VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

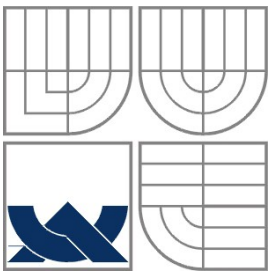
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

SYSTÉMY PARALELNÍCH GRAMATIK

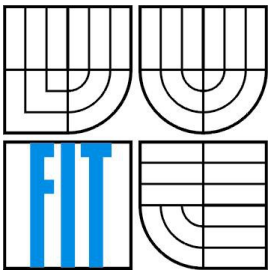
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JIŘÍ SKÁCEL
AUTHOR

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

SYSTÉMY PARALELNÍCH GRAMATIK

SYSTEMS OF PARALLEL GRAMMARS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ SKÁCEL
AUTHOR

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

BRNO 2014

Abstrakt

Tato práce definuje kooperující distribuované gramatické systémy založené na E0L systémech namísto bezkontextových gramatik. Zkoumá jejich generativní sílu především vzhledem k ET0L systémům. Připomíná hlavní závěry o CD GS, následně definuje upravené systémy a poskytuje důkazy o síle jednotlivých jejich derivačních módů. Většina módů se silou rovná ET0L systémům, s výjimkou ukončovacího módu, pro který je poskytnut důkaz o větší síle. Prezentováno je několik vysvětlených příkladů nových systémů.

Abstract

This article defines cooperating distributed grammar systems with E0L components instead of context-free grammars and discusses its generative power mainly in respect to ET0L systems. It recapitulates results about CD GS, then defines modified systems and shows that this combination has in most derivative modes equal strength to ET0L, except for terminating mode, which is shown to be more powerful. There are also explained examples of this new kind of systems.

Klíčová slova

CD GS, ET0L, Lindenmayerovy systémy, gramatické systémy, paralelní gramatiky

Keywords

CD GS, ET0L, Lindenmayer systems, grammar systems, parallel grammars

Citace

Jiří Skácel: Systémy paralelních systémů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Systémy paralelních gramatik

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením profesora Alexandera Meduny. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Skácel
20.května 2014

Poděkování

Zde bych rád poděkoval vedoucímu mé bakalářské práce, profesoru Alexanderu Medunovi, za jeho náměty k přemýšlení, čas a vedení.

Obsah

1 Úvod

Formální jazyky představují matematický základ informačních technologií. Přináší širokou hierarchii nástrojů jako jsou gramatiky, automaty nebo různé systémy. Tato hierarchie je postavena na porovnávání síly těchto nástrojů, tedy schopnosti generovat formální jazyky se specifickými podmínkami. S tímto jde ruku v ruce i složitost těchto nástrojů a snadnost jejich použití. Používaným kompromisem jsou bezkontextové gramatiky. Z nich se později začaly vytvářet systémy – konkrétně důležité pro tuto práci jsou *E0L systémy* [7] a *kooperující distribuované gramatické systémy* (CD GS) [1].

Myšlenkou za prvním z nich je zavedení paralelismu, který je zejména v přírodě všudypřítomný. Jako první byly zavedeny 0L systémy teoretickým biologem Aristidem Lindenmayerem, po němž jsou také tyto systémy pojmenované, pro modelování růstu řas a jiných jednoduchých mnohobuněčných organismů. Tyto *Lindenmayerovy systémy* využívají pouze bezkontextová pravidla a nerozlišují terminály a neterminály. Existují i Lindenmayerovy systémy využívající kontext, ale ty nás v této práci nebudou zajímat. Rozšíření 0L systémů zahrnují *deterministické D0L systémy*, *E0L systémy* se zavedením *neterminálů* a *tabulkové T0L systémy*, které rozdělují pravidla do skupin, přičemž v jednom kroku se používají pravidla pouze v rámci jedné skupiny. Spolu se svými kombinacemi tvoří tyto tří rozšíření nelineární hierarchii [7, s. 20]. V této práci vycházím z neterminálových E0L systémů a nové systémy porovnávám s *tabulkovými neterminálovými ET0L systémy*, které jsou z nich nejsilnější.

Druhé systémy – CD GS – vznikly snahou o zachycení distribuce práce mezi více agentů a jejich spolupráce na společném řešení. Z agentů je vždy aktivní jeden, jenž vede další postup a využívá své znalosti pro změnu stavu. Výměna tohoto vedoucího je předem dána *protokolem pro spolupráci*, jež v literatuře obyčejně nedeterministicky vybírá dalšího agenta a zastavuje ho po určitém počtu kroků nebo po splnění určitých podmínek. Tyto systémy jsou zpravidla založené na bezkontextových gramatikách, jelikož systémy ostatních gramatik Chomského hierarchie nepřekračují svou sílu [1, s. 162].

Lindenmayerovy systémy i kooperující distribuované gramatické systémy jsou do velké míry už prozkoumány i porovnány navzájem [5]. Dosud ale nebylo řádně definováno jejich spojení, konkrétně *kooperující distribuované gramatické systémy založené na E0L gramatikách*. Právě toto jsem si vzal za cíl v této práci. Hlavní otázkou je, jak se změní síla CD systémů, pokud v každém kroku místo aplikace jednoho pravidla na nedeterministicky vybraný symbol, aplikujeme nějaké pravidlo na každý symbol z aktuálního řetězce.

Ve druhé kapitole krátce shrnu definici a poznatky o CD GS spolu s několika příklady, zejména jako krátký úvod do problematiky, ze které vycházím. Ve třetí kapitole pak definuji nové systémy a rozeberu příklady. Ve čtvrté kapitole se věnuji zkoumání generativní síly. Uvádím závěry a jejich důkazy pro ekvivalence různých derivačních módů a omezení počtu gramatik. V poslední části kapitole se věnuji ukončovacímu módu, který je nejzajímavější,

protože jeho síla je větší než síla ostatních módů. V páté kapitole se věnuji implementaci představených algoritmů v demonstračním skriptu, který vytváří ekvivalentní systémy se zadaným a také zkouší získat příklady řetězců, které zadaný systém generuje.

1.1 Základní pojmy a definice

V této práci předpokládám základní znalosti z teorie formálních jazyků na úrovni bezkontextových gramatik. Velmi dobrým úvodem do této problematiky je [9], ze které také přebírám notace. Znalosti Lindenmayerových systémů a CD systémů nejsou nezbytné, ale velmi doporučené. Zde stručně shrnu pojmy, které v této práci budu používat.

Abeceda Σ je konečná neprázdná množina *symbolů*. Sekvence symbolů tvoří *řetězec*. *Prázdný řetězec* ϵ je řetězec, který neobsahuje žádný symbol.

Nad abecedou Σ je ϵ řetězcem. Dále je řetězcem xy, kde x je řetězcem a y je znak v abecedě Σ. *Konkatenací* řetězců x a y je řetězec xy.

Mocnina řetězce x, značená x^n je definována pro $n=0$ jako $x^0=\epsilon$, pro kladná n jako $x^n=xx^{n-1}$.

Množinu všech řetězců nad abecedou Σ značíme Σ^*. *Množinu všech neprázdných řetězců* pak Σ^+. *Jazykem* nad abecedou Σ chápeme libovolnou podmnožinu Σ^*.

2 CD gramatický systém

Kooperující distribuované gramatické systémy je možno chápat jako spojení několika bezkontextových gramatik – *komponent systému*. Původní motivací pro jejich zavedení byl model řešení problémů pomocí *tabule* [1, s. 158]. Principem je spolupráce několika agentů (v případě CD GS gramatik), kteří vždy po jednom přistupují k tabuli, na které je počáteční problém, a podle svých možností pokračují v řešení problému na tabuli aplikováním svých znalostí. Po určitém počtu kroků se agent od tabule vzdálí a nabídne možnost pokračovat ostatním agentům. *Protokol spolupráce* určuje, kdy je možné se vyměnit, a může i určit, kdo má dále pokračovat.

V CD GS se nedeterministicky vybírají gramatiky systému, které poté aplikují svá pravidla. Odstoupení, neboli *přepnutí komponenty*, je určeno módem, který je použit při generování. V literatuře se obvykle zavádí několik stálých módů, čehož se budu držet i já.

Neterminály značí otevřené problémy, které musí být ještě řešeny. Aplikací pravidel se může jeden problém převést na jiný, který aktuální agent není schopen řešit, ale některý z ostatních spolupracujících agentů může mít znalosti o tomto problému. Toto je jediná forma komunikace mezi komponentami systému, tedy pomocí prostředí – tabule. *Terminály* jsou pak chápany jako části konečného řešení, o kterém už žádný agent nemá pochybnosti.

Tato kapitola je pouze krátkým úvodem do CD GS. Mnohé definice jsou obdobné jako u později zavedených CDE0L systémů. Abych se neopakoval a aby nově zavedené systémy byly definované na jednom místě, jsou zde uvedeny jen ty nejdůležitější definice.

2.1 Definice CD GS

Kooperující distribuovaný gramatický systém stupně $n : n \geq 1$ definujeme jako ($n+3$)-tici:

$$\Gamma = (N, T, S, P_1, \ldots, P_n),$$

kde N je abeceda neterminálů,

T je abeceda terminálů a tyto abecedy jsou disjunktní $N \cap T = \emptyset$,

$S : S \in N$ je axiom,

$P_i : 1 \leq i \leq n$ je množina pravidel ve tvaru $a \to x : a \in N, x \in (N \cup T)^*$ a nazývá se komponenta systému Γ .

Dále se v literatuře definují *derivační módy*. Tyto módy určují kolik derivačních kroků je třeba udělat před přepnutím komponenty. Zavadí se podmínka na alespoň k *kroků* ($\geq k$), *právě* k *kroků* ($= k$), *nejvíce* k *kroků* ($\leq k$) a *ukončovací mód* (t), který umožňuje přepnutí pouze pokud už nelze dále použít stávající komponentu. Také se využívá *mód bez omezení* ($*$). Korektní definice jsou uvedeny v kapitole 3.2.

Jazyk generovaný systémem rozlišujeme podle módu. Řetězce takového jazyka jsou řetězce terminálů, které získáme postupným uplatňováním právě jednoho derivačního módu. Podrobnější definice je uvedena v kapitole 3.3.

2.2 Shrnutí generativní síly

Generativní síla CD GS založených na bezkontextových gramatikách tvoří komplexní hierarchii. V rámci porovnání s nově zavedenými systémy nás budou zajímat hlavně tyto tři věty:

1. $\mathscr{L}(CF) = CD_{\infty}(f)$, pro $f \in \{*, =1, \geq 1\} \cup \{\leq k : k \geq 1\}$,
2. $\mathscr{L}(CF) = CD_1(f) \subset CD_2(f) \subseteq CD_r(f) \subseteq CD_{\infty}(f) \subseteq \mathscr{L}(M)$, pro $r \geq 3$, $f \in \{=k, \geq k : k \geq 2\}$,
3. $\mathscr{L}(CF) = CD_1(t) = CD_2(t) \subset CD_3(t) = CD_{\infty}(t) = \mathscr{L}(ET0L)$.

Podle první věty žádný z módů obsahující možnost přepnout komponentu po jednom kroku nezvyšuje sílu původních gramatik. Není důvod zavádět rozdělení pravidel do komponent, pokud můžeme kdykoli vybrat libovolné pravidlo z libovolné komponenty. Druhá věta stanovuje postupující hierarchii pro vícekrokové módy v alespoň dvoukomponentních systémech. Třetí věta rozděluje systémy do dvou komponent, které nemají zvýšenou sílu, a na ostatní, které jsou rovné s ET0L systémy. Tento mód tedy může jednoduše simulovat paralelismus.

2.3 Příklady

Pro názornost a hlubší pochopení zde uveďme dva příklady těchto klasických systémů i s odvozením jazyků, jež generují.

2.3.1 Jazyk $a^i b^i c^i$

První systém definujme jako $\Gamma_1 = (N, T, S, P_1, P_2)$, kde

$N = \{S, A, \overline{A}, B, \overline{B}\}$,

$T = \{a, b, c\}$,

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

$P_2 = \{$
$A \to a\overline{A}b$,
$B \to \overline{B}c$,
$\}$,

$P_3 = \{$
$\overline{A} \to \epsilon$,
$\overline{B} \to \epsilon$
$\}$.

Generované jazyky jsou:

$L_f(\Gamma_1)_{f\in\{*,=1,\geq 1\}\cup\{\leq k:k\geq 1\}}\{a^i b^i c^j : i, j \geq 1\}$,
$L_{=2}(\Gamma_1) = L_{\geq 2}(\Gamma_1) = L_t(\Gamma_1) = \{a^i b^i c^i : i \geq 1\}$,
$L_{=k}(\Gamma_1)_{k\geq 3} = L_{\geq k}(\Gamma_1)_{k\geq 3} = \emptyset$.

Jako první prozkoumáme jednokrokové módy. Silou se neliší od bezkontextových gramatik, protože rozdělení na komponenty nehraje roli, pokud nemáme omezení jejích přepínání. Z počátečního znaku S tedy získáme řetězec AB , který pomocí opakovaného užití druhé a první komponenty můžeme změnit na $a^i \overline{A} b^i \overline{B} c^j$. Pak už jen pomocí třetí komponenty smažeme neterminály a dostaneme výsledný řetězec $a^i b^i c^j$.

Zajímavějším jazykem je ten generovaný dvoukrokovými módy $=2$ a ≥ 2 . V každé komponentě musíme použít dvě pravidla, proto v tomto případě musíme při použití pravidla pro generovaní a a b použít i pravidlo pro c . Získáme tedy jazyk $a^i b^i c^i$.

Pro více než dvoukrokové módy dostaneme prázdné jazyky. Kvůli tomu, že v generovaném řetězci je nejvýše po jednom neterminálu A a B a ve druhé komponentně jsou jen pravidla pro přepis těchto znaků, můžeme tuto komponentu použít nejvíce dvakrát po sobě, což je v rozporu s principem vícekrokových módů.

Ukončovací mód díky rozdělení systému na tři komponenty funguje obdobně jako dvoukrokové módy. V rámci jednoho použití komponenty aplikujeme analogická pravidla na A a B a poté se derivace zastaví a dovolí přepnutí. Nutně tedy dostaneme jazyk $a^i b^i c^i$.

2.3.2 Jazyk $a^i b^{2^i}$

Druhý systém definujeme jako $\Gamma_2 = (N, T, S, P_1, P_2, P_3)$, kde
$N = \{S, A, \overline{A}, B, \overline{B}\}$,
$T = \{a, b\}$,

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

$P_2 = \{$
$A \to a\overline{A},$
$B \to \overline{B}\,\overline{B}$
$\},$

$P_3 = \{$
$A \to \epsilon,$
$B \to b$
$\}.$

Generované jazyky jsou:
$L_f(\Gamma_2)_{f\in\{*,=1,\geq 1\}\cup\{\leq k:k\geq 1\}} = \{a^i b^j : i \geq 0, j \geq 1\}$,
$L_t(\Gamma_2) = \{a^i b^{2^i} : i \geq 0\}$,
$L_{=k}(\Gamma_2)_{k\geq 2} = L_{\geq k}(\Gamma_2)_{k\geq 2} = \emptyset$.

Jednokrokové módy k řetězci AB zleva připisují a a nakonec mažou znak A , zprava pak připisují B , které pak přepisují na b . Výsledný jazyk je jednoduchý – $a^i b^j$, přičemž počet b je nenulový.

Vícekrokové módy nás v tomto případě nezajímají, protože neprojdou ani přes první komponentu. Po aplikaci pravidla $S \to AB$, nemáme v této komponentě další pravidla, která bychom mohli použít, takže nemůžeme vygenerovat žádný řetězec.

Ukončovací mód postupuje přes první a druhou komponentu, v nichž se po jedné aplikaci přidá jeden znak a a každý neterminál B se zdvojí. Po přepsání všech symbolů B se přepne komponenta na první a všechny neterminály s pruhem se zpátky „odblokují“, aby bylo možné znovu použít druhou komponentu pro generování delších řetězců. Pokud se použije třetí komponenta místo druhé, smažeme tím neterminály a získáme řetězec terminálů a tedy i jazyk $a^i b^{2^i}$.

3 CDE0L gramatický systém

Hlavním cílem této práce je definovat a prozkoumat CD gramatické systémy založené na paralelních gramatikách, konkrétně E0L gramatikách. Ty samy používají bezkontextová pravidla, a tak se z hlediska definic neuskuteční příliš změn. Využijeme však konvenci z Lindenmayerových systémů ohledně definice nad celkovou abecedou, namísto rozdělení na abecedu terminálů a neterminálů.

Získáme systémy, které v každém jednom kroku aplikují svá pravidla na všechny symboly aktuálního řetězce. Rozdělení systému na komponenty zaručuje větší kontrolu nad výběrem použitelných pravidel. Pokud se zaměříme na jednokrokové módy, můžeme vidět značné podobnosti s ET0L systémy, na které se při zkoumání generativní síly také zaměříme.

3.1 Definice CDE0L GS

Definujeme *kooperující distribuovaný gramatický systém* stupně $n : n \geq 1$ *založený na E0L gramatikách* jako ($n+3$)-tici:

$$\Gamma = (V, T, w, P_1, \ldots, P_n),$$

kde V je celková abeceda,

$T : T \subseteq V$ je abeceda terminálů,

$w : w \in V^+$ je axiom,

$P_i : 1 \leq i \leq n$ je množina pravidel ve tvaru $a \to x : a \in V, x \in V^*$ a nazývá se komponenta systému Γ.

Značíme $G_i = (V, T, P_i, w)$ pro $1 \leq i \leq n$ jako i -tou *E0L gramatiku* systému.

3.2 Derivační módy

Definujeme derivační krok pro i -tou komponentu CDE0L systému jako:

$$a_1 a_2 \ldots a_n \Rightarrow_i x_1 x_2 \ldots x_n,$$

pokud pro všechna $1 \leq j \leq n$ platí $a_j \to x_j \in P_i$. Sekvence derivačních kroků definujeme obdobně jako u jiných gramatik ([2, s. 15]). Dále definujeme módy podle původních systémů vždy pro i -tou gramatiku $G_i = (V, T, P_i, w)$:

1. *ukončující derivaci* jako

   $x \Rightarrow_i^t y$ právě tehdy, když $x \Rightarrow^* y$ v gramatice G_i a $\nexists z (y \Rightarrow z)$ pro $z \in V^*$,

2. k *-krokovou derivaci* jako

$x \Rightarrow_i^{=k} y$ právě tehdy, když $x \Rightarrow^k y$ v gramatice G_i,

3. *nejvíce* k *-krokovou derivaci* jako
   $x \Rightarrow_i^{\leq k} y$ právě tehdy, když $x \Rightarrow^j y$ v gramatice G_i pro všechna $j \leq k$,
4. *nejméně* k *-krokovou derivaci* jako
   $x \Rightarrow_i^{\geq k} y$ právě tehdy, když $x \Rightarrow^j y$ v gramatice G_i pro všechna $j \geq k$.

3.3 Generovaný jazyk

Množinou derivačních módů $D = \{*, t\} \cup \{\leq k, =k, \geq k : k \geq 1\}$ rozumíme množinu symbolů značících tyto derivace a *množinou možných derivací gramatiky* rozumíme $F(G_j, u, f) = \{v : u \Rightarrow_j^f v\}$, kde $1 \leq j \leq n$, $f \in D$, $u \in V^*$.

Jazyky generované gramatickým systémem $\Gamma = (V, T, w, P_1, \ldots, P_n)$ definujeme podle módu takto:

$$L_f(\Gamma) = \{s \in T^* : \exists (v_0, v_1, \ldots, v_m)$$
$$(v_i \in F(G_{j_i}, v_{i-1}, f), 1 \leq i \leq m, 1 \leq j_i \leq n, f \in D, v_0 = w, v_m = s, m \geq 1)\}$$

3.4 Jazykové rodiny

Různé jazykové rodiny budeme značit podle derivačního módu a omezení stupně, tedy počtu gramatik:

$$CDE0L_x(f),$$

kde:

f: $f \in D$ je derivační mód,

x je $n : n \geq 1$ pro stupeň nejvýše n, nebo ∞ pro neomezený stupeň.

$CDE0L_\infty(=)$ značí sjednocení všech jazykových rodin $CDE0L_\infty(=k)$ pro $k \geq 1$.

$CDE0L_\infty(\geq)$ značí sjednocení všech jazykových rodin $CDE0L_\infty(\geq k)$ pro $k \geq 1$.

3.5 Příklady

Ukažme si nyní tři příklady takovýchto systémů a odvoďme jazyky, které generují v jednotlivých módech.

3.5.1 Jazyk $a^i b^i a^i$

První systém definujeme jako $\Gamma_1 = (V, T, S, P_1, P_2, P_3)$, kde

$V = \{S, A, B, a, b\}$,

$T = \{a, b\}$,

$P_1 = \{$
$S \to aAbBaA$,
$B \to bB$,
$A \to aA$,
$a \to a$,
$b \to b$
$\}$,

$P_2 = \{$
$B \to bB$,
$A \to A$,
$a \to a$,
$b \to b$
$\}$,

$P_3 = \{$
$B \to \epsilon$,
$A \to \epsilon$,
$a \to a$,
$b \to b$
$\}$.

Generované jazyky pro různé módy jsou:

$L_f(\Gamma_1)_{f \in \{*, =1, \geq 1\} \cup \{\leq k : k \geq 1\}} = \{a^i b^{i+j} a^i : i \geq 1, j \geq 0\}$,

$L_{=k}(\Gamma_1)_{k>1} = \{a^{ki} b^{k(i+j)} a^{ki} : i \geq 1, j \geq 0\}$,

$L_{\geq k}(\Gamma_1)_{k>1} = \{a^i b^i a^i : i \geq k\} \cup \{a^i b^{i+j} a^i : i \geq k, j \geq k\}$,

$L_t(\Gamma_1) = \emptyset$.

Nejprve prozkoumáme neomezený mód $\Rightarrow^*$. První komponenta přidává v každém kroku jeden znak a zleva a zprava a uprostřed přidává jeden znak b. Tedy derivuje $S \ {}_1\Rightarrow\ aAbBaA \ {}_1\Rightarrow^{i-1}\ a^i Ab^i Ba^i A$. Nyní můžeme přepnout na třetí komponentu, která pouze odstraní neterminály a můžeme přijmout řetězec $a^i b^i a^i$. Jiná možnost je přejít na druhou komponentu, která podobně jako první rozšiřuje prostřední část o jeden znak b. Dojdeme tedy k derivaci $a^i Ab^i Ba^i A \ {}_2\Rightarrow^j a^i Ab^{i+j} Ba^i A$. Po odstranění neterminálů třetí komponentou získáme řetězec $a^i b^{i+j} a^i$. Výsledný jazyk získaný jednokrokovými módy je tedy $\{a^i b^{i+j} a^i : i \geq 1, j \geq 0\}$.

Pro právě k-krokové módy musíme zůstat v dané komponentě právě k kroků. Místo jednoho znaku tak mezi přepnutím komponenty získáme právě k znaků. Výsledný jazyk tedy bude obsahovat pouze k-násobky mocnin znaků, $\{a^{ki} b^{k(i+j)} a^{ki} : i \geq 1, j \geq 0\}$.

S nejméně k-krokovými módy nemůžeme přepnout komponentu před dosažením k kroků. Zajistíme tak zvýšení omezení pro i a j. Musíme však rozlišit, jestli použijeme druhou

komponentu. Bez jejího využití získáme stejné mocniny znaků a a b, zatímco pokud druhou komponentu alespoň jednou využijeme, mocniny b budou alespoň o k vyšší. Získáme tedy jazyk $\{a^i b^i a^i : i \geq k\} \cup \{a^i b^{i+j} a^i : i \geq k,\ j \geq k\}$.

V ukončovacím módu se první komponenta nikdy nezastaví, takže výsledný jazyk je prázdný.

3.5.2 Jazyk $(a^i b)^j c^j$

Druhý systém definujeme jako $\Gamma_2 = (V, T, S, P_1, P_2, P_3)$, kde

$V = \{S, R, Q, A, B, a, b, c\}$,

$T = \{a, b, c\}$,

$P_1 = \{$
$S \to BSc,$
$S \to BRc,$
$B \to B,$
$c \to c$
$\}$,

$P_2 = \{$
$R \to R,$
$R \to Q,$
$B \to Aab,$
$A \to Aa,$
$a \to a,$
$b \to b,$
$c \to c$
$\}$,

$P_3 = \{$
$Q \to \epsilon,$
$A \to \epsilon,$
$a \to a,$
$b \to b,$
$c \to c$
$\}$.

Generované jazyky jsou:

$L_f(\Gamma_2)_{f \in \{*, =1, \geq 1\} \cup \{\leq k : k \geq 1\}} = L_t(\Gamma_2) = \{(a^i b)^j c^j : i, j \geq 1\}$,

$L_{=k}(\Gamma_2)_{k>1} = \{(a^{ki} b)^{kj} c^{kj} : i, j \geq 1\}$,

$L_{\geq k}(\Gamma_2)_{k>1} = \{(a^i b)^j c^j : i, j \geq k\}$.

Opět nejprve rozebereme neomezený mód. První komponenta generuje znaky B a c a v posledním kroku nutně musí přepsat S na R. Před prvním přepnutím tedy máme řetězec $B^j R c^j$. Znak R vynucuje použití druhé komponenty, která zachovává znaky c, přepisuje B na $Aa^i b$ a nakonec přepíše R na Q. Před druhým přepnutím tedy máme řetězec $(Aa^i b)^j Q c^j$. Poslední komponenta podobně jako v minulém příkladě pouze odstraňuje neterminály, aby mohl být řetězec přijat. Získáme tedy jazyk $\{(a^i b)^j c^j : i, j \geq 1\}$.

Právě k-krokové módy opět pouze vyřadí některé mocniny znaků. Přepnutí na jinou komponentu je možné pouze po k krocích, pokud poslední krok přepsal S na R, resp. R Na Q. Získáme tedy podmnožinu předchozího jazyka – $\{(a^{ki} b)^{kj} c^{kj} : i, j \geq 1\}$.

V nejméně k-krokových módech máme určenou dolní hranici počtu kroků, po kterých můžeme přepnout komponentu, takže se opět jen zvednou minimální mocniny znaků v řetězci. Generujeme tedy jazyk $\{(a^i b)^j c^j : i, j \geq k\}$.

Díky nutnosti projít sekvenci S, R a Q, je možné zastavit ukončovací mód vždy až po takovém přepisu. Do té doby je možné prodlužovat řetězce podle potřeby, takže tento mód generuje stejný jazyk jako ty jednokrokové.

3.5.3 Jazyk $a^{2^i}b^j$

Třetí systém definujeme jako $\Gamma_3=(V, T, S, P_1, P_2, P_3, P_4)$, kde
$V=\{S, A, B, \overline{B}, C, a, b, c, \overline{c}\}$,
$T=\{a, b\}$,

$P_1=\{$
$S\to AB,$
$A\to CC,$
$C\to A,$
$C\to ac$
$\},$

$P_2=\{$
$B\to b\overline{B},$
$\overline{B}\to\overline{B},$
$a\to a,$
$b\to b,$
$c\to c,$
$c\to\overline{c}$
$\},$

$P_3=\{$
$\overline{B}\to B,$
$a\to a,$
$b\to b,$
$c\to c,$
$\overline{c}\to\epsilon$
$\},$

$P_4=\{$
$B\to\epsilon,$
$a\to a,$
$b\to b$
$\}.$

Generované jazyky jsou:
$L_f(\Gamma_3)_{f\in\{*,=1,\geq 1\}\cup\{\leq k:k\geq 1\}}=\{a^{2^i}b^j : i, j\geq 1\}$,
$L_{=k}(\Gamma_3)=L_{\geq k}(\Gamma_3)=\emptyset$,
$L_t(\Gamma_3)=\{a^{2^i}b^j : i\geq 1, 2^i\geq j\geq 1\}$.

Tento systém je zajímavý především svým ukončovacím módem. Nejprve ale neomezený mód. V něm se nejprve vygeneruje řetězec $(ac)^{2^i}B$, potom se použitím druhé komponenty připíše jeden znak b a zároveň se libovolný počet c přepíše na $\overline{c}$. Ve třetí komponentě se „odblokuje“ symbol $\overline{B}$, aby bylo možné připsat další znak b ve druhé komponentě. Také se smažou všechny znaky $\overline{c}$. Takto se v tomto módu nakonec smažou všechna c a připíše se libovolný počet b. Po použití poslední komponenty pak dostaneme řetězec terminálů $a^{2^i}b^j$, který můžeme přijmout. Generovaný jazyk je pak $\{a^{2^i}b^j : i, j\geq 1\}$.

Vícekrokové módy v tomto systému kvůli třetí komponentě selhávají. V ní se totiž přepisuje $\overline{B}$ na B, pro které však nejsou obsažena žádná pravidla, takže se systém zastaví předčasně. Důsledkem toho, že nelze využít třetí komponentu, nemůžeme smazat neterminály $\overline{c}$, takže nemůžeme získat žádný řetězec, jenž by se dal přijmout.

Nyní tedy ukončovací mód. Tady fungují symboly c jako určitá forma počítadla, protože při každém využití druhé komponenty se alespoň jedno c musí přepsat na $\overline{c}$, jinak se ukončovací mód nezastaví. Všechna c se smažou nejméně za jedno použití a nejvíce za počet c, tedy 2^i, použití druhé komponenty. Každé toto využití vygeneruje jedno b, čímž získáme nakonec řetězec se shora omezeným počtem b. Výsledný jazyk je $\{a^{2^i}b^j : i\geq 1, 2^i\geq j\geq 1\}$.

4 Generativní síly jednotlivých módů

4.1 Neomezený derivační mód

Každý CDE0L gramatický systém lze v módu $*$ převést na ekvivalentní gramatický systém, který obsahuje pouze dvě komponenty. Mějme původní systém $\Delta = (W, T, w, R_1, ..., R_n)$. Ekvivalentní dvoukomponentní systém pak definujeme jako:

$$\Gamma = (V, T, w, P_1, P_2),$$

kde:

$V = W \cup \{\langle a, i\rangle : a \in W, 1 \le i \le n\}$ je rozšířená abeceda,

$P_1 = \{a \to \langle a, 1\rangle : a \in W\} \cup$
$\{\langle a, i\rangle \to \langle a, i+1\rangle : a \in W, 1 \le i < n\}$ je množina pravidel pro výběr simulované gramatiky a

$P_2 = \{\langle a, i\rangle \to x : a \in W, 1 \le i \le n, a \to x \in R_i\}$ je množina pravidel pro simulaci vybrané gramatiky.

Mějme řetězec původního systému $x = a_1 a_2 ... a_m$, kde $a_i \in W$. Druhá komponenta neobsahuje pravidla pro nezměněné symboly, tudíž musíme použít první komponentu. Po k krocích v první komponentě nutně získáme řetězec $y = \langle a_1, k\rangle\langle a_2, k\rangle \cdots \langle a_m, k\rangle$. Takto jsme označili, kterou gramatiku z původního systému budeme simulovat ve druhé komponentě.

4.1.1 ET0L systémy

Pro dokázání ekvivalence mezi ET0L systémy a CDE0L systémy využijeme možnosti převodu obou systémů na jejich dvoutabulkové resp. dvoukomponentní ekvivalenty. Mějme ET0L systém $H = (W, T, R_1, R_2, ... R_n, w)$ zavedený podle [3, s. 21]. Z něho vytvoříme dvoutabulkový systém $G = (V, T, P_1, P_2, w)$, kde:

$V = W \cup \{\langle a, i\rangle : a \in W, 1 \le i \le n\}$ je rozšířená abeceda,

$P_1 = \{a \to \langle a, 1\rangle : a \in W\} \cup$
$\{\langle a, i\rangle \to \langle a, i+1\rangle : a \in W, 1 \le i < n\}$ je množina pravidel pro výběr tabulky a

$P_2 = \{\langle a, i\rangle \to x : a \in W, 1 \le i \le n, a \to x \in R_i\}$ je množina pravidel pro simulaci vybrané tabulky.

Z tohoto systému vytvoříme dvoukomponentní CDE0L systém se stejnými množinami $\Gamma = (V, T, w, P_1, P_2)$.

ET0L systémy uplatňují derivační pravidla z jedné tabulky na všechny symboly řetězce právě jednou za derivační krok a tabulku vybírají nedeterministicky. CDE0L gramatické

systémy také vybírají následnou komponentu nedeterministicky a v $*$ módu uplatňují libovolný počet kroků z vybrané komponenty, tedy i právě jeden krok. V rámci jednoho kroku také uplatňují derivační pravidla právě jednou na všechny symboly v daném řetězci.

Poněvadž mají ET0L systém G a CDE0L systém Γ stejné prvky i stejný postup derivací, dokázali jsme, že jazykové rodiny $\mathscr{L}(ET0L)$ a $CDE0L_{\infty}(*)$ jsou ekvivalentní.

4.2 Nejvíce *k*-krokové derivační módy

Tyto módy, stejně jako mód $*$, obsahují vždy možnost derivace s jedním krokem. Derivace s větším počtem kroků pouze zabraňují přepnutí komponenty, tudíž derivace s jedním krokem je schopna vygenerovat všechny možné řetězce a další derivace generují jen podmnožinu z tohoto jazyka. Proto módy, které obsahují jednokrokovou derivaci – tedy módy $*$, $=1$, ≥ 1 a $\leq k$ pro $k \geq 1$ – jsou navzájem ekvivalentní.

4.3 Nejméně a právě *k*-krokové derivační módy

Pro každý derivační mód se zadaným nejkratším krokem větším než jedna lze sestrojit ekvivalentní dvoukomponentní systém. Nechť $\Delta = (W, T, w, R_1, R_2, ..., R_n)$ je původní systém, pak:

$$\Gamma = (V, T, w, P_1, P_2),$$

je ekvivalentní dvoukomponentní systém ve stejném módu, kde:

$$\begin{aligned}
V = {} & W \cup \{\langle a, i, j\rangle : a \in W, 1 \leq i \leq n, 0 \leq j < k\} \cup \\
& \{\langle a, i\rangle : a \in W, 1 \leq i \leq n\}, \\
P_1 = {} & \{a \to \langle a, 1, k-1\rangle : a \in W\} \cup \\
& \{\langle a, i, j\rangle \to \langle a, i, j-1\rangle : a \in W, 1 \leq i \leq n, 1 \leq j < k\} \cup \\
& \{\langle a, i, 0\rangle \to \langle a, i+1, k-1\rangle : a \in W, 1 \leq i < n\}, \\
P_2 = {} & \{\langle a, i, 0\rangle \to \langle x_1, i\rangle\langle x_2, i\rangle ... \langle x_d, i\rangle : a \in W, 1 \leq i \leq n, a \to x_1 x_2 ... x_d \in R_i\} \cup \\
& \{\langle a, i\rangle \to \langle x_1, i\rangle\langle x_2, i\rangle ... \langle x_d, i\rangle : a \in W, 1 \leq i \leq n, a \to x_1 x 2 ... x_d \in R_i\} \cup \\
& \{\langle a, i\rangle \to x : a \in W, 1 \leq i \leq n, a \to x \in R_i\}.
\end{aligned}$$

První komponenta zajišťuje vybrání simulované komponenty z původního systému s vložením čekacích symbolů pro splnění nejkratšího kroku. První část pravidel zakóduje původní symbol a nastaví zpoždění. Druhá část pravidel snižuje zpoždění do vypršení, po kterém je možné přepnout komponentu. Jiné symboly než s vypršeným zpožděním nemají v druhé komponentě pravidla. Třetí část pravidel zajišťuje vybrání další původní komponenty a zároveň znovunastavení zpoždění.

Druhá komponenta obsahuje simulační smyčku ve druhé části pravidel, jež umožňuje použít pravidla z vybrané komponenty bez dalších omezení. Před ukončením simulace je použito pravidlo ze třetí části, které zajistí původní symboly. První část simuluje jeden krok a zároveň maže počítadlo zpoždění. Toto je důležité pro jedinečnost symbolů. Pokud by se použila pravidla z druhé a třetí části v jednom kroku, obsahoval by řetězec původní symboly i symboly s výběrem komponenty. Protože v první gramatice neexistují pravidla pro tyto jedinečné symboly a ve druhé gramatice neexistují pravidla pro původní symboly, nebylo by možné pokračovat ani přijmout řetězec.

4.3.1 Ekvivalence s jednokrokovými módy

Vícekrokové derivační módy umožňují simulovat jednokrokové módy a to vložením čekacích symbolů do druhé komponenty. Změníme tedy druhou komponentu na:

$$P_2=\{\langle a,i,j\rangle\rightarrow\langle a,i,j+1\rangle : a\in W, 1\le i\le n, 0\le j<k-1\} \cup$$
$$\{\langle a,i,k-1\rangle\rightarrow x : a\in W, 1\le i\le n, a\rightarrow x\in R_i\},$$

takto se po k krocích odsimuluje právě jeden krok v původním systému.

Jednokrokové módy dovolují simulovat vícekrokové módy, pokud budeme počítat čísla kroků v rámci symbolů. Definujeme tedy systém $\Gamma=(V,T,w,P_1,P_2)$, který v jednokrokovém módu simuluje původní systém $\Delta=(W,T,w,R_1,R_2,...,R_n)$:

$$V=W \cup \{\langle a,i,j\rangle : a\in W, 1\le i\le n, 0\le j<k\} \cup$$
$$\{\langle a,i\rangle : a\in W, 1\le i\le n\},$$
$$P_1=\{a\rightarrow\langle a,1,0\rangle : a\in W\} \cup$$
$$\{\langle a,i,0\rangle\rightarrow\langle a,i+1,0\rangle : a\in W, 1\le i<n\},$$
$$P_2=\{\langle a,i,j\rangle\rightarrow\langle x_1,i,j+1\rangle\ldots\langle x_d,i,j+1\rangle : a\in W, 1\le i\le n, 0\le j<k-1, a\rightarrow x_1\ldots x_d\in R_i\} \cup$$
$$\{\langle a,i,k-1\rangle\rightarrow x : a\in W, 1\le i\le n, a\rightarrow x\in R_i\} \cup$$
$$\{\langle a,i,k-1\rangle\rightarrow\langle x_1,i\rangle\langle x_2,i\rangle\ldots\langle x_d,i\rangle : a\in W, 1\le i\le n, a\rightarrow x_1x_2\ldots x_d\in R_i\} \cup$$
$$\{\langle a,i\rangle\rightarrow\langle x_1,i\rangle\langle x_2,i\rangle\ldots\langle x_d,i\rangle : a\in W, 1\le i\le n, a\rightarrow x_1x_2\ldots x_d\in R_i\} \cup$$
$$\{\langle a,i\rangle\rightarrow x : a\in W, 1\le i\le n, a\rightarrow x\in R_i\}.$$

První komponenta pracuje pouze s původními symboly a symboly s nulovým počítadlem. Proto pokud se využije druhá komponenta, musí se používat až do doby, než vygeneruje řetězec z původních symbolů.

Druhá komponenta se skládá z velkého množství pravidel. První část tvoří simulace s počítáním, kolik kroků už proběhlo. Po $k-1$ krocích se použije druhá část pravidel, která vygeneruje přijímatelný původní řetězec, nebo třetí část pravidel, která zajistí přepis na unikátní formu, ve které čtvrtá část pravidel může bez horního omezení počtu kroků dál simulovat. Pátá část pravidel pak už slouží k ukončení simulace a vytvoření řetězce původních symbolů.

Opět pokud se v řetězci objeví původní symboly a některé jiné, nelze využít ani jednu komponentu.

Z tohoto tedy vyplývá, že rodiny jazyků $CDE0L_{\infty}(\geq)$ a $CDE0L_{\infty}(=)$ jsou stejné jako rodina $CDE0L_{\infty}(*)$ a tedy i $\mathscr{L}(ET0L)$.

4.4 Ukončovací derivační mód

Ukončovací derivační mód CDE0L systému s neomezeným počtem komponent lze převést na ekvivalentní tříkomponentní systém. Protože v tomto módu lze přepnout komponentu pouze, když už nelze aplikovat další pravidla, je nutné rozdělit výběrovou komponentu dvoukomponentního systému z výše uvedeného algoritmu na dvě samostatné komponenty, které zajistí, že se systém vždy po jednom derivačním kroku zastaví. Toho lze dosáhnout například takto:

Nechť $\Delta = (W, T, w, R_1, R_2, ..., R_n)$ je původní systém, pak ekvivalentní tříkomponentní systém v ukončovacím módu definujeme jako:

$$\Gamma = (V, T, w, P_1, P_2, P_3),$$

kde:

$$\begin{aligned}
V = {} & W \cup \{\overline{\langle a \rangle} : a \in W\} \cup \\
& \{\langle a, i \rangle : a \in W, 1 \leq i \leq n\} \cup \\
& \{\overline{\langle a, i \rangle} : a \in W, 1 \leq i \leq n\}, \\
P_1 = {} & \{a \to \overline{\langle a, 1 \rangle} : a \in W\} \cup \\
& \{\langle a, i \rangle \to \overline{\langle a, i+1 \rangle} : a \in W, 1 \leq i < n\}, \\
P_2 = {} & \{\overline{\langle a, i \rangle} \to \langle a, i \rangle : a \in W, 1 \leq i \leq n\} \cup \\
& \{\overline{\langle a, i \rangle} \to \overline{\langle a \rangle} : a \in W, 1 \leq i \leq n\}, \\
P_3 = {} & \{\overline{\langle a, i \rangle} \to \overline{\langle x_1, i \rangle}\,\overline{\langle x_2, i \rangle} \ldots \overline{\langle x_d, i \rangle} : a \in W, 1 \leq i \leq n, a \to x_1 x_2 \ldots x_d \in R_i\} \cup \\
& \{\overline{\langle a \rangle} \to a : a \in W\}.
\end{aligned}$$

Mějme řetězec původního systému $x = a_1 a_2 \ldots a_m$, kde $a_i \in W$. Jediná použitelná komponenta je první, která se zastaví po jednom kroku na řetězci se symboly s pruhy a jedničkou. Využitím první poloviny pravidel druhé komponenty můžeme smazat pruh a znovu použít první komponentu pro získání symbolů s pruhem a dalším číslem. Pro simulaci samotné gramatiky použijeme třetí komponentu. Ta se po odsimulování původní gramatiky zastaví na řetězci $y = \overline{\langle a_1, k \rangle}\,\overline{\langle a_2, k \rangle} \cdots \overline{\langle a_m, k \rangle}$. Dále se použije druhá komponenta pro odstranění číslování a třetí komponenta pro odstranění pruhu.

Druhá komponenta odstraňuje buď pouze pruh, nebo pouze číslo. Pokud by se odstranilo částečně jedno i druhé, nedala by se použít první komponenta, protože nemá pravidla pro pruhové symboly, ani třetí komponenta, která nemá pravidla pro bezpruhové symboly.

4.4.1 Generativní síla ukončovacího derivačního módu

Ukončovací mód lze využít pro simulaci jednokrokových módů a tedy i ET0L systémů:

Mějme původní systém $\Delta = (W, T, w, R_1, R_2, ..., R_n)$ v jednokrokovém módu, pak definujeme ekvivalentní systém v ukončovacím módu $\Gamma = (V, T, w, P_1, P_2, P_3)$:

$V = W \cup \{\langle a, i\rangle : a \in W, 1 \le i \le n\} \cup$
$\{\overline{\langle a, i\rangle} : a \in W, 1 \le i \le n\},$
$P_1 = \{a \to \overline{\langle a, 1\rangle} : a \in W\} \cup$
$\{\langle a, i\rangle \to \overline{\langle a, i+1\rangle} : a \in W, 1 \le i < n\},$
$P_2 = \{\overline{\langle a, i\rangle} \to \langle a, i\rangle : a \in W, 1 \le i \le n\},$
$P_3 = \{\overline{\langle a, i\rangle} \to x : a \in W, 1 \le i \le n, a \to x \in R_i\}.$

První komponenta opět postupně vybírá původní komponentu a blokuje symboly. Druhá komponenta odblokuje symboly pro další použití první komponenty. Třetí komponenta simuluje právě jeden krok podle původního systému.

Nyní si vezměme jazyk, který nelze generovat ET0L systémem, a vytvořme systém, který tento jazyk generuje. Příkladem takového jazyka je $\{(ab^n)^m : m \ge n \ge 1\}$, který je podle [3, s. 23] kontextový jazyk, který nelze generovat ET0L systémem.

Mějme systém $\Gamma = (V, T, S, P_1, P_2, P_3, P_4)$:

$V = \{S, F, A, B, \overline{B}, a, b, c, \overline{c}\},$
$T = \{a, b\},$

$P_1 = \{$
$S \to F,$
$S \to ABSc,$
$c \to c,$
$B \to B,$
$A \to A$
$\},$

$P_2 = \{$
$F \to \epsilon,$
$B \to b\overline{B},$
$c \to c,$
$\overline{c} \to \epsilon,$
$b \to b,$
$A \to A$
$\},$

$P_3 = \{$
$c \to c,$
$c \to \overline{c},$
$\overline{B} \to B,$
$B \to B,$
$b \to b,$
$A \to A$
$\},$

$P_4 = \{$
$\overline{c} \to \epsilon,$
$B \to \epsilon,$
$b \to b,$
$A \to a$
$\}.$

První komponenta generuje ze symbolu S řetězec $(AB)^m Sc^m$ a poté $(AB)^m Fc^m$. Jediná další použitelná komponenta je druhá, která generuje $(Ab\overline{B})^m c^m$. Další komponenta je třetí, která generuje $(AbB)^m c^{m-i} \overline{c}^i$, kde $1 \le i \le m$. Protože pro $i = 0$ se komponenta nezastaví, musí být v ukončovacím módu provedena alespoň jedna změna. Tato metoda je umožněna právě až CDE0L systémy, protože ukončovací mód se zde zastaví, už když nelze použít pravidlo pro alespoň jeden symbol, na rozdíl od CD systémů, kde se zastaví až když nelze použít pravidlo na žádný symbol. Následně se opakuje použití druhé a třetí komponenty dokud se nevyčerpají symboly c, tedy nanejvýš m-krát. Získáme řetězec $(Ab^n B)^m \overline{c}^i$, kde

$1 \leq n \leq m$. Poslední krok využije čtvrtou komponentu a odstraní všechny neterminály. Získáme $(ab^n)^m$, kde $m \geq n \geq 1$, tedy žádaný jazyk.

Tímto jsme dokázali, že generativní síla CDE0L systémů v ukončovacím módu je větší než síla ET0L systémů, tedy $\mathscr{L}(ET0L) \subset CDE0L_3(t) = CDE0L_\infty(t)$.

4.5 Shrnutí generativní síly

Pokud shrneme dosažené závěry do vět podobně jako v kapitole 2.2, získáme:

1. $\mathscr{L}(E0L) = CDE0L_1(f) \subset CDE0L_2(f) = CDE0L_\infty(f) = \mathscr{L}(ET0L)$, pro $f \in \{*\} \cup \{=k, \geq k, \leq k : k \geq 1\}$ a
2. $\mathscr{L}(E0L) \subseteq CDE0L_1(t) \subseteq CDE0L_2(t) \subseteq CDE0L_3(t) = CDE0L_\infty(t) \supset \mathscr{L}(ET0L)$.

Jak je vidět, většina hierarchie kolabuje do ekvivalence buď s E0L systémy, nebo ET0L systémy. Nejzajímavější výsledek je síla ukončovacího módu, protože převyšuje sílu komponentních E0L systémů i ukončovacího módu CD systémů.

5 Demonstrační skript

Pro názornou demonstraci CDE0L systémů a použitých algoritmů jsem vytvořil skript v jazyce Python 3.3, který implementuje převody na dvou a tříkomponentní systémy. Dále simulace jednokrokových módů vícekrokovými a naopak a simulace jednokrokových módů ukončujícím módem. Navíc také implementuje hledání příkladů řetězců jazyka takového systému v daném módu.

5.1 Textový formát systému

Pro vstup a výstup je dána pevná forma. Mezi kterýmikoli prvky může být libovolný počet bílých znaků. Výstup však dodržuje klasické typografické zásady použití mezer za čárkami a u závorek. Je použito odsazení dvou mezer pro jednu úroveň zanoření.

Celý systém je uzavřen v kulatých závorkách. Axiom a jednotlivé množiny – abecedy a pravidla – jsou od sebe odděleny čárkou, za poslední množinou pravidel však čárka být nesmí. Prvky množin jsou uzavřeny ve složených závorkách a odděleny čárkou, která může být i za posledním prvkem.

Symboly abecedy stejně jako axiom musí mít pouze jedno písmeno nebo řetězec ohraničený lomenými závorkami (symboly menší `<` a větší `>`). Případně je možné přidat apostrof `'`, který nahrazuje pruh v použitých algoritmech. Znaky v lomených závorkách jsou využívány pouze jako výstup z algoritmů a číst je lze jen v režimu vypisování příkladů.

Pravidla jsou ve tvaru $a->x_1x_2...x_n$. Mezi členy pravé strany pravidla se nesmí vyskytovat žádný bílý znak. V případě prázdné pravé strany ji lze vynechat nebo použít znak ϵ (U+03F5).

Ve výstupním formátování je každá položka systému na jednom řádku, kromě komponent, pro které platí, že jeden řádek odpovídá jednomu pravidlu.

5.2 Použití skriptu

Skript pracuje se standardním vstupem a výstupem, což lze změnit pomocí parametrů `--input` a `--output`. Ze vstupu vždy načte zadaný systém a zpracuje ho. Na výstup posílá podle módu přečtený systém, upravený systém nebo příklady řetězců. Jednotlivé módy se volí pomocí parametrů, jak lze vidět v tabulce 1.

Dlouhý parametr	Krátký parametr	Význam
`--print`	`-p`	vytisknutí přečteného systému
`--examples`	`-e`	hledání příkladů
`--reduce`	`-r`	vytvoření dvou nebo tříkomponentního systému
`--convert`	`-c`	vytvoření systému simulující jiný mód

Tabulka 1: Parametry pro módy skriptu

5.2.1 Režim hledání příkladů

Pro hledání příkladů je nutné zadat derivační mód, který chceme zkoušet. To se provádí parametrem `--mode` po kterém následuje označení `*`, `t`, `<=k`, `=k` nebo `>=k`, kde `k` je konkrétní číslo.

V tomto módu se postupně hledají řetězce, které lze přijmout zadaným systémem. Jelikož není zaručeno, že vůbec takový řetězec existuje, je k dispozici parametr `-t`, za kterým následuje kolik sekund má skript běžet od posledního úspěšného nalezení řetězce. Dále lze nastavit, kolik příkladů se má hledat, a to pomocí parametru `-n`.

Množství vypisovaných informací o řetězci určuje přepínač `--full`. Bez této volby se vypisují jen přijaté řetězce, každý na jeden řádek. Naopak při zapnutí se na každý řádek vypisuje série derivací, které vedly až k tomuto řetězci.

5.2.2 Režim redukce

Pro jedno i vícekrokové módy vytváří režim redukce dvoukomponentní systém, který generuje stejný jazyk. Pro ukončovací mód je generován tříkomponentní systém. Jediným povinným parametrem je `--mode`, kterým se nastavuje, ze kterého módu se má tvořit nový systém.

5.2.3 Režim konverze

Skript podporuje tři druhy konverzí, tedy vytvoření nového systému v zadaném módu, který generuje stejný jazyk jako zadaný systém v jiném módu. Lze konvertovat systémy v jednokrokovém módu na systémy v ukončovacím nebo vícekrokovém módu nebo vícekrokové na jednokrokové. Jiné konverze, třebaže teoreticky možné, nejsou povoleny.

Pro zadání módu vstupního systému, tedy simulovaného, se využívá parametr `-in` a pro mód výstupního, tedy vygenerovaného, systému slouží parametr `-out`.

5.3 Principy skriptu

Nejdůležitější částí skriptu je soubor CDE0L_class.py, ve kterém je definována třída, která implementuje požadované metody, tedy načítání a ukládání z a do řetězce, konverze, redukce a hledání řetězců. Symboly jsou vnitřně reprezentovány jako řetězce. Abecedy jsou množinami řetězců a pravidla dvojicemi symbolu a n-tice symbolů.

Druhý soubor CDE0L_demo.py pouze zpracovává argumenty a komunikuje s objektem třídy CDE0L.

5.3.1 Načtení systému z řetězce

Pro načtení systému z řetězce je použito pouze regulárních výrazů. V první fázi se nahradí všechny bílé znaky právě jednou mezerou. Ve druhé fázi se řetězec rozdělí na jednotlivé komponenty, tedy množiny a axiom. V těchto komponentách se dále rozpoznávají oddělovače a symboly. Na závěr se kontroluje, zda axiom a všechny symboly v abecedě terminálů a v pravidlech byly zadány v celkové abecedě.

5.3.2 Hledání řetězce jazyka daného systému

Hledání řetězce využívá algoritmus hledání do šířky, avšak místo normální fronty, do které se ukládají rozgenerované stavy, zavádí prioritní frontu, ve které je zavedeno řazení podle délky aktuálního řetězce kombinované s počtem již proběhlých derivací. Tímto jsou upřednostňovány kratší řetězce a pokud systém neobsahuje epsilon pravidla, algoritmus nalézá řetězce seřazeně podle délky.

V každém kroku se podle omezení buď vybere nová komponenta, nebo zůstane stávající a vygenerují se všechny možné kombinace použitelných pravidel na všechny symboly. Poté se každá kombinace testuje, jestli už v dané sekvenci derivací nebyla přítomna, čímž se vyloučí sekvence se smyčkou.

Po vytažení z fronty se řetězec testuje a pokud splňuje podmínky, je vrácen z funkce. Takový řetězec musí být pouze z terminálů, dále musí být v kroku, kdy je možné přepnout komponentu, a také musí být unikátní vzhledem k již vráceným řetězcům.

(1)	Nastavení počítadla kroků podle módu
(2)	Dokud nebylo posláno N příkladů:
(3)	.....Zkontrolování času od posledního nalezení řetězce
(4)	.....Vyzvednutí řetězce ze počátku fronty
(5)	.....Pokud je řetězec z terminálů, je unikátní a je správný krok:
(6)	.....Odeslání řetězce
(7)	.....Přičtení počítadla kroku
(8)	.....Omezení komponent k použití podle kroku
(9)	.....Pro každou komponentu:
(10)	.....Vygenerování všech možných kombinací uplatnění pravidel z komponenty na daný řetězec
(11)	.....Zapsání vygenerovaných řetězců do fronty s prioritou délka(historie) + délka(řetězce)

Algoritmus 1: Hledání řetězce

5.3.3 Aplikace algoritmů

V jazyce Python je zápis matematických konstrukcí do značné míry přímočarý. Jednotlivé algoritmy tak pouze vytváří odpovídající rozšířené abecedy a nové množiny pravidel, z nichž se pak sestaví a vrátí nový systém. Pro symboly s pruhem je využit apostrof za daným symbolem a složené symboly z algoritmů jsou zapsány v lomených závorkách.

6 Závěr

V této práci jsem definoval kombinaci E0L a CD gramatických systémů. Ukázal jsem jejich vlastnosti na příkladech a dále se zabýval generativní silou hlavních módů používaných v literatuře, tedy neomezeným a ukončovacím módem a dále nejméně, právě a nejvíce k-krokovými módy.

Ukázal jsem algoritmy pro redukci počtu gramatik na dvě, resp. na tři pro ukončovací mód pomocí číslování pravidel podle komponenty a postupný výběr simulované gramatiky. Další oblastí byly algoritmy pro simulaci vícekrokových módů jednokrokovými a naopak pomocí počítání kroků. Také jsem uvedl důkaz jejich ekvivalence s ET0L systémy. Na závěr jsem se věnoval ukončovacímu módu a jeho simulování jednokrokových módů. Hlavním závěrem je pak důkaz, že ukončovací mód má větší generativní sílu než ET0L systémy a tedy i než ostatní módy. Toho jsem docílil prezentováním systému, který generuje jazyk $(ab^n)^m$, kde $m \geq n \geq 1$. Tento jazyk podle literatury nelze generovat ET0L systémem.

Dané algoritmy jsou pak implementované v rámci demonstračního skriptu v jazyce Python. Skript čte zadaný systém v textové formě a vypisuje systémy po aplikaci vybraných algoritmů. Také je možné pokusit se vypsat příklady řetězců z takového systému.

Pro další práci je otevřena otázka síly ukončovacího módu z hlediska kontextových gramatik. Tato práce dokázala, že přesahuje ET0L systémy, ale v rámci hledání tohoto důkazu jsem zkoumal např. kontextový jazyk a^{2^c}, který možná nelze generovat ani v ukončovacím módu. Hlavním důvodem pro tuto hypotézu je způsob generování jazyka $(ab^n)^m$ v kapitole 4.4.1. Ukončovací mód vyžaduje přepsání alespoň jednoho c využívaného jako počítadlo. Nedokáže však vynutit právě jednu změnu, takže symboly c nemohou být použité jako počítadlo jedna k jedné, ale pouze jako horní omezení. Je však možné, že existuje jiná struktura komponent, která by toto umožňovala.

Další otevřenou otázkou je síla jedno a dvoukomponentních systémů v ukončovacím módu. Dá se předpokládat, že jednokomponentní systém je stejně silný jako E0L systém, ze kterého je tvořen, otázkou však je kam zařadit dvoukomponentní systém. Podle CD systémů se dá předpokládat, že by měl být stejně silný jako jednokomponentní.

Jinou možností je pak zkoumání jiných módů, např. kombinací stávajících jako je definováno v [6]. Nebo zavést úplně nový mód, který by využíval vlastností E0L gramatik, ze kterých je systém složen.

Asi poslední, ale nikoli nejméně zajímavou možností je zavést podobně jako u CD systémů jejich hybridní verze. Ty se liší rozdílným přístupem k módům. Zatímco u normálních systémů je jeden mód určen pro generování celého jazyka, hybridní verze vážou mód na komponentu.

Literatura

[1] DASSOW, Jürgen, Gheorghe PĂUN a Grzegorz ROZENBERG. Grammar Systems. **In:** ROZENBERG, Grzegorz a Arto SALOMAA, eds. *Handbook of Formal Languages.* Berlin: Springer, 1997, sv. 2, s. 155-214, ISBN 3-540-60648-3.

[2] MEDUNA, Alexander a Petr ZEMEK. *Regulated Grammars and Their Transformations*. Brno, 2010, ISBN 978-80-214-4203-0.

[3] MASOPUST, Tomáš, Alexander MEDUNA a Jiří TECHET. *Lindenmayer Systems*[online]. Brno, 2007[cit. 2014-02-12]. Dostupné z: http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:08-lsystemspres.pdf.

[4] MASOPUST, Tomáš, Alexander MEDUNA a Jiří TECHET. *Cooperating Distributed Grammar Systems*[online]. Brno, 2007[cit. 2014-02-12]. Dostupné z: http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:09-cdgspres.pdf.

[5] BORDIHN, Henning, Erzsébet CSUHAJ-VARJÚ a Jürgen DASSOW. CD Grammar Systems Versus L Systems. **In:** PĂUN Gheorghe, Arto SALOMAA, eds. *Grammatical Models of Multi Agent Systems.* Amsterdam: Gordon & Breach, 1999, s. 18-32, ISBN 90-5699-177-9.

[6] FERNAU, H., M. Holzer a R. Freund. Hybrid modes in cooperating distributed grammar systems: internal versus external hybridization. *Theoretical Computer Science.* Elsevier, květen 2001, **259**(1-2), 405-426.

[7] MAURER, H. A., A. SALOMAA, D. WOOD. ET0L forms. *Journal of Computer and System Sciences.* Elsevier, červen 1978, **16**(3), 345-361.

[8] EHRENFEUCHT, A. a G. ROZENBERG. On the Structure of derivation in Deterministic ET0L Systems. *Journal of Computer and System Sciences.* Elsevier, prosinec 1978, **17**(3), 331-347.

[9] MEDUNA, Alexander. *Automata and Languages: Theory and Applications.* London: Springer, 2000, ISBN 1-85233-074-0.

Obsah CD

Přiložené CD obsahuje:

- písemnou zprávu ve formátu PDF,
- zdrojový tvar písemné zprávy ve formátu ODT,
- zdrojové texty demonstračního skriptu a manuál,
- ukázkové vstupní soubory.