^{n+1} S_3)$$
$$\Rightarrow (a^{n+3} b^{n+1} S_2, S_2, c^{n+1} S_3) \Rightarrow (a^{n+3} b^{n+3} Q_3, bS_2, c^{n+2} S_3) \Rightarrow (a^{n+3} b^{n+3} c^{n+2} S_3, bS_2, S_3)$$
$$\Rightarrow (a^{n+3} b^{n+3} c^{n+3}, b^2 S_2, cS_3)$$

Dostávame jazyk $L_r(\Gamma) = L_{nr}(\Gamma) = \{a^n b^n c^n : n \geq 1\}$, ktorý nie je bezkontextový. PC gramatické systémy majú teda vyššiu generatívnu silu ako bezkontextové gramatiky. (viď [7])

Centralizovaný PC gramatický systém

Centralizovaný PC gramatický systém je taký PC gramatický systém, kde len prvý komponent, P_1, môže obsahovať pravidlá, ktoré na svojej pravej strane majú komunikačné symboly. V takomto systéme nemôže dôjsť k deadlocku (viď 3.2). Definícia:

Nech $\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n))$ je PC gramatický systém. Γ je *centralizovaný* PC gramatický systém, ak $A \to x \in P_i$, pre všetky $i = 2, \ldots, n$ platí $alph(x) \cap K = \varnothing$. (viď [7])

PC gramatické systémy s návratom a bez návratu

PC gramatické systémy s návratom fungujú tak, že ak komponent poskytne svoj reťazec inému komponentu, ktorý ho požadoval vygenerovaním zodpovedajúceho komunikačného symbolu, nepokračuje komponent v generovaní reťazca ďalej, ale vráti sa k štartovaciemu symbolu a pokračuje od začiatku, teda tak, ako je to definované v postupe komunikačného kroku. Jazyk generovaný takýmto systémom označíme $L_r(\Gamma)$ (viď [7]).

PC gramatické systémy s návratom pracujú tak, že komponent, ktorý poskytol svoj reťazec inému komponentu v rámci komunikačného kroku, neprepisuje tento reťazec na štartovací symbol, ale pokračuje ďalej v jeho derivovaní. Jazyk generovaný PC gramatickým systémom bez návratu označíme $L_{nr}(\Gamma)$ (viď [7]).

Označenie tried jazykov generovaných PC gramatickými systémami

Definujme nasledujúce označenie:

$$XPC_xY$$

- X môže nadobúdať hodnotu N a potom ide o gramatický systém bez návratu, hodnotu C, vtedy ide o centralizovaný PC gramatický systém. Ak sa hodnota X neuvedie, ide o klasický PC gramatický systém.
- x označuje počet komponentov, nadobúda rovnaké hodnoty ako pri označení CD gramatických systémov.
- Y špecifikuje typ pravidiel, nadobúda hodnoty REG, ak ide o pravidlá regulárnej gramatiky, LIN, ak ide o pravidlá lineárnej gramatiky alebo CF, ak ide o pravidlá bezkontextovej gramatiky.

(viď [7])

Generatívna sila PC gramatických systémov

Ako sme ukázali na príklade, PC gramatické systémy majú vyššiu generatívnu silu ako bezkontextové gramatiky. Podobne ako u CD gramatických systémov je aj u PC gramatických systémov otázka generatívnej sily široko otvorená, napriek tomu je známych niekoľko zaujímavých teorémov:

- $PC_nREG - \mathcal{L}(LIN) \neq \varnothing$ pre $n \geq 2$
- $PC_nREG - \mathcal{L}(CF) \neq \varnothing$ pre $n \geq 3$
- $PC_nLIN - \mathcal{L}(CF) \neq \varnothing$ pre $n \geq 2$
- $PC_nREG \subset PC_{n+1}REG$ pre $n \geq 1$
- $CPC_nREG \subset CPC_nLIN \subset CPC_nCF$ pre $n \geq 1$
- $NPC_\infty CF \subseteq PC_\infty CF$
- $\mathcal{L}(M) \subset PC_\infty CF$
- $\mathcal{L}(LIN) \subset PC_\infty REG$

(viac viď [2][7])

Kapitola 4

Modifikácia gramatických systémov na báze lineárnych gramatík

Klasické gramatické systémy definované v [2] sú založené na báze bezkontextových gramatík. V tejto kapitole bude popísaná modifikácia bázy na lineárne gramatiky. Zameriame sa na definovanie takýchto systémov a na ich generatívnu silu. Budeme skúmať, či dostaneme vyššiu generatívnu silu ako majú lineárne, či bezkontextové gramatiky.

4.1 Modifikácia CD gramatických systémov

Definícia CD lineárneho gramatického systému

CD lineárny gramatický systém stupňa n, $n \geq 1$, je definovaný nasledovne:

$$\Gamma = (N, T, S, P_1, \ldots, P_n, \text{ kde})$$

- N je konečná množina neterminálnych symbolov.
- T je konečná množina terminálnych symbolov, pričom $N \cap T = \varnothing$.
- S je počiatočný neterminálny symbol.
- P_i je konečná množina pravidiel tvaru $A \to x$, kde $A \in N$ a $x \in T^*(N \cup \{\varepsilon\})T^*$ pre všetky $i = 1, \ldots, n$.

Množina derivačných módov a jazyk generovaný CD lineárnym gramatickým systémom

Množina derivačných módov CD lineárneho gramatického systému je definovaná rovnako, ako v prípade klasického CD gramatického systému, taktiež jazyk generovaný CD gramatickým systémom má rovnakú definíciu ako jazyk generovaný klasickým CD gramatickým systémom, viď podkapitola 3.1.

Označenie tried jazykov generovaných CD lineárnymi gramatickými systémami

Definujme nasledujúce označenie:

$$CD_x^y(f)LIN, \text{ kde:}$$

- f je derivačný mód.
- y môže nadobúdať hodnotu ε a potom sú povolené ε-pravidlá, alebo ak y zadaný nie je, ε-pravidlá povolené nie sú.
- x môže nadobúdať hodnotu $n,\ n \geq 1$, a teda určitý konečný počet komponentov, alebo hodnotu ∞, kde počet komponentov nie je obmedzený.

4.1.1 Generatívna sila CD lineárnych gramatických systémov

V klasických CD gramatických systémoch sme na príklade ukázali, že ich sila bola vyššia ako je sila komponentov, z ktorých sa skladajú – bezkontextových gramatík (príklad v podkapitole 3.1).

Prvá myšlienka je, že rovnakú vlastnosť budú mať aj CD lineárne gramatické systémy, a teda že ich generatívna sila bude silnejšia ako je generatívna sila lineárnych gramatík. Pokúsime sa teda definovať takýto CD lineárny gramatický systém na príklade jazyka, ktorý nie je lineárny, ale bezkontextový.

Príklad: Definujte CD lineárny gramatický systém, ktorý generuje Dyckov jazyk. Dyckov jazyk nad $2n$-prvkovou abecedou je bezkontextový jazyk, ktorý však nie je lineárny, pomenovaný podľa matematika Walthera von Dycka, ktorý je generovaný nasledujúcou gramatikou G.

$$G = (\{S\}, \{a_1, b_1, a_2, b_2, \ldots, a_n, b_n\}, P, S), \text{ kde:}$$

$$P = \{S \to \varepsilon,\ S \to a_1 S b_1 S,\ S \to a_1 S b_1 S,\ \ldots,\ S \to a_n S b_n S\}$$

V prípade, že $a_1, a_2, \ldots, a_n$ sú nejaké typy zátvoriek, zodpovedá Dyckov jazyk jazyku všetkých dobre uzátvorkovaných nad daným typom zátvoriek [3].

Zjednodušme teda abecedu na jeden typ zátvoriek a dostávame nasledujúcu bezkontextovú gramatiku.

$$G_1 = (\{S\}, \{(,)\}, P, S), \text{ kde:}$$

$$P = \{S \to \varepsilon,\ S \to (S)S\}$$

Reťazec $()(())$, ktorý gramatika G_1 generuje, by sa vygeneroval nasledovne:

$$S \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S)S \Rightarrow ()((S)S)S \Rightarrow ()(()S)S \Rightarrow ()(())S \Rightarrow ()(())$$

Pri pokuse vytvoriť CD lineárny gramatický systém však prídeme na to, že takýto CD lineárny gramatický systém neexistuje. Dôvodom je, že sme obmedzení pravidlami lineárnej gramatiky. Keďže na pravej strane pravidla môžeme mať maximálne jeden neterminál, aj keď budeme tento reťazec postupne predávať z jedného komponentu druhému sekvenčným spôsobom a pri využití derivačných krokov sa nám to nikdy nepodarí. Fakt, že v reťazci sa môže stále vyskytovať nanajvýš jeden neterminál, vedie k záveru, že generatívna sila CD lineárnych gramatických systémov je **rovnaká** ako sila lineárnych gramatík. CD lineárne gramatické systémy nemajú teda vyššiu generatívnu silu ako lineárne gramatiky, čo je

prekvapujúci záver, vzhľadom na klasické CD gramatické systémy na báze bezkontextových gramatík. Z poznatkov z tohoto príkladu môžeme odvodiť nasledujúci teorém o generatívnej sile CD gramatických systémov:

$$\mathcal{L}(LIN) = CD_1^{\varepsilon}LIN = CD_2^{\varepsilon}LIN = \ldots = CD_n^{\varepsilon}LIN$$

$$\mathcal{L}(LIN) = CD_{\infty}^{\varepsilon}LIN$$

4.2 Modifikácia PC gramatických systémov

Definícia PC lineárneho gramatického systému

PC lineárny gramatický systém stupňa $n,\ n \geq 1$, je definovaný nasledovne:

$$\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n)), \text{ kde}$$

- N je konečná množina neterminálnych symbolov.
- K je konečná množina komunikačných symbolov, $K = \{Q_1, \ldots, Q_n\}$
- T je konečná množina terminálnych symbolov, pričom N, K, T sú párovo disjunktné.
- S_i je počiatočný neterminálny symbol i-tého komponentu pre všetky $i = 1 \ldots, n$.
- P_i je konečná množina pravidiel tvaru $A \to x$, kde $A \in N$ a $x \in (T \cup K)^*(N \cup \{\varepsilon\})(T \cup K)^*$ pre všetky $i = 1, \ldots, n$.

Jazyk generovaný PC lineárnym gramatickým systémom

Jazyk generovaný PC gramatickým systémom má rovnakú definíciu ako jazyk generovaný klasickým PC gramatickým systémom.

Označenie tried jazykov generovaných PC lineárnymi gramatickými systémami

Pre definíciu použijeme označenie z podkapitoly 3.2, kde za hodnotu Y dosadíme LIN, čím sa obmedzíme len na používanie lineárnych pravidiel. Výsledné označenie je teda definované nasledovne:

$$XPC_xLIN$$

- X môže nadobúdať hodnotu N a potom ide o gramatický systém bez návratu, hodnotu C, vtedy ide o centralizovaný PC gramatický systém. Ak sa hodnota X neuvedie, ide o klasický PC gramatický systém.
- x môže nadobúdať hodnotu $n,\ n \geq 1$, a teda určitý konečný počet komponentov, alebo hodnotu ∞, kde počet komponentov nie je obmedzený.

4.2.1 Generatívna sila PC lineárnych gramatických systémov

V podkapitole 3.2 sme uviedli teorém, že klasické PC gramatické systémy na báze bezkontextových gramatík majú dokonca vyššiu generatívnu silu ako maticové gramatiky [8]. Predpoklady na generatívnu silu PC lineárnych gramatických systémov môžu byť preto vysoké, ale taktiež po závere, vyvodenom pri CD lineárnych gramatických systémoch, mierne.

Pomocou príkladu sa pokúsime ukázať silu PC lineárneho gramatického systému.

Príklad 1: Použijeme rovnaký príklad, na ktorom sa nám podarilo demonštrovať silu CD lineárneho gramatického systému, Dyckov jazyk.

Zo skúseností s CD lineárnymi gramatickými systémami vieme, že pri snahe vygenerovať v reťazci viac neterminálnych symbolov za pomoci ľubovoľného počtu komponentov, boli pravidlá lineárnej gramatiky natoľko limitujúce, že sa to nepodarilo. PC lineárne gramatické systémy však majú oproti CD lineárnym gramatickým systémom paralelný chod, kedy pracujú v jednom okamihu všetky komponenty a okrem derivačných krokov existuje aj špeciálny, komunikačný krok, pomocou ktorého sa pokúsime do reťazca dostať viacero neterminálnych symbolov. Budeme sa snažiť vytvoriť reťazec $[\,][\,[\,]\,]$ (kvôli prehľadnosti pri použití $()(())$ reťazca).

Definujme nasledujúci PC lineárny gramatický systém:

$$\Gamma = (\{S\}, K, \{[,]\}, (S, P_1), (S, P_2)), \text{ kde:}$$

$$P_1 = \{S \to \varepsilon,\ S \to [Q_2]Q_2\},\ P_2 = \{S \to S\}$$

Reťazec vygeneruje zadaný PC gramatický systém nasledovne:

$$(S,S) \Rightarrow ([Q_2]Q_2, S) \Rightarrow ([S]S, S) \Rightarrow ([\,][Q_2]Q_2, S) \Rightarrow ([\,][S]S, S) \Rightarrow ([\,][\,[Q_2]Q_2]S, S)$$

$$\Rightarrow ([\,][\,[S]S]S, S) \Rightarrow ([\,][\,[\,]S]S, S) \Rightarrow ([\,][\,[\,]\,]S, S) \Rightarrow ([\,][\,[\,]\,], S)$$

PC lineárny gramatický systém generujúci daný bezkontextový jazyk, sa nám podarilo definovať, aby sme obsiahli celý Dyckov jazyk, zmenili by sme definíciu PC lineárneho gramatického systému nasledovne:

$$\Gamma = (\{S\}, K, \{a_1, b_1, a_2, b_2, \ldots, a_n, b_n\}, (S, P_1), (S, P_2)), \text{ kde:}$$

$$P_1 = \{S \to \varepsilon,\ S \to a_1Q_2b_1Q_2,\ S \to a_2Q_2b_2Q_2,\ \ldots,\ S \Rightarrow a_nQ_2b_nQ_2\},\ P_2 = \{S \to S\}$$

Ukázali sme teda, že PC lineárny gramatický systém má **vyššiu** silu ako lineárne gramatiky, čo možno zapísať nasledovným teorémom:

$$\mathcal{L}(LIN) = CD^{\varepsilon}_{\infty}LIN \subset PC_nLIN$$

Príklad 2: V predchádzajúcom príklade sme ukázali, že generatívna sila PC lineárnych gramatických systémov je vyššia ako CD lineárnych gramatických systémov a lineárnych gramatík.

Pokúsime sa ísť ešte ďalej a definovať PC lineárny gramatický systém, ktorý bude generovať jazyk $a^nb^nc^n, n \geq 1$, ktorý nie je bezkontextový. Definujme nasledujúci PC lineárny gramatický systém:

$$\Gamma = (\{S_1, S_1', S_2, S_3\}, K, \{a, b, c\}, (S_1, P_1), (S_3, P_3), (S_3, P_3)), \text{ kde:}$$

$$P_1 = \{S_1 \to abc, S_1 \to a^2b^2c^2, S_1 \to aS_1', S_1' \to a^3Q_2, S_2 \to b^2Q_3, S_3 \to c\}$$

$$P_2 = \{S_2 \to bS_2\}$$
$$P_2 = \{S_3 \to cS_3\}$$

Hľadaný reťazec vygeneruje zadaný PC gramatický systém nasledovne:

$$(S_1, S_2, S_3) \Rightarrow (aS_1', b_S 2, cS_3) \Rightarrow^* (a^n S_1', b^n S_2, c^n S_3) \Rightarrow (a^{n+3} Q_2, b^{n+1} S_2, c^{n+1} S_3)$$
$$\Rightarrow (a^{n+3} b^{n+1} S_2, S_2, c^{n+1} S_3) \Rightarrow (a^{n+3} b^{n+3} Q_3, bS_2, c^{n+2} S_3) \Rightarrow (a^{n+3} b^{n+3} c^{n+2} S_3, bS_2, S_3)$$
$$\Rightarrow (a^{n+3} b^{n+3} c^{n+3}, b^2 S_2, cS_3)$$

Dostávame teda hľadaný jazyk a zapisujeme $L(\Gamma) = \{a^n b^n c^n : n \geq 1\}$. Vidíme, že sila lineárneho gramatického systému je **vyššia** ako sila bezkontextovej gramatiky. Obávané predpoklady, nadobudnuté z výsledku generatívnej sily CD lineárnych gramatických systémov, sa nám podarilo vyvrátiť a ukázať, že aj keď je sila lineárnych pravidiel oproti pravidlám bezkontextovým značne limitujúca, použitím komunikačmých krokov sa dá tento nedostatok obísť a dosiahnuť tak požadovaný výsledok. Po rozbore tohoto príkladu môžeme odvodiť nasledujúci teorém:

$$\mathcal{L}(CF) \subset PC_n LIN$$

Celkovo tak z oboch príkladov môžeme vyvodiť nasledujúci teorém, ktorý hovorí o sile PC a CD lineárnych gramatických systémov:

$$\mathcal{L}(LIN) = CD^{\varepsilon}_{\infty} LIN \subset \mathcal{L}(CF) \subset PC_n LIN$$

4.2.2 Generatívna sila PC lineárnych gramatických systémov v porovnaní s klasickými PC lineárnymi systémami

Predpokladajme pravdivosť nasledujúcich teorémov:

$$PC_\infty LIN \subseteq PC_\infty$$

$$PC_\infty - PC_\infty LIN = \varnothing$$

Ak by boli pravdivé, znamená to, že PC lineárne gramatické systémy sú rovnako silné ako klasické PC gramatické systémy na báze bezkontextových gramatík.

Ak chceme dokázať, resp. vyvrátiť pravdivosť týchto teorémov, musíme sa pozrieť na odlišnosti medzi klasickými a lineárnymi PC gramatickými systémami. Jedinou odlišnosťou sú samozrejme pravidlá. Klasické PC gramatické systémy používajú bezkontextové pravidlá, ktoré sú silnejšie ako lineárne. Ak by sa nám však podarilo ukázať algoritmus, pomocou ktorého dokážeme klasický PC gramatický systém prerobiť na PC lineárny gramatický systém, dokázali by sme pravdivosť horeuvedených teorémov.

Postup pri konverzii klasického PC gramatického systému na PC lineárny gramatický systém je nasledovný:

1. Pre každé pravidlo v každom komponente PC gramatického systému skontrolovať, či je lineárne. Ak nie je, pokračujeme nasledujúcim bodom.

2. Pre každý neterminál A v nelineárnom bezkontextovom pravidle, ktorý sa v ňom vyskytuje viac ako jedenkrát, vytvoríme v PC lineárnom gramatickom systéme nasledujúci komponent: $(A, \{A \to A\})$, ktorý označuje komunikačný symbol Q_x a všetky výskyty neterminálu A v pôvodnom pravidle nahradíme komunikačným symbolom Q_x.

Príklad: Transformujte zadaný PC gramatický systém Γ na PC lineárny gramatický systém Γ_{LIN}.

$$\Gamma = (\{S, A\}, K, \{a, b\}, (S, P_1)), \text{ kde:}$$

$$P_1 = \{S \to AA,\ A \to aAb,\ A \to ab\}$$

Jazyk generovaný týmto systémom je $L(\Gamma) = \{a^n b^n a^m b^m : m, n \geq 1\}$. Napríklad reťazec a^2b^2ab dostaneme nasledovne:

$$S \Rightarrow AA \Rightarrow aAbA \Rightarrow aabbA \Rightarrow aabbab = a^2b^2ab$$

Aplikujme algoritmus popísaný vyššie a dostaneme nasledujúci PC lineárny gramatický systém:

$$\Gamma_{LIN} = (\{S, A\}, K, \{a, b\}, (S, P_1), (A, P_2)), \text{ kde:}$$

$$P_1 = \{S \to Q_2Q_2,\ A \to aAb,\ A \to ab\},\ P_2 = \{A \to A\}$$

K rovnakému reťazcu dospejeme nasledujúco:

$$(S, A) \Rightarrow (Q_2Q_2, A) \Rightarrow (AA, A) \Rightarrow (aAbA, A) \Rightarrow (aabbA, A) \Rightarrow (aabbab, A)$$

Dostávame reťazec a^2b^2ab, teda rovnaký, ako pri použití klasického PC gramatického systému s bezkontextovými pravidlami. Na jednoduchom prípade sme teda ukázali, že je možné transformovať každý PC gramatický systém na PC lineáarny gramatický systém, ktorý generuje rovnaký jazyk.

Pre názornosť ukážka transformácie zložitejšieho PC gramatického systému na PC lineárny gramatický systém (pozn.: v uvedenom PC gramatickom systéme ide o tvar pravidiel kvôli ilustrácii algoritmu, nie o generovaný jazyk).

Príklad: Transformujte zadaný PC gramatický systém Γ na PC lineárny gramatický systém Γ_{LIN}.

$$\Gamma = (\{A, B, C\}, K, \{a, b, c\}, (A, P_1), (B, P_2)), \text{ kde:}$$

$$P_1 = \{A \to abBCbBc,\ B \to CabcQ_2C,\ B \to \varepsilon,\ C \to \varepsilon\},\ P_2 = \{B \to bB\}$$

Použitím algoritmu pre transformáciu PC gramatického systému na PC lineárny gramatický systém dostávame:

$$\Gamma = (\{A, B, C\}, K, \{a, b, c\}, (A, P_1), (B, P_2), (B, P_3), (C, P_4)), \text{ kde:}$$

$$P_1 = \{A \to abQ_3CbQ_3c,\ B \to Q_4abcQ_2Q_4,\ B \to \varepsilon,\ C \to \varepsilon\},\ P_2 = \{B \to bB\},$$

$$P_3 = \{B \to B\},\ P_4 = \{C \to C\}$$

Na príkladoch sme dokázali pravdivosť vyššie uvedených teorémov, a teda generatívna sila PC gramatických systémov a PC lineárnych gramatických systémov s neobmedzeným počtom komponentov je rovnaká, rozdiel jazykov, ktoré generuje klasický PC gramatický systém a PC lineárny gramatický systém, je prázdna množina.

Kapitola 5

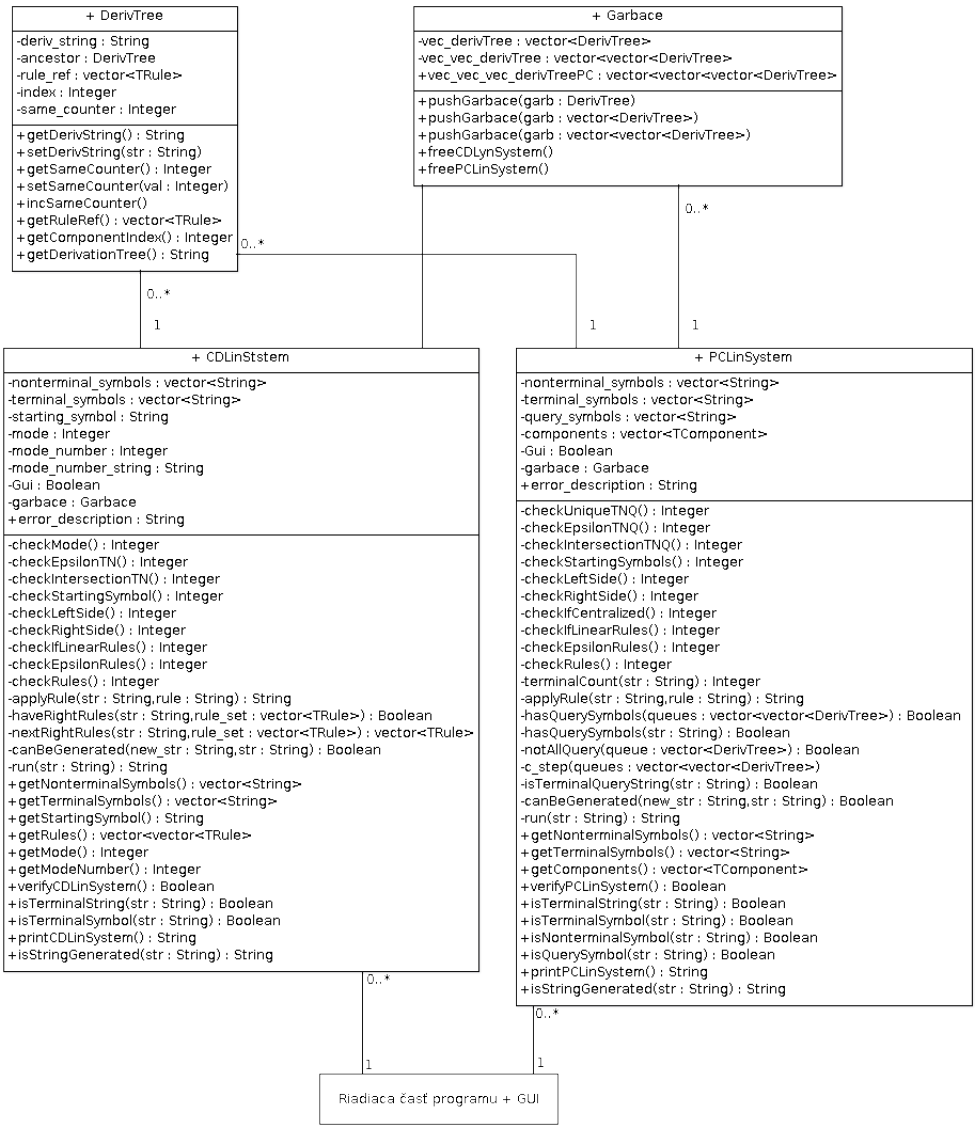
Implementácia lineárnych gramatických systémov

V tejto kapitole transformujeme teoreticky navrhnutý koncept CD a PC lineárnych gramatických systémov na reálnu programovú implemetáciu. Popíšeme algoritmy realizujúce určité činnosti CD lineárnych gramatických systémov a PC lineárnych gramatických systémov a bližšie sa pozrieme na niektoré zložitejšie problémy, ktoré sa pri implementácii objavili. Na záver budú niektoré príklady z teoretickej časti (podkapitola 4.1.1 a 4.2.1) demonštrované v programe Linear Grammar System Creator, ktorý je výsledkom implementačnej časti.

V rámci implementačnej časti boli vytvorené triedy v jazyku C++, ktoré reprezentujú CD a PC lineárny gramatický systém. Tieto triedy je možné použiť pri výstavbe syntaktického analyzátora, pomocou objektov reprezentujúcich CD a PC gramatické systémy. Je možné si nadefinovať vlastné CD alebo PC lineárne gramatické systémy, ktoré následne dokážu vyhodnotiť, či zadaný reťazec nimi je, alebo nie je generovaný. Nad týmito triedami bolo vytvorené užívateľské rozhranie Linear Grammar System Creator pre lepšiu ilustráciu práce implementovaných CD a PC lineárnych gramatických systémov, kde je možné prehliadnuť si, akými derivačnými krokmi daný reťazec vznikal.

5.1 Štruktúra aplikácie

Základné jadro programu ukazuje obrázok 5.1. Tento diagram tried neobsahuje väzbu na grafické užívateľské rozhranie, zobrazuje len objektový návrh jadra programu. Významné funkcie jednotlivých tried budú ďalej popísané v podkapitolách 5.3 a 5.4. Triedy `cdLinSystem` a `pcLinsystem` zodpovedajú implementácii CD a PC lineárneho gramatického systému, trieda `derivTree` reprezentuje reťazce tvorené pri generovaní hľadaného reťazca lineárnym gramatickým systémom a trieda `garbace` je implementácia jednoduchého kvázi garbace collectoru, kde sú ukladané referencie na naalokovanú pamäť, ktorá je podľa potreby uvoľnená.



Obrázek 5.1: Diagram tried aplikácie Linear Grammar System Creator

5.2 Reprezentácia dát

5.2.1 Reprezentácia CD lineárneho gramatického systému

Z definície v podkapitole 3.1 vidíme, že CD lineárny gramatický systém sa skladá z neterminálov, terminálov, štartovacieho symbolu, módu a komponentov – množiny množín pravidiel. Neterminály a terminály sú reprezentované pomocou vektoru reťazcov, štartovací symbol pomocou reťazca. V programovej implementácii je možný výber z dvoch módov, k-krokého a t-módu (ukončovacieho módu), ktoré sú reprezentované zodpovedajúcim celým číslom (makrami) a pri k-krokom móde je potrebné ďalšie celé číslo reprezentujúce k. Komponenty obsahujúce pravidlá, sú reprezentované vektorom vektorov reťazcov, kde tvar pravidla $A \rightarrow x$ odpovedá reťazcu "A x". V konečnej implementácii ide o vektor vektorov štruktúry `TRule`, ktorá okrem reťazca reprezentujúceho celé pravidlo, obsahuje aj zvlášť jeho pravú a ľavú stranu, kvôli lepšej a prehľadnejšej práci s pravidlami.

Reprezentácia PC lineárneho gramatického systému

Podľa definície v podkapitole 3.2 je zrejmé, že pri PC lineárnom gramatickom systému budeme reprezentovať neterminály a terminály rovnako. Reprezentácia komponentov bude iná. Pravidlá budú reprezentované rovnako ako v CD lineárnom gramatickom systéme, taktiež aj jednotlivé štartovacie symboly. Každý komponent má navyše unikátny komunikačný symbol, ktorý ho jednoznačne identifikuje (nie nutne $Q_1, \ldots, Q_n$). Ak spojíme všetky tieto zložky, ktoré tvoria komponent, dostaneme výslednú štruktúru `TComponent` zloženú zo štartovacieho symbolu, komunikačného symbolu a vektoru vektorov pravidiel (štruktúra `TRule`).

5.3 Implementácia metód CD lineárneho gramatického systému

V tejto podkapitole budú predstavené významné a implementačne zaujímavé metódy, ktoré reprezentujú činnosť CD lineárneho gramatického systému.

5.3.1 Kontrola zadania validného CD lineárneho gramatického systému

Predtým, než sa systém začne používať k zisťovaniu, či zadaný reťazec je alebo nie je systémom generovaný, je potrebné systém skontrolovať. K tomuto slúži metóda `verifyCDLinSystem`, ktorá postupne volá privátne metódy, ktoré riešia podproblémy validácie systému. V prvom rade ide o kontrolu validne zadaného módu a kontrolu, či prienik terminálnych a neterminálnych symbolov nie je náhodou neprázdna množina, taktiež, či sa nejedná o multimnožiny. Taktiež sa kontrolujú terminály a neterminály na prítomnosť symbolu 'e', ktorý reprezentuje symbol ε. Kontrolou musí prejsť aj štartovací symbol, a to konkrétne kontrolami dvomi, či sa nachádza v množine neterminálnych symbolov, a či existuje v prvom komponente pravidlo s ľavou stranou rovnajúcou sa štartovaciemu symbolu, čo nás privádza k ďalšej konrole, kontrole pravidiel.

Najprv sa skontroluje ľavá strana pravidiel, teda či ide skutočne o neterminálne symboly, následne pravá strana, kde sa kontroluje, či sa tu vyskytujú skutočne len neterminálne a terminálne symboly, alebo v špeciálnom prípade len symbol 'e' reprezentujúci ε. Poslednou kontrolou je skontrolovať, či sa v každom pravidle vyskytuje najviac jeden neterminálny symbol, podmienka lineárnosti pravidiel.

5.3.2 Kontrola aplikácie pravidiel na reťazec

Pri generovaní hľadaného symbolu začíname štartovacím symbolom, na ktorý aplikujeme pravidlá. Vzniknú nám tak reťazce, ktoré, ak sú zložené len z terminálnych symbolov, porovnáme s hľadaným na zhodu a reťazce obsahujúce neterminálny symbol, na ktoré opäť aplikujeme pravidlá. Ak by sme na daný reťazec aplikovali všetky pravidlá, ktoré sú teoreticky aplikovateľné, počet reťazcov by stúpal exponenciálne a pri hľadaní dlhších reťazcov by nám pravdepodobne došla pamäť.

Aby sme naivne neaplikovali všetky pravidlá, existuje metóda `canBeGenerated`, ktorá zistí, či z daného reťazca je možné získať hľadaný reťazec. Algoritmus je nasledovný:

1. ak je reťazec zložený len z terminálnych symbolov, porovnáme ho s hľadaným, ak sú rovnaké, vrátime **true**, inak **false**.

2. ak počet neterminálnych symbolov v reťazci je väčší ako dĺžka hľadaného reťazca, vrátime **false**.

3. ak sa reťazec skladá len z jedného neterminálu, vrátime **true**.

4. ak sa neterminál vyskytuje v reťazci na prvom mieste, porovnáme vyznačené podreťazce (ilustrujeme príklad, kde hľadaným reťazcom je *baabbab* a náš aktuálny reťazec je *Aaabb*): *A***aabb** *b***aabb***ab*. Pri označení hľadaného reťazca ako `str` a aktuálneho ako `newStr`, dostávame nasledujúci výraz:

   ```
   newStr.substr(1,newStr.size()-1).compare(str.substr(str.size()
               -(newStr.size()-1), newStr.size()-1))
   ```

   Ak sú vyznačené časti rovnaké, vraciame **true**, inak vraciame **false**.

5. ak sa neterminál vyskytuje v reťazci na poslednom mieste, porovnáme vyznačené podreťazce (ilustrujeme príklad, kde hľadaným reťazcom je *aabbab* a náš aktuálny reťazec je *aabbA*): **aabb***A* **aabb***ab*. Pri označení hľadaného reťazca ako `str` a aktuálneho ako `newStr`, dostávame výraz:

   ```
   newStr.substr(0,newStr.size()-1).compare(str.substr(0,newStr.size()-1))
   ```

   Ak sú vyznačené časti rovnaké, vraciame **true**, inak vraciame **false**.

6. ak sa neterminál nevyskytuje v reťazci na prvom ani poslednom mieste, porovnáme vyznačené podreťazce (ilustrujeme príklad, kde hľadaným reťazcom je *aaabbb* a náš aktuálny reťazec je *aaAbb*): **aa***Abb b***aa***abbbab* a *aaA***bb** *aaab***bb**. Ak opäť označíme hľadaný reťazec ako `str` a aktuálny `newStr`, dostávame výraz:

   ```
          newStr.substr(0,index).compare(str.substr(0,index)) &&
        newStr.substr(index+1,newStr.size()-index-1).compare(str
   .substr(str.size()-(newStr.size()-index-1),newStr.size()-index-1))
   ```

   Ak sú vyznačené časti v oboch dvojiciach rovnaké, vraciame **true**, inak vraciame **false**.

Prípadom, ktoré sme ošetrili, sa vymyká ešte jeden, a to, ak by pravidlá, ktoré majú na pravej strane len jeden neterminál a žiadne terminály, mali cyklické závislosti, napr. množina pravidiel $P = \{A \to B,\ B \to C,\ C \to A\}$. Tieto situácie rieši počítadlo, ktoré sa zvýši vtedy, keď reťazec má po prevedení derivačného kroku rovnakú veľkosť ako pred ním. Po prekročení stanovenej hranice je takýto reťazec vyradený z derivačného procesu.

5.3.3 Implementácia hlavného cyklu CD lineárneho gramatického systému

Beh CD lineárneho gramatického systému zabezpečuje metóda `isStringGenerated(string str)`, kde parameter `str` je hľadaný reťazec. Metóda sa stará o správu pamäte a volá privátnu metódu `run(string str)`, v ktorej je implementovaný samotný hlavný cyklus CD lineárneho gramatického systému.

Pre ukladanie generovaných reťazcov sú vytvorené dve fronty, jedna obsahujúca reťazce pred aplikovaním pravidiel a druhá, do ktorej sa uložia reťazce po aplikovaní pravidiel. Popis algoritmu realizujúceho samotný beh je nasledovný:

1. skontroluj, či je hľadaný reťazec zložený len z terminálov, ak nie, vráť prázdny reťazec.
2. vytvor prvý reťazec zodpovedajúci štartovaciemu symbolu a vlož ho do prvej fronty, ak je mód CD lineárneho gramatického systému k-kroký, vynuluj počítadlo.
3. ak je prvá fronta prázdna, vráť prázdny reťazec.
4. pre každý prvok z prvej fronty rozhodni, či je zložený len z terminálnych symbolov, ak áno, porovnaj ho s hľadaným reťazcom a v prípade zhody vráť reťazec popisujúci celý derivačný proces, inak pokračuj ďalším bodom
5. ak je mód CD lineárneho gramatického systému ukončovací, aplikuj na každý prvok z prvej fronty aktuálnu sadu pravidiel, pokiaľ je to možné, inak nájdi nasledujúcu možnú sadu a aplikuj tú. Nové reťazce po aplikácii pravidiel ulož do druhej fronty.
6. ak je mód CD lineárneho gramatického systému k-kroký, zvýš hodnotu počítadla a porovnaj ju s hodnotou k. Ak sú si rovné, počítadlo nastav na hodnotu 1 a aplikuj na každý prvok z prvej fronty nasledujúcu sadu pravidiel, inak aplikuj aktuálnu sadu. Nové reťazce po aplikácii pravidiel ulož do druhej fronty.
7. Vymaž obsah prvej fronty a prekopíruj do nej obsah druhej fronty, následne obsah druhej fronty vymaž. Pokračuj bodom 3.

Po ukončení behu privátnej metódy `run(string str)` je uvoľnená dynamicky naalokovaná pamäť a vrátený výsledok vyprodukovaný touto metódou.

5.4 Implementácia metód PC lineárneho gramatického systému

V tejto podkapitole ukážeme metódy, ktoré reprezentujú činnosť PC lineárneho gramatického systému. Pôjde hlavne o algoritmus generovania hľadaného reťazca a heuristiky urýchľujúce generovanie takéhoto reťazca.

5.4.1 Kontrola zadania validného PC lineárneho gramatického systému

Kontrola validity PC lineárneho gramatického systému je v mnohých veciach podobná až rovnaká ako pri kontrole CD lineárneho gramatického systému. Odlišnosť pri kontrole neterminálnych a terminálnych symbolov je tá, že je potrebné vziať do úvahy aj komunikačné symboly a skontrolovať, či množiny neterminálnych, terminálnych a komunikačných symbolov sú párovo disjunktné, taktiež či nejde o multimnožiny. Keďže program podporuje len centralizované PC lineárne gramatické systémy, je potrebná aj kontrola na výskyt komunikačných symbolov v komponentoch. Existencia pravidla pre štartovací symbol sa testuje pri každom komponente na rozdiel od CD lineárnych gramatických systémov, kde kontrola prebieha len v prvom komponente. Všetky tieto kontroly realizuje metóda `verifyPCLinSystem`, ktorá volá privátne metódy implementujúce riešenia jednotlivých problémov.

5.4.2 Kontrola aplikácie pravidiel na reťazec

Oproti kontrole v CD lineárnych gramatických systémoch je pri PC lineárnych gramatických systémoch situácia o niečo zložitejšia. V prvom rade je potrebné vziať do úvahy komunikačné symboly. Ak sa v reťazci vyskytujú, nebudeme takýto reťazec kontrolovať hneď, ale počkáme, kým sa za komunikačný symbol nahradí zodpovedajúci reťazec.

Existencia komunikačných symbolov prináša situácie, kedy je možné dostať v reťazci viacero neterminálnych symbolov, musíme teda zvoliť iný prístup, ako keď sme očakávali maximálne jeden. Reťazec si rozdelíme na časti, tak, že prvú časť vezmeme od začiatku reťazca po prvý neterminálny symbol, ďalšie časti od jedného neterminálneho symbolu po ďalší neterminálny symbol a nakoniec od posledného neterminánehо symbolu po koniec reťazca (neterminálne symboly do jednotlivých častí nezaratávame). V hľadanom reťazci postupne vymazávame jednotlivé časti, ktoré sme získali, prvú a poslednú je možné skontrolovať a vymazať pochopiteľne len na začiatku a konci hľadaného reťazca. Ak sa tam nejaká časť nenachádza, znamená to, že z daného reťazca nie je možné získať hľadaný reťazec, a teda ho z procesu aplikácie pravidiel vylučujeme. Napr. ak by hľadaným reťazcom bol reťazec $aabbababaaabaa$ a my sa rozhodujeme, či ho môžeme získať z reťazca $aaAababBaabAa$, kde množina terminálnych symbolov je $\{a, b\}$ a množina neterminálnych symbolov je $\{A, B\}$, rozdelíme najprv reťazec na časti aa, $abab$, aab, a. Pokúsime sa porovnať začiatok a koniec hľadaného reťazca s prvou a poslednou časťou, **aa***bbababaaaba***a** a po vymazaní nám ostáva $bbababaaaba$. Teraz hľadáme jednotlivé časti, najprv teda nájdeme a vymažeme $abab$, ostane nám $bbaaaba$, vezmeme nasledujúcu a v tomto príklade poslednú časť aab, ktorú opäť nájdeme a po vymazaní nám ostáva $bbaa$. V danom príklade sa nám podarilo úspešne nájsť všetky časti. V prípade, že by sa to nepodarilo, nie je z daného reťazca možné vygenerovať hľadaný reťazec.

Pokiaľ máme reťazec zložený len z jedného neterminálneho symbolu, postupujeme rovnako ako v prípade CD lineárnych gramatických systémov, aby sme ošetrili možné cyklické závislosti.

5.4.3 Implementácia hlavného cyklu PC lineárneho gramatického systému

Beh PC lineárneho gramatického systému zabezpečuje rovnomenná metóda ako v prípade CD gramatického systému, `isStringGenerated(string str)`, ktorá sa taktiež stará o správu pamäte a volá privátnu funkciu `run(string str)`, ktorá implementuje hlavný cyklus PC gramatického systému.

Pre ukladanie generovaných reťazcov nestačí vytvoriť dve fronty, ako tomu bolo v prípade CD lineárnych gramatických systémov, ale vytvoriť dve fronty pre každú komponentu PC lineárneho gramatického systému. V prvej budú opäť ukladané reťazce pred aplikovaním pravidiel a v druhej reťazce po aplikovaní pravidiel. Výsledný algoritmus potom vyzerá takto:

1. skontroluj, či je hľadaný reťazec zložený len z terminálov, ak nie, vráť prázdny reťazec.
2. vytvor prvé reťazce zodpovedajúce štartovacím symbolom jednotlivých komponentov a vlož ich do prvých front zodpovedajúcich komponentov.
3. ak je niektorá prvá fronta prázdna, vráť prázdny reťazec.
4. preveď komunikačný krok – nahraď komunikačné symboly za zodpovedajúce reťazce
5. pre každý prvok z prvej fronty z prvého komponentu rozhodni, či je zložený len z terminálnych symbolov, ak áno, porovnaj ho s hľadaným reťazcom a v prípade zhody vráť reťazec popisujúci celý derivačný proces, inak pokračuj ďalším bodom
6. na každý prvok z prvých front komponentov aplikuj pravidlá. Nové reťazce po aplikácii pravidiel ulož do druhých front zodpovedajúcich komponentov.
7. vymaž obsahy prvých front a prekopíruj do nich obsahy druhých front, následne obsahy druhých front vymaž. Pokračuj bodom 3.

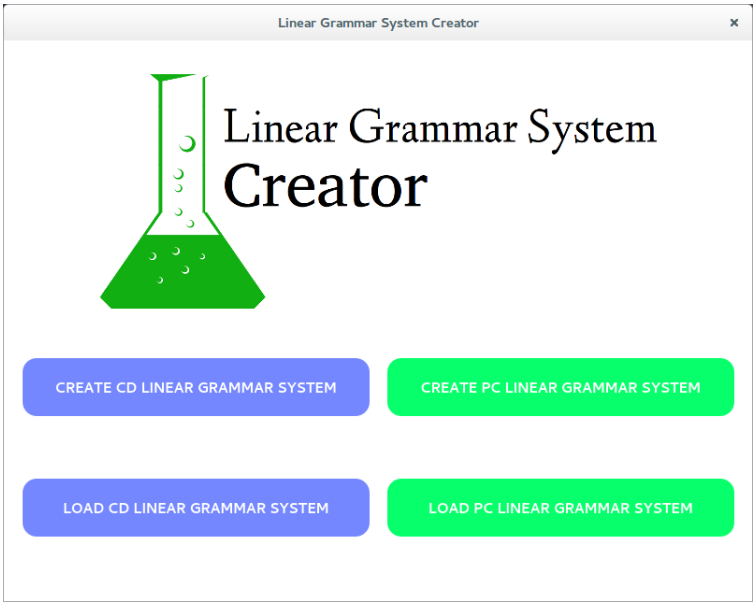
5.5 Linear Grammar System Creator

Linear Grammar System Creator je program využívajúci triedy vytvorené pre implementáciu CD a PC lineárnych gramatických systémov. Ide o nadstavbu s grafickým užívateľským rozhraním. Grafické užívateľské rozhranie bolo vytvorené pomocou frameworku Qt (viac viď [1]). Linear Grammar System Creator slúži pre demonštráciu teoretických poznatkov v praxi.

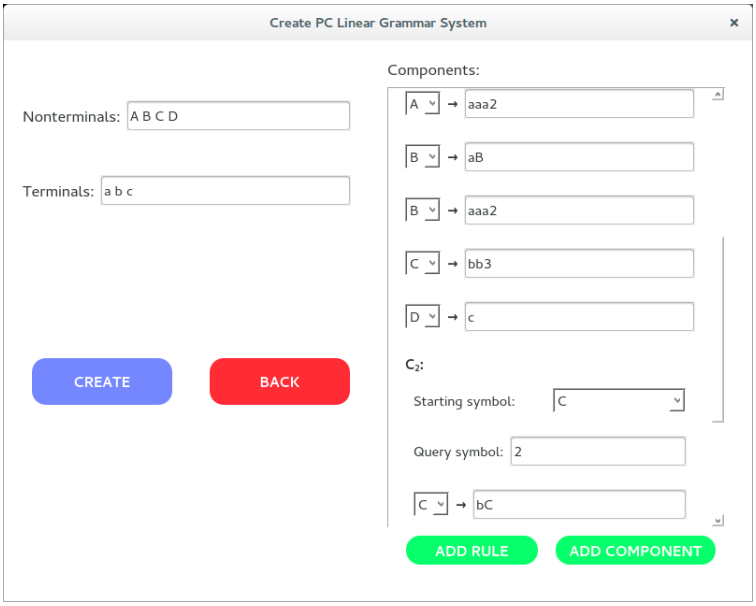
V programe je možné vytvoriť si vlastný CD, či PC lineárny gramatický systém, následne s ním pracovať, prípdne si ho uložiť v podobe XML súboru a v budúcnosti načítať. Po zadaní lineárneho gramatického systému program skontroluje sémantickú validitu zadaných údajov, syntax kontroluje už pri zadávaní. Po úspešnom vytvorení lineárneho gramatického systému je možné testovať, či zadaný reťazec je daným systémom generovaný alebo nie. V prípade, že je, program vypíše celý derivačný proces, teda kroky, ktoré viedli k vygenerovaniu výsledného reťazca lineárnym gramatickým systémom. Lineárne gramatické systémy je možné kedykoľvek upravovať.

Na obrázku 5.5 je vidieť, ako by sme nadefinovali a pracovali s PC lineárnym gramatickým systémom z druhého príkladu v teoretickej časti v podkapitole 4.2.1, ktorý generuje jazyk $\{a^n b^n c^n : n \geq 1\}$ a vyskúšali ním vygenerovať reťazec *aaabbbccc*, ktorý týmto systémom generovaný je a reťazec *aaabbbcccc*, ktorý, naopak, PC lineárny gramatický systém negeneruje.

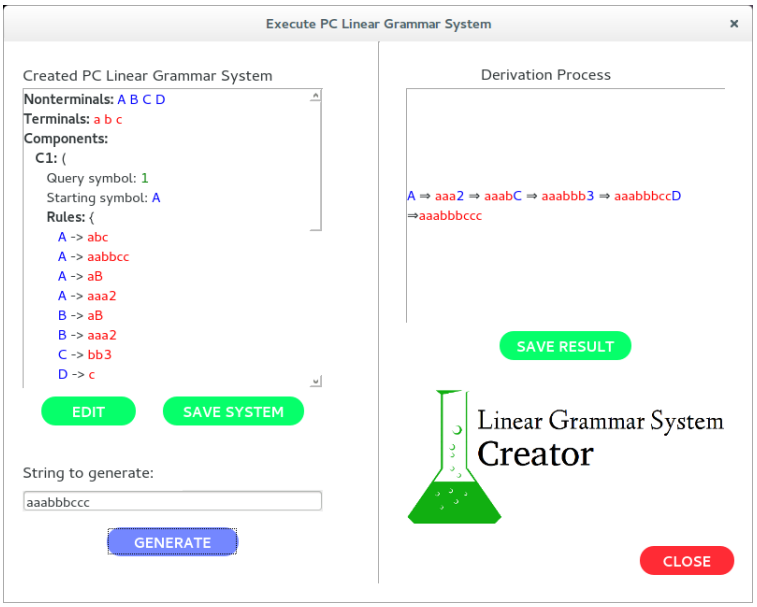
Z hlavného menu je možné otvoriť si viacero okien a vytvoriť a pracovať súčasne s viacerými lineárnymi gramatickými systémami. Výpočet, resp. beh lineárneho gramatického systému, je stále spúšťaný na novom vlákne.



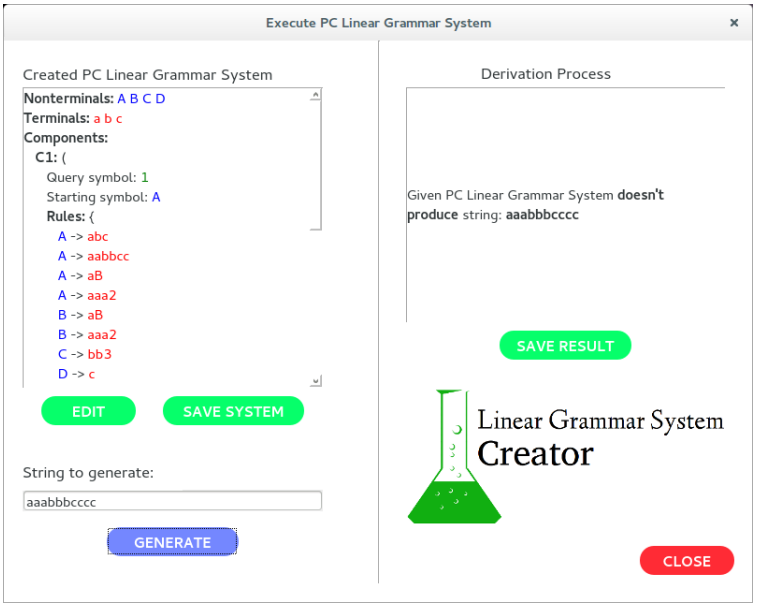
Obrázek 5.2: Úvodné okno



Obrázek 5.3: Vytvorenie PC lineárneho gramatického systému



Obrázek 5.4: Ukážka reťazca, ktorý PC lineárny gramatický systém generuje



Obrázek 5.5: Ukážka reťazca, ktorý PC lineárny gramatický systém negeneruje

Kapitola 6

Záver

Cieľom tejto práce bolo naštudovať si problematiku gramatických systémov a následne ich modifikovať na systémy, ktoré nebudú založené na klasickej báze bezkontextových gramatík, ale obmedziť silu pravidiel na lineárne gramatiky. Bolo potrebné zistiť vlastnosti CD a PC lineárnych gramatických systémov, hlavne generatívnu silu takto modifikovaných sytémov a programovo ich implemetovať.

Výsledky teoretickej časti boli prekvapivé pri CD aj PC gramatických systémoch. Zistili sme, že CD gramatické systémy majú rovnakú generatívnu silu ako lineárne gramatiky a že ani existencia ďalších komponentov a ľubovoľného módu CD linárneho gramatického systému nie je dostačujúca na dosiahnutie podobných výsledkov ako pri klasických CD lineárnych gramatických systémoch, ktorých generatívna sila je vyššia ako bezkontextová, obe tieto skutočnosti sa nám podarilo demonštrovať na príklade. Na rozdiel od zistení pri CD lineárnych gramatických systémoch bola situácia markantne odlišná pri PC lineárnych gramatických systémoch. Ukázali sme, že ich sila je vyššia ako lineárnych gramatík, dokonca aj vyššia ako bezkontextových gramatík. Ďalej sme ukázali, že ľubovoľný klasický PC gramatický systém dokážeme transformovať na PC lineárny gramatický systém, a tým dokázať, že ich generatívna sila je pri neobmedzenom počte komponentov rovnaká.

Teoretické poznatky, nadobudnuté v teoretickej časti, sme transformovali do programovej implementácie a vytvorili sme triedy, ktoré implemetujú CD a PC lineárne gramatické systémy a ktoré je možné využiť pri tvorbe syntaktického analyzátora. Pre demonštráciu bol vytvorený program Linear Grammar System Creator, ktorý je nadstavbou nad týmito triedami s grafickým užívateľským rozhraním, kde je možné vytvárať, editovať, ukladať, načítať a pracovať s CD, či PC lineárnymi gramatickými systémami a ktorý hlavne zodpovedá otázku syntaktickej analýzy, či je hľadaný reťazec syntakticky správny (či je generovaný), alebo nie je. Do budúcnosti by možnými rozšíreniami programu mohli byť implementácia najmenej k-krokého a nanajvýš k-krokého módu pri CD lineárnom gramatickom systéme a možnosť tvorby necentralizovaného PC lineárneho gramatického systému.

Prínos práce je jednak v oblasti teoretickej informatiky zisteniami o vlastnostiach CD a PC lineárnych gramatických systémov, ale aj implementačná časť ukázala, že je možné navrhnúť aj jazyky, ktoré nie sú generované bezkontextovou gramatikou a môžu mať vysokú vyjadrovaciu schopnosť. Preklad takýchto jazykov je stále náročný, avšak pri zlepšovaní navrhnutých heuristík a vytváraní nových je možné, že v budúcnosti sa bude siahať po gramatických systémoch aj v implementácii prekladačov a nebudú len intelektuálnou hrátkou v oblasti teoretickej informatiky.

Literatura

[1] Qt Project – Documentation [online]. Dostupné na: `http://www.qt-project.org/doc/`, Posledná zmena 2014-02-14 [cit. 2014-04-17].

[2] Csuhaj-Varjú E., e. a.: *Grammar systems: A Grammatical Approach to Distribution and Cooperation*. Langhorne, Pennsylvania: Gordon and Breach, 1994, ISBN 28-812-4957-4.

[3] Daintith J., E., Wright: *A dictionary of computing*. New York: Oxford University Press, 6 vydání, 2008, ISBN 01-992-3401-9.

[4] Lukáš, R.: *Multigenerativní gramatické systémy*. Brno, FIT VUT v Brně: disertační práce, 2006.

[5] Meduna, A.: *Automata and Languages: theory and applications*. London: Springer, 2000, ISBN 18-523-3074-0.

[6] Meduna, A.: CD Grammar Systems [online]. Dostupné na: `www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:09-cdgspres.pdf`, Posledná zmena 2009-10-21 [cit. 2014-01-12].

[7] Meduna, A.: PC Grammar Systems [online]. Dostupné na: `http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:10-pcgspres.pdf`, Posledná zmena 2009-10-21 [cit. 2014-01-16].

[8] Meduna, A.: Matrix Grammars [online]. Dostupné na: `http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:04-matgrampres.pdf`, Posledná zmena 2009-10-21 [cit. 2014-01-18].

Příloha A

Obsah CD

Priložené CD obsahuje nasledujúcu adresárovú štruktúru:

`bin`	adresár obsahujúci binárny spustiteľný súbor po preklade
`examples`	adresár obsahujúci príklady PC a CD lineárnych gramatických systémov definovaných v XML súboroch
`obj`	adresár obsahujúci objektové súbory po preklade
`src`	adresár obsahujúci zdrojové súbory
`Makefile`	súbor pre preklad a spustenie programu
`technicka_sprava.pdf`	text bakalárskej práce

Příloha B

Inštalácia

Program využíva Qt framework pre grafické užívateľské rozhranie, konkrétne verziu 5.2.0, ktorá je dostupná aj na školskom serveri Merlin. Pre inštaláciu stačí skopírovať adresárovú štruktúru z CD do požadovaného adresára a preklad previesť pomocou príkazu `make`, ktorý prevedie preklad programu za pomoci programu `qmake`. Pokiaľ sa preklad uskutočňuje na školskom serveri Merlin alebo na počítači s viacerými verziami Qt frameworku, pričom prioritne je volaný program `qmake` staršej verzie ako je verzia 5.2.0, je potrebné zavolať program `qmake` explicitne pomocou absolútnej cesty (v prípade prekladu na školskom serveri Merlin stačí zakomentovať v Makefile riadok s príkazom `qmake` a odkomentovať riadok s explicitne zadanou absolútnou cestou `/usr/local/share/Qt-5.2.1/5.2.1/gcc_64/bin/qmake`).

Příloha C

Manuál k programu

Po spustení programu Linear Grammar System Creator sa zobrazí úvodná obrazovka s hlavným menu, kde je možné vybrať si jednu zo štyroch hlavných možností: vytvorenie nového CD alebo PC lineárneho gramatického systému alebo načítanie už vytvorených. Hlavné menu je stále prístupné a je možné otvoriť z neho viacero okien a pracovať tak súčasne s viacerými lineárnymi gramatickými systémami.

Pri voľbe vytvorenia nového CD či PC lienárneho gramatického systému je potrebné dodržiavať nasledujúce pravidlá: neterminály a terminály sa môžu skladať len z jedného znaku anglickej abecedy, pričom sa rozlišujú veľké a malé písmená, jednotlivé znaky sa oddeľujú medzerou. Predvolený komunikačný symbol je možné zmeniť na vlastný, pričom platia rovnaké pravidlá ako pri neterminálach a terminálach, naviac je možné okrem písmen anglickej abecedy použiť pre označenie aj čísla od 0 do 9. Pravá strana pravidiel môže pochopiteľne pozostávať len z neterminálov, terminálov a v prípade PC lineárneho gramatického systému aj z komunikačných symbolov, medzi znakmi sa nezadáva žiadna medzera. Pri vytváraní PC gramatického systému je potrebné rešpektovať, že program podporuje len centralizované PC lineárne gramatické systémy bez návratu. Program informuje užívateľa o syntaktických chybách už pri vytváraní systému v stavovom riadku v spodnej časti okna. Sémantická stránka sa skontroluje po požiadavke pre vytvorenie systému, pričom je užívateľ na prípadné chyby upozornený rovnako, ako v prípade syntaktických.

Po úspešnom vytvorení lineárneho gramatického systému sa zobrazí okno aplikácie, kde je možné zistiť, či vytvorený lineárny gramatický systém generuje alebo negeneruje zadaný reťazec. Výsledok je možné si uložiť, rovnako ako aj vytvorený lineárny gramatický systém. Na výber je taktiež možnosť editácie systému, teda návrat do predchádzajúceho okna.