Gramatické systémy - teorie a aplikace

Luděk Dolíhal

January 8, 2010

Contents

1 Úvod

Cílem této práce je diskutovat problematiku gramatických systémů ve vztahu k moderním programovacím metodám a jazykům. Bude předestřeno teoretické pozadí gramatických systémů a některých příbuzných modelů. Detailně budou probrány dva základní typy gramatických systémů a to CD gramatické systémy a PC gramatické systémy. Pokusím se popsat aplikaci těchto systémů v praxi a najít příklady jejich reálného využití.

1.1 Teorie gramatických systémů

V běžné teorii programovacích jazyků je výpočet prováděn jednou výpočetní jednotkou. To znamená, že jazyk je generován jednou gramatikou a přijímán jedním automatem. Nicméně v dnešní době hraje stále větší roli distribuované zpracování. Takovéto systémy si však nevystačí se základní teorií programovacích jazyků. Jsou potřeba prostředky pro zápis paralelismu, distribuování výpočtu, konkurence a jiných. Proto byly zavedeny gramatické systémy.

Gramatické systémy ve své podstatě představují množinu gramatik pracujících pohromadě. Pokud běží výpočet na více stanicích, jsou stanice synchronizovány pomocí protokolu tak, aby produkovaly jeden jazyk. Důvodů, proč se překlad děje na více místech, může být celá řada. Uveďme například výpočetní náročnost nebo zvýšení generativní síly. Je jasné, že klíčovou roli zde bude hrát právě souhra jednotlivých prvků množiny. Rozlišují se dva základní typy gramatických systémů.

- ♢ sekvenční
- ♢ paralelní

CD gramatické systémy[6] jsou systémy sekvenční. Význam zkratky, stejně jako zkratky PC použité o odstavec níže, si můžete najít v seznamu zkratek, který je umístěn na konci práce. Důležitým faktem, který odlišuje tento systém od systému PC je to, že všechny části systému mají společnou větnou formu. Aktivní je vždy pouze jedna gramatika, a to ta, která právě pracuje na společné větné formě. V takovémto systému jsou nutné nějaké prvky, které udávají, kdy se právě aktivní gramatika stane neaktivní a předá řízení a větnou formu gramatice jiné. Které, o to se postará protokol. Příkladem takovýchto stop prvků mohou být například závorky nebo if podmínka. Nebo lze také říci, že daná gramatika bude pracovat právě k kroků.

Naproti tomu PC gramatické systémy[6] jsou paralelní. Zde má každá gramatika svou vlastní větnou formu a mohou tedy všechny pracovat paralelně, každá na své vlastní větné formě. V tomto typu gramatických systémů je nutný globální časovač. Taktéž je velmi podstatná existence takzvaných dotazovacích (query) symbolů. Jejich funkce je následující. Pokud část i vygeneruje symbol Q_j, pak aktuální větná forma komponenty j je zaslána části i, kde nahradí všechny výskyty Q_j. V každém PC systému existuje jedna komponenta, která

má status master, a jazyk generovaný touto komponentou je jazyk generovaný celým systémem.

2 CD gramatické systémy

Nyní se podívejme blíže na CD gramatické systémy. Abychom si mohli nadefinovat CD gramatický systém, musíme si nejdříve nadefinovat gramatiku samotnou. Gramatika G je tedy čtveřice $G = (N, \Sigma, P, S)$, kde

1. N je konečná množina non terminálních symbolů
2. Σ je konečná množina terminálních symbolů
3. P je konečná množina kartézského součinu

$$(N \cup \Sigma)^* N.(N \cup \Sigma)^* \times (N \cup \Sigma)^*$$

4. S$\in$N je startovací symbol gramatiky G

2.1 Definice

Pokud tedy máme definici gramatiky, můžeme si nadefinovat gramatický systém. CD gramatický systém nadefinujeme následovně:
CD gramatický systém stupně n, n$\geq$ 1 je n-tice

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

kde N a T jsou disjunktní abecedy, S $\in$ N a $P_1, ..., P_n$ jsou konečné množiny přepisovacích pravidel nad N $\cup$ T. N je množina neterminálu a T je množina terminálů. $P_1, ..., P_n$ jsou nazývány komponenty systému. Ještě, než přejdeme k definici jazyka, který je generován daným gramatickým systémem, musíme si nadefinovat derivační krok. Derivační kroky jsou totiž celkem čtyři. Uvedeme si všechny.

Nechť $\Gamma = (N, T, S, P_1, ..., P_n)$ je CD gramatický systém.
1. Pro každé $i \in 1, ..., n$ koncový derivační krok, provedený i-tou komponentou, značený $\Longrightarrow^t_{Pi}$ je definován následovně:
x $\Longrightarrow^t_{Pi}$ y pokud x $\Longrightarrow^*_{Pi}$ y a neexistuje žádné z $\in V_*$ takové, že y $\Longrightarrow_{Pi}$ z.

2. Pro každé $i \in 1, ..., n$ k-kroková derivace, provedená i-tou komponentou, značená $\Longrightarrow^{=k}_{Pi}$ je definována následovně:
x $\Longrightarrow^{=k}_{Pi}$ y pokud existují $x_1, ..., x_k \in (N\ cupT)^*$ takové, že
$x = x_1$, $y = y_{k+1}$, a pro každé $1 \leq j \leq k$,
$x \Longrightarrow_{Pi} x_{j+1}$.

3. Pro každé $i \in 1, ..., n$ nejvýše k-kroková derivace, provedená i-tou komponentou, značená $\Longrightarrow^{\leq k}_{Pi}$ je definována následovně:

x $\Longrightarrow_{P_i}^{\leq k}$ y pokud $x \Longrightarrow_{P_i}^{=k'} y$ pro nějaké $k' \leq k$.

4. Pro každé $i \in 1, ..., n$ nejméně k-kroková derivace, provedená i-tou komponentou, značená $\Longrightarrow_{P_i}^{\geq k}$ je definována následovně:
x $\Longrightarrow_{P_i}^{\geq k}$ y pokud $x \Longrightarrow_{P_i}^{=k'} y$ pro nějaké $k' \geq k$.

Sémantika této notace je docela jednoduchá. Pokud pracujeme v t- módu, musí každá komponenta přispět k řešení problému právě tak dlouho, dokud může. V k- módu udělá komponenta právě k po sobě jdoucích kroků. V dalších dvou módech je uděláno alespoň k nebo maximálně k kroků. V *-módu zůstane komponenta u provádění kroků tak dlouho, jak potřebuje. Nyní přejděme k definici jazyka generovaného daným systémem. Nejdříve nadefinujme množinu D následovně: D= $\{*, t\} \cup \{\leq k, \geq k, = k | k \geq 1\}$
Jazyk generovaný CD gramatickým systémem $\Gamma = (N, T, S, P_1, ..., P_n)$ v derivačním módu f $\in$ D je $L_f(\Gamma) = \{w \in T^* | S \Longrightarrow_{P_{i_1}}^{f} w_1 \Longrightarrow_{P_{i_2}}^{f} ... \Longrightarrow_{P_{i_m}}^{f} w_m = w,$ $m \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq m\}$.

2.2 Příklad

Z definice výše vyplývá, že v Γ je obsaženo několik jazyků. Která z komponent bude použita, je určeno na základě podmínek v D. Komponenta P_i začne pracovat v momentě, kdy se v řetězci w objeví levá strana pravidla z P_i. Na začátku je nedeterministicky určeno, která komponenta začne pracovat. Uvažujme následující příklad. Máme následující gramatické systémy:

$\Gamma_1 = (\{S, A, A', B, B'\}, \{a, b, c\}, S, P_1, P_2)$,

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

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

$\Gamma_2 = (\{S, A\}, \{a\}, S, P_1, P_2, P_3)$,

$P_1 = \{S \to AA\}$,

$P_2 = \{A \to S\}$,

$P_3 = \{A \to a\}$.

$\Gamma_3 = (\{S, A_1, ..., A_k, A'_1, ..., A'_k\}, \{a, b\}, S, P_1, P_2)$,

$P_1 = \{S \to S, S \to A_1 b A_2 b ... b A_k b\} \cup \{A'_i \to A_i | 1 \leq i \leq k\}$,

$P_2 = \{A_i \to aA_i'a, A_i \to aba | 1 \leq i \leq k\}, K \geq 1.$

$\Gamma_4 = (\{S, S'\} \cup \{A_i, A_i', A_i'' | 1 \leq i \leq k\}, \{a, b\}, S, P_0, P_1, ..., P_{3k}),$

$P_0 = \{S \to S', S' \to A_1 b A_2 b ... b A_k b\},$

$P_1 = \{A_1 \to A_1, A_1 \to aA_1'a\},$

$P_{i+1} = \{A_i' \to A_i'', A_{i+1} \to aA_{i+1}'a\}, 1 \leq i \leq k-1,$

$P_{k+1} = \{A_k' \to A_k'', A_1'' \to A_1'\},$

$P_{k+i+1} = \{A_i' \to A_i, A_{i+1}'' \to A_{i+1}'\}, 1 \leq i \leq k-2,$

$P_{2k} = \{A_{k-1}' \to A_{k-1}, A_k'' \to A_k\},$

$P_{2k+i} = \{A_i \to A_i, A_i \to aba\}, 1 \leq i \leq k, k \leq 2.$

$\Gamma_5 = (\{S, A, A'\}, \{a, b\}, S, P_1, P_2, P_3),$

$P_1 = \{S \to S, S \to AA, A' \to A\},$

$P_2 = \{A \to AA', A \to a\},$

$P_3 = \{A \to bA, A \to b\}.$

Tyto gramatické systémy generují následující jazyky:

$L_f(\Gamma_1) = \{a^n b^n c^m | m, n \leq 1\}, f \in \{= 1, \leq 1, , t\} \cup \{\leq k | k \geq 1\},$
$L_{=2}(\Gamma_1) = L_{\geq 2}(\Gamma_1) = \{a^n b^n c^n | n \leq 1\},$
$L_{=2}(\Gamma_1) = L_{\geq 2}(\Gamma_1) = \emptyset, k \geq 3,$
$L_t(\Gamma_2) = \{a^{2^{\overline{n}}} | n \geq 1\},$
$L_{=k}(\Gamma_3) = L_{\geq k}(\Gamma_3) = \{(a^n b)^{2k} | n \geq 1\},$
$L_{=2}(\Gamma_4) = L_{\geq 2}(\Gamma_4) = \{(a^n b)^{2k} | n \geq 1\},$
$L_{=2}(\Gamma_5) = L_{\geq 2}(\Gamma_5) = \{ww | w \in \{a, b\}^+\}.$

Nyní si dokážeme první tvrzení, abychom viděli, jak systém funguje. Nechť Γ_1 pracuje v módu 2. Stejně jako u libovolné jiné gramatiky musíme začít naši derivaci ze startovního symbolu S. Podíváme-li se na množinu pravidel, pak je jasné, že pouze P_1 může být aplikováno. Derivace tedy bude vypadat následovně:

$$S \Longrightarrow_{P_1} S \Longrightarrow_{P_1} AB.$$

Nyní, když se ve větné formě již neobjevuje S, nemůžeme nadále využívat P_1. Na AB se dá aplikovat pouze množina pravidel P_2. Pokud použijeme nonterminální pravidla, tak dostaneme:

$$AB \Longrightarrow_{P_2} aA'bB \Longrightarrow_{P_2} aA'bcB'.$$

Obecně můžeme z větné formy ve tvaru $a^iAb^jc^kB$ udělat derivační krok do tvaru $a^{i+1}A'b^{j+1}c^{k+1}B'$ Zapsáno matematicky:

$$a^iAb^jc^kB \Longrightarrow^{=2}_{P_1} a^{i+1}A'b^{j+1}c^{k+1}B'.$$

Na takovýto řetězec pak aplikujeme pravidlo z množiny P_1 a dostaneme:

$$a^{i+1}A'b^{j+1}c^{k+1}B' \Longrightarrow^{=2}_{P_2} a^{i+1}Ab^{j+1}c^{k+1}B.$$

Toto je jediná možnost, jak použít množinu pravidel P_1 v módu 2. V množině pravidel P_2 však máme možností více. Můžeme například použít jedno pravidlo s neterminálem a jedno pravidlo obsahující pouze terminální symboly. Ovšem u řetězce, který obsahuje pouze jeden neterminál různý od S, neexistuje řešení v množině P_1. Proto tedy musíme vždy použít buď dvě neterminální, nebo dvě terminální pravidla z množiny P_2. To znamená, že všechny derivace Γ_1 v módu =2 lze vyjádřit ve tvaru:

$$S \Longrightarrow^{=2}_{P_1} AB \Longrightarrow^{=2}_{P_2} aA'bcB' \Longrightarrow^{=2}_{P_1} aAbcB \Longrightarrow^{=2}_{P_2} ...$$

$$\Longrightarrow^{=2}_{P_2} a^nA'b^nc^n \Longrightarrow^{=2}_{P_1} a^nAb^nc^nB \Longrightarrow^{=2}_{P_2} a^{n+1}b^{n+1}c^{n+1}.$$

Toto platí pro libovolné $n \leq 0$ a proto tedy platí, že $L_{=2}(\Gamma_1) = \{a^nb^nc^n | n \leq 1\}$.

2.3 Generativní síla

Pokud se zaměříme na generativní sílu CD systémů pracujících v libovolném módu, brzy zjistíme, že se jejich síla nijak nemění. To znamená, že máme-li CD gramatický systém, který obsahuje pravidla regulární, bezkontextová, kontextová nebo pravidla třídy 0, pak je takovýto systém schopný generovat jazyky dané třídy. Tj. opět jazyky regulární, bezkontextové, kontextové nebo jazyky rekurzivně vyčíslitelné.

2.4 Hybridní systémy

Na tomto místě bych rád zmínil pár faktů o hybridních systémech, které jsou velmi zajímavé a především velmi blízké reálnému životu. Zatím jsme uvažovali pouze systémy homogenní. Tj. systémy, kde všechny komponenty pracují ve stejném režimu. V praxi se však ukázalo, že je výhodnější uvažovat o systému, kde jeho části pracují v různých módech. Takovýto systém se nazývá hybridní systém. Ten je definován následovně:

$$\Gamma = (N, T, S, (P_1, f_1), ..., (P_n, f_n)), n \geq 1,$$

kde $(N, T, S, P_1, ..., P_n)$ je běžný gramatický systém tak, jak jsme si ho nadefinovali a $f_i \in D, 1 \leq i \leq n$ je derivační mód i-té komponenty. Tyto módy mohou být obecně rozdílné! Více o těchto systémech se lze dočíst zde [6].

2.5 Týmový CD gramatický systém

Další z modifikací, které by neměly být opomenuty, jsou týmové CD gramatické systémy. U běžných CD systémů je podmínka, že pouze jedna komponenta může být v daný okamžik aktivní. Pokud tuto podmínku odstraníme, dostáváme tým CD gramatických systémů. CD gramatický systém s týmy je definován následovně:

$$\Gamma = (N, T, S, P_1, ..., P_n, R_1, ..., R_n), n, m \leq 1$$

$(N, T, S, P_1, ..., P_n)$ značí běžný gramatický systém a $R_1, ..., R_n$ jsou podmnožiny $P_1, ..., P_n$ zvané týmy.

Derivace s použitím týmu pak vypadá následovně:

$$x \Longrightarrow R_i \; tehdyajentehdy \; x = x_1A_1x_2A_2...x_sA_sx_{s+1} = x_1y_1x_2y_2...x_sy_sx_{s+1}$$

$$x_l \in (N \cup T)^*, 1 \leq l \leq s+1, A_r \rightarrow y_r \in P_{j_r}, 1 \leq r \leq s$$

Vzhledem k tomu, že se jedná o množinu, tak není určeno žádné pořadí aplikace jednotlivých pravidel. Zajímavý je především fakt, že týmy dokáží zásadně zvýšit generativní sílu CD gramatických systémů. Podle [6] stačí týmy o velikosti dva, abychom byli schopni generovat jazyky třídy 0. Přitom nám stačí, aby pravidla v týmu byla bezkontextová. Toto nám mimo jiné dává zcela nový způsob, jak zapisovat rekurzivně vyčíslitelné jazyky velmi přehledně.

2.6 Aplikace v praxi

Tento druh gramatického systému má své uplatnění při překladu programovacích jazyků. Tímto způsobem je interpretován například jazyk Python. Interpret je totiž složen ze dvou komponent, které si navzájem předávají řízení.

Podobný koncept jsem zvolil i ve své bakalářské práci, kde analyzátor jazyka taktéž sestává ze dvou komponent a předávání řízení je prováděno na základě stop prvků. V mém případě je jedna část analyzátoru založena na rekurzivním sestupu a druhá část na precedenční analýze. V momentě, kdy jedna z částí narazí na stop prvek, předá řízení druhé části. Tento způsob lze formálně popsat jako terminální mód gramatického systému. To znamená, že daná komponenta pracuje na překládaném řetězci tak dlouho, dokud je to možné. Pro ilustraci si si uvedeme jednoduchý příklad:

$if(((a+b)/2) > ((2*a)||(2*b)));$
{
$int\ c;$
$c := a + b/12;$
}

V tomto případě jsou veškeré aritmetické výrazy přeloženy jednou komponentou a příkazy jako if nebo for jsou přeloženy druhou komponentou. Samozřejmě se může také vyskytnout případ, že celý zdrojový kód bude mít na starosti pouze jedna komponenta a druhá se vůbec nedostane k překladu.

CD gramatciké systémy mají své využití také v multiagentních systémech jak dokládá kniha s názvem Grammar systems: a grammatical approach to distribution and cooperation [1]. V tomto případě jsou CD gramatické systémy použity pro popis komunikace mezi jednotlivými agenty a pro jejich kooperaci.

V této kapitole bylo osvětleno, co to jsou a jak fungují CD gramatické systémy. Tyto systémy jsou stejně jako PC systémy používány především pro zápis distribuovaného zpracování. Stojí na relativně jednoduchém základě a jsou snadno pochopitelné. Po zavedení určitých změn však disponují velmi velkou sílou. Při použití bezkontextových pravidel získáme totiž sílu přesahující sílu bezkontextových gramatik. Bylo také nastíněno použití tohoto modelu v praxi.

3 PC gramatické systémy

V této části se již budeme zabývat PC gramatickými systémy, které oproti CD systémům dovolují práci na více větných formách současně. Nejdříve si takový systém nadefinujme.

3.1 Definice

PC gramatický systém stupně n, $n \geq 1$, je (n+3)-tice

$$\Gamma = (N, K, T, (S_1, P_1), ..., (S_n, P_n)),$$

kde N je abeceda neterminálů, T je abeceda terminálů, K= $Q_1, ..., Q_n, P_i$ je konečná množina přepisovacích pravidel nad $N \cup K \cup T$ a $S_i \in N$ pro všechna $1 \leq i \leq n$.

Množiny $P_i,\ 1 \leq i \leq n$ jsou nazývány komponenty systému a elementy $Q_1, ..., Q_n$ z K se nazývají dotazovací symboly. Index i u Q_i odkazuje na i-tou komponentu z Γ.

Pokud bychom chtěli vyjádřit gramatiku jako součást PC gramatického systému Γ, lze to zapsat následovně $\Gamma = (N, K, T, G_1, ..., G_i)$, kde $G_i = (N \cup K, T, S_i, P_i),\ 1 \leq i \leq n$. Ještě než přejdeme k jazyku, který může gramatický systém generovat, je nutné si nadefinovat jeho derivaci.

Máme-li PC gramatický systémΓ $= (n, k, t, (S_1, P_1, ..., (S_n, P_n)))$ v podobě, v jaké byl nadefinován výše, pak pro dvě n-tice$(x_1, x_2, ...x_n), (y_1, y_2, ..., y_n)$, kde $x_i, y_i \in V_\Gamma^*, 1 \leq i \leq n$ a $x_1 \notin T^*$ pak píšeme $(x_1, ...x_n) \Longrightarrow (y_1, ..., y_n)$, pokud nastal jeden z následujících případů:

Pro každé $i, 1 \leq i \leq n,\ |x_i|_K = 0,\ 1 \leq i \leq n$ a pro každé i1 $\leq i \leq n$ máme buď $x_i \Longrightarrow y_i$ při použití pravidla z P_i, nebo $x_i = y_i \in T_*$

Existuje takové i, $1 \leq i \leq n$ takové, že $|x_i|_K > 0$. Nechť pro každé takové i, $x_i = z_1 Q_{i_1} z_2 Q_{i_2} ... z_t Q_{i_t} z_{t+1}, t \geq 1$, pro $z_j \in (N \cup T)^*, 1 \leq j \leq t+1$. Pokud $|x_{i_j}|_k = 0$, pro všechna j1 $\leq j \leq t$, pak $y_i = z_1 x_{i_1} z_2 x_{i_2} ... z_t x_{i_t} z_{t+1}$ a $y_{i_j} = S_{i_j}, 1 \leq j \leq t$. Pokud pro nějaké j $1 \leq j \leq t, |x_{i_j}|_K \neq 0$, pak $y_i = x_i$. Pro všechna i,1 $\leq i \leq n$, taková, že y_i není specifikováno výše, platí $x_i = y_i$.

Každá n-tice $(x_1, ..., x_n)$, kde $x_i \in V_\Gamma^*$ pro všechna i $1 \leq i \leq n$ je nazývána konfigurace. Konfigurace $(x_1, ..., x_n)$ přímo přechází v konfiguraci $(y_1, ..., y_n)$ pokud:

1. Žádný dotazovací symbol se neobjevuje v $(x_1, ..., x_n)$, pak máme komponentní derivaci, $x_i \Longrightarrow y_i$, v každé komponentě $P_i, 1 \leq i \leq n$ tzn., že v každé komponentě je použito jedno pravidlo, vyjma případu, že x_i je složeno z terminálů. V tom případě píšeme $x_i = y_i$.

2. Dotazovací symbol se vyskytuje v nějakém x_i. Pokud toto nastane, pak je provedena fáze, která se nazývá komunikační krok. Každý výskyt Q_i v x_i je nahrazen x_j za předpokladu, že samotné x_j neobsahuje žádný dotazovací symbol. Jinak řečeno, komponenta x_i je měněna, pouze pokud všechny cíle, do kterých směřují dotazovací symboly, samy neobsahují dotazovací symboly. V komunikačním kroku je dotazovací symbol Q_j nahrazen příslušným řetězcem x_j.Poté začne gramatika G_j opět přepisovat od startovacího symbolu. Důležité je, že komunikace má přednost před přepisováním. To znamená, že během komunikace nemohou být přepisovány žádné symboly. Dokud tedy nejsou nahrazeny všechny výskyty dotazovacích symbolů, které být nahrazeny mohou, neděje se žádné přepisování. Pokud nemůže být nějaký dotazovací symbol nahrazen v daném komunikačním kroku, protože obsahuje dotazovací symbol,

je nahrazen při prvním možném následujícím kroku.

Vezmeme-li v potaz výše řečené, pak je jasné, že pravidla typu $x_1 Q_i x_2 \Longrightarrow x$ nemohou být nikdy použita. Můžeme je tedy vypustit z PC gramatických systémů. Také nemá smysl definovat pravidla typu $(x_1, ...x_n) \Longrightarrow (y_1, ..., y_n)$, pokud je levá strana pravidla složena z terminálů.

Vzhledem k tomu, že PC gramatické systémy pracují paralelně, hrozí zde reálné nebezpečí uváznutí celého systému. Uváznutí může nastat ve dvou případech. Ten první je docela zřejmý. Je to stav, kdy se žádný dotazovací symbol nevyskytuje v řetězci, řetězec $(x_1, ...x_n)$ není terminálním řetězcem a zároveň nelze žádné pravidlo na daný řetězec aplikovat. Tento typ uváznutí může nastat po obou fázích. Jak po fázi přepisování, tak po fázi komunikace. Druhá možnost, jak by mohl celý systém uváznout, je také dobře známá z paralelních systémů. Jde o kruhovou závislost. Komponenta (P_{i_1}) volá (Q_{i_2}), následně (P_{i_2}) zavolá (Q_{i_3}), až $(P_{i_{k-1}})$ zavolá (Q_{i_k}) a (P_{i_k}) zavolá opět (Q_{i_1}). Je jasné, že takováto závislost nemůže být vyřešena, protože se komunikují pouze řetězce, které neobsahují dotazovací symboly. Nyní přejděme k definici jazyka, který je v případě, že nenastane uváznutí, daným systémem generován.

Jazyk generovaný PC gramatickým systémem Γ vypadá následovně:

$$L(\Gamma) = \{x \in T \mid (S_1, S_2, ..., S_i) \Longrightarrow^* (x, \alpha_2, ..., \alpha_n), \alpha_i \in V_\Gamma^*, 2 \leq i \leq n\}$$

Začneme tedy od n-tice startujících symbolů $(S_1, S_2, ..., S_i)$ a pokračujeme přes přepisovací a komunikační kroky, dokud komponenta P_1 nevyprodukuje řetězec terminálů. Komponenta P_1 se nazývá master. Je to proto, že v momentě, kdy tato první komponenta vytvoří řetězec terminálů, tak celý systém ukončí derivování nezávisle na tom, co obsahují ostatní komponenty.

PC gramatické systémy mají také svoje dělení. Existují dvě základní třídy gramatických systémů. Ty lze klasifikovat podle toho, jaký graf vytváří. Respektive podle tvaru grafu, jaký vytváří. Pro úplnost si uveďme přesnou definici.

Nechť $\Gamma = (N, K, T, (S_1, P_1), ..., (S_n, P_n))$ je PC gramatický systém. Pokud je pouze P_1 povoleno generovat dotazovací symboly($P_i \subseteq (N \cup T)^* \times (N \cup T)^*$ pro $2 \leq i \leq n$), pak říkáme, že Γ je centralizovaný gramatický systém. Pokud může generovat dotazovací symboly libovolná komponenta, pak je Γ necentralizovaný gramatický systém.

Další hledisko dělení gramatických systémů je podle toho, zda se vrací, či nevrací ke svému počátečnímu neterminálu. Gramatiku nazveme navracející se, pokud se poté, co poslala svůj řetězec jiné komponentě, vrátí ke svému startujícímu neterminálu S_i. Pokud se tak nestane, a gramatika tedy pokračuje v provádění derivačních kroků, pak ji nazveme nenavracející gramatikou. Pokud je gramatika nenavracející, pak je nutné vynechat podmínku $y_{i_j} = S_{i_j}$ v definici derivace.

PC gramatický systém nazveme regulární, bezkontextový, kontextový, nebo třídy 0, pokud pravidla tohoto systému jsou dané třídy. Pokud nazveme systém regulárním, tak se většinou jedná o systém pravě lineární. To znamená, že

pravidla mají tvar $A \longrightarrow xB$, $A \longrightarrow x$ kde A, B jsou neterminály a x je terminál.

3.2 Příklady

Ukažme si vše na několika jednoduchých příkladech. Uvažujme následující gramatické systémy:

$\Gamma_1 = (\{S_1, S_1^{'}, S_2, S_3\}, K, \{a, b\}, (S_1, P_1), (S_2, P_2), (S_3, P_3))$,

$P_1 = \{S_1 \rightarrow abc, S_1 \rightarrow a^2b^2c^2, S_1 \rightarrow aS_1^{'}, S_1 \rightarrow a^3Q_2, S_1^{'} \rightarrow aS_1^{'}, S_1^{'} \rightarrow a^3Q_2, S_2 \rightarrow bQ_3, S_3 \rightarrow c\}$,

$P_2 = \{S_2 \rightarrow bS_2\}$,

$P_3 = \{S_3 \rightarrow cS_3\}$,

$\Gamma_2 = (\{S_1, S_2\}, K, \{a, b\}, (S_1, P_1), (S_2, P_2))$,

$P_1 = \{S_1 \rightarrow S_1, S_1 \rightarrow Q_1Q_2\}$,

$P_2 = \{S_2 \rightarrow aS_2, S_2 \rightarrow bS_2, S_2 \rightarrow a, S_2 \rightarrow b\}$,

$\Gamma_3 = (\{S_1, S_2\}, K, \{a, b\}, (S_1, P_1), (S_2, P_2))$,

$P_1 = \{S_1 \rightarrow S_1, S_1 \rightarrow Q_1Q_2\}$,

$P_2 = \{S_2 \rightarrow aS_2, S_2 \rightarrow S_2b, S_2 \rightarrow ab\}$.

Výše uvedené gramatické systémy zjevně generují následující jazyky:

$L_r(\Gamma_1) = L_{nr}(\Gamma_1) = \{a^nb^nc^n \| n \geq 1\}$,

$L_r(\Gamma_2) = L_{nr}(\Gamma_2) = \{xx \| x \in \{a, b\}^+1\}$,

$L_r(\Gamma_3) = L_{nr}(\Gamma_3) = \{a^nb^ma^nb^m \| n, m \geq 1\}$.

Ukažme si nyní na gramatickém systému Γ_1, jak pracuje. Musíme začít přepisovat od startujících terminálů, tedy od (S_1, S_2, S_3). Poté aplikujeme třetí a páté pravidlo z P_1 a pravidla z P_2 a P_3. Pokud provedeme tyto derivační kroky, dostáváme:

$$(S_1, S_2, S_3) \Longrightarrow_r (aS_1^{'}, bS_2, cS_3) \Longrightarrow_r^* (a^{n+1}S_1^{'}, b^{n+1}S_2, c^{n+1}S_3).$$

Případně můžeme použít šesté pravidlo z množiny P_1:

$$(a^{n+1}S_1^{'}, b^{n+1}S_2, c^{n+1}S_3) \Longrightarrow_r (a^{n+4}Q_2^{'}, b^{n+1}S_2, c^{n+2}S_3).$$

Protože se nám vyskytl ve větné formě dotazovací symbol Q_2, musíme provést komunikační krok. $b^{n+2}S_2$ je posláno první komponentě, kde nahradí Q_2.

$$(a^{n+4}Q_2^{'}, b^{n+1}S_2, c^{n+2}S_3) \Longrightarrow_r (a^{n+4}b^{n+2}S_2, S_2, c^{n+2}S_3).$$

Deterministickým způsobem pak budeme provádět následující kroky:

$$(a^{n+4}b^{n+2}S_2, S_2, c^{n+2}S_3) \Longrightarrow_r (a^{n+4}b^{n+4}Q_3, bS_2, c^{n+3}S_3)$$

$$\Longrightarrow_r (a^{n+4}bn + 4c_{n+3}S_3, bS_2, S_3) \Longrightarrow_r (a^{n+4}b^{n+4}c_{n+3}S_3, bS_2, cS_3).$$

Tímto způsobem můžeme vyprodukovat všechny řetězce $a^nb^nc^n, n \geq 4$. To nám ovšem nestačí. Musíme být schopni vytvořit i řetězce kratší.

$$(S_1, S_2, S_3) \Longrightarrow_r (a^3Q_2, bS_2, cS_3) \Longrightarrow_r (a^3bS_2, S_2, cS_3) \Longrightarrow_r$$

$$\Longrightarrow_r (a^3b^3Q_3, bS_2, c^2S_3) \Longrightarrow_r (a^3b^3c^2S_3, bS_2, S_3) \Longrightarrow_r (a^3b^3c^3, b^2S_2, cS_3).$$

Takto vytvoříme řetězec $a^3b^3c^3$. Řetězce abc a $a^2b^2c^2$ dokáže vytvořit komponenta P_1 sama.

Systémy Γ_2 a Γ_3 pracují následovně: P_1 nedělá nic, dokud P_2 generuje řetězec. Pak P_1 vytvoří řetězec $Q1Q_2$. Tím pádem je řetězec z P_2 zaslán P_1 a je duplikován.Řetězec musí být tvořen terminály, jinak dojde k uváznutí. Všechny výše zmíněné gramatické systémy jsou centralizované. Jak vidíme, tak tyto relativně jednoduché gramatické systémy mohou generovat jazyky, které nejsou bezkontextové.

3.3 Nesynchronizované PC gramatické systémy

U všech předchozích gramatických systémů jsme uvažovali synchronizaci přepisovacích kroků. Tím pádem existovaly nějaké hodiny, které řídily celý systém. S každým tikem hodin musela každá komponenta použít přepisovací pravidlo, pokud systém není v komunikačním módu. Pokud je řetězec v komponentě již složen z terminálů, pak žádné pravidlo není aplikováno. Pokud takovou synchronizaci nevyžadujeme, pak mluvíme o nesynchronizovaném systému.

Označíme $L_{u,r}(\Gamma)$ a $L_{u,nr}(\Gamma)$ jazyky generované PC gramatickými systémy Γ v nesynchronizovaném návratném a nesynchronizovaném nenávratném módu. UY_nX označíme rodinu jazyků generovanou nesynchronizovanými gramatickými systémy. Uveďme si několik teorémů, které můžeme najít v této literatuře. [3]

$(i) UCPC_\infty X = X$, pro $x \in \{REG, LIN\}$.

$(ii) UPC_2REG - REG \neq \emptyset, UPC_\infty REG \subseteq CF, UPC_2LIN - CF \neq \emptyset, UCPC_2CF - CF \neq \emptyset$.

$(iii) UCPC_2REG$ obsahuje nesemilineární jazyky $UNCPC_2CF$ obsahuje jednopísmenné a neregulární jazyky.

$(iv)(CPC_\infty REG \cap NCPC_\infty REG) - UCPC_\infty CF \neq \emptyset$.

Body (i) a (iv) ukazují, že pokud rozsynchronizujeme gramatický systém, snížíme podstatně generativní sílu, zatímco nesynchronizované PC gramatické systémy jsou stále velmi silné, jak ukazují body (ii) a (iii).

$\Gamma_1 = (\{S_1, S_2, A, B, \}, K, \{a, b\}, \{S_1, P_1\}, \{S_2, P_2\}),$

$P_1 = \{S_1 \longrightarrow bQ_2, B \longrightarrow bQ_2, B \longrightarrow b\},$

$P_2 = \{S_2 \longrightarrow aS_2, S_2 \longrightarrow aA, A \longrightarrow aB\},$

$\Gamma_2 = (\{S_1, S_2, A, B, \}, K, \{a, b\}, \{S_1, P_1\}, \{S_2, P_2\}),$

$P_1 = \{S_1 \longrightarrow aQ_2, S_1 \longrightarrow aA, B \longrightarrow bA\},$

$P_2 = \{S_2 \longrightarrow aQ_1, A \longrightarrow bB, A \longrightarrow b\}.$

Jazyky, které z takovýchto gramatických systémů dostaneme, jsou následující:

$L_{u,nr}(\Gamma_1) = \{(ba^n)^m b | m \geq 1, n \geq 2\}$, nesemilineární jazyk,
$L_{u,r}(\Gamma_2) = \{a^{2m} a^{2s+1} b^{2t+1} | m, s, t \geq 1, s \geq t\}$, neregulární jazyk.

3.4 Gramatické systémy s komunikací na příkaz

Ve všech předchozích PC gramatických systémech jsme uvažovali případ, kdy byla komunikace vyžádána komponentou, která vygenerovala dotazovací symbol případně symboly. Tomuto způsobu iniciace komunikace proto říkáme na žádost. To ovšem není jediná možnost, jak komunikovat. Z praktického hlediska je však stejně důležitý i jiný přístup nazývaný komunikace na příkaz. Tento přístup je iniciován komponentou, která posílá zprávy. PC gramatické systémy komunikující na příkaz modelující WAVE diagram byly uvedeny E. Tsuhaj-Varjuem v tomto díle [2].

Základní myšlenkou je systém sestávající z několika gramatik tak, jak je to běžné u všech PC gramatických systémů pracujících odděleně, každá na své vlastní větné formě. Každá z těchto gramatik má na svém vstupu regulární nebo jiný jazyk, který pracuje jako vstupní filtr. V určitých případech je přepisování přerušeno a každá komponenta pošle svou aktuální větnou formu ostatním komponentám. Zejména těm, jejichž vstupní filtr obsahuje danou větnou formu. Stejně jako u ostatních PC gramatických systémů, tak i u těchto je jazyk generován master komponentou jazykem, který generuje celý systém.

Musí však být vyřešeno ještě několik otázek, než bude možné formulovat formální model:

1. Kdy má být ověřeno, zda mohou být větné formy komunikovány? Lze to dělat po každém kroku, pokud jsme v synchronizovaném módu, nebo můžeme nechat všechny komponenty pracovat tak dlouho, dokud mohou, a poté provést komunikaci.
2. Co by mělo být komunikováno? Celá větná forma, nebo pouze její část?
3. Jaká bude další větnou formou komponenty, která komunikovala svůj řetězec? Tedy jde o systém, který se navrací, či nikoli.
4. Uvažujme, že je několik zpráv zasláno jedné komponentě? Zde je možné najít několik řešení. Nejpřirozenější se jeví dvě varianty. Buď ze všech zpráv, které byly zaslány dané komponentě, vybereme jednu, a to buď nedeterministicky, nebo s přihlédnutím k prioritám komponent, nebo provedeme konkatenaci všech zpráv podle pořadí komponent. Jiné přístupy, které se inspirují podobnými praktickými problémy, lze nalézt zde [5].

3.5 Aplikace v praxi

Na rozdíl od CD gramatických systémů je aplikace PC gramatických systémů možné tam, kde je zapotřebí paralelismu. I zde lze uvažovat o aplikaci v multiagentních systémech. Pokud například nelze nebo nemůže libovolný agent provést zadanou operaci, vygeneruje symbol, pomocí kterého požádá o provedení operace jiného agenta nebo budou na dané operaci spolupracovat.

Nejdůležitější aplikací PC gramatických systémů však nadále zůstává překlad jazyků. Můžeme si snadno představit, že používáme v našem souboru se zdrojovým kódem několik programovacích jazyků. Uvažujme pro začátek situaci, že jsou programovací jazyky od sebe navzájem odděleny například takto:

```
**********Jazyk C**********
 double a,b,c;

scanf("%f", a);
printf("soucet je:%f",a+b+c);



    **********Assembler**********


ohrkone:
push bp ; stav zasobniku (shora dolu):
```

```
mov bp,sp ; x1:2,y1:2,x2:2,y2:2,cs:2,ip:2,bp:2
mov al,0 ; false (kone se neohrozuji)
mov bl,[bp+12] ; x1 do bl
sub bl,[bp+8] ; (x1-x2) do bl
jz konec1 ; kone ve stejnem radku se neohrozuji
jns pokr1 ;
neg bl ; -(x1-x2) do bl
```

V takovém případě není potřeba generovat žádné dotazovací symboly. Je možné zpracovat celý zdrojový soubor paralelně. Každá komponenta zpracuje svoji část zdrojového kódu následujícím způsobem. Najde si svůj startující neterminální symbol a pak aplikuje přepisovací pravidla ve vhodném pořadí. Pokud je u takovéhoto typu překladu vygenerován dotazovací symbol, lze to považovat za chybu.

Nyní uvažujme situace, že jednotlivé programovací jazyky jsou smíchány dohromady:
ohrkone:

```
push bp ; stav zasobniku (shora dolu):
mov bp,sp ; x1:2,y1:2,x2:2,y2:2,cs:2,ip:2,bp:2
mov al,0 ; false (kone se neohrozuji)
mov bl,[bp+12] ; x1 do bl
sub bl,[bp+8] ; (x1-x2) do bl
jz konec1 ; kone ve stejnem radku se neohrozuji
jns pokr1 ;
neg bl ; -(x1-x2) do bl
var x1,x2,y1,y2,i,j:Byte;
ch:Char;
Function ohrkone(x1,y1,x2,y2:Byte):Boolean;External;
begin
Randomize;
ClrScr;
repeat
GoToXY(2,1);
Write('Zadejte pozadovanou cinnost (O..Ohrozuje, K..Konec): ');
repeat
ch:=UpCase(ReadKey);
until ch in ['O','K'];
ClrScr;
case ch of
'O':begin
x1:=Random(8)+1;
y1:=Random(8)+1;
repeat
x2:=Random(8)+1;
```

```
y2:=Random(8)+1;
until not ((x1=x2) and (y1=y2));
for i:=1 to 8 do begin
for j:=1 to 8 do begin
GoToXY(20+2*j,3+i);
if ((i=x1) and (j=y1)) or ((i=x2) and (j=y2))
then Write('X')
else Write('.');
end;
end;
GoToXY(22,15);
WriteLn('Kone se ohrozuji: ',ohrkone(x1,y1,x2,y2),' ');
end;
end;
until ch='K';
end.
```

Zde lze při kontrole zdrojového kódu uplatnit naplno PC gramatické systémy. Každá komponenta bude překládat svou část kódu a v momentě kdy narazíme na assemblerovský kód uvnitř jazyka pascal se dotážeme této komponenty, zda je assemblerovská část v pořádku.

Dá se uvažovat i o aplikaci tohoto postupu v kryptografii. Při lámání šifry bude každému procesoru, počítači nebo jiné výpočetní jednotce přidělena určitá část výpočtu. Jednotky pak mohou pracovat nezávisle na sobě a v případě potřeby budou spolu komunikovat přes dotazovací symboly.

Tato kapitola diskutovala gramatické systémy typu PC. Nejdříve byla provedena definice tohoto typu gramatického systému a následně na příkladech ukázáno, jak v praxi fungují. Bylo provedeno rozdělení systémů podle různých hledisek. Dále se kapitola zabývala několika specifickými typy gramatických systémů, jako jsou nesynchronizované gramatické systémy nebo systémy komunikující na příkaz. Závěrem jsem se pokusil osvětlit kde se dají takovýto model použít v praxi.

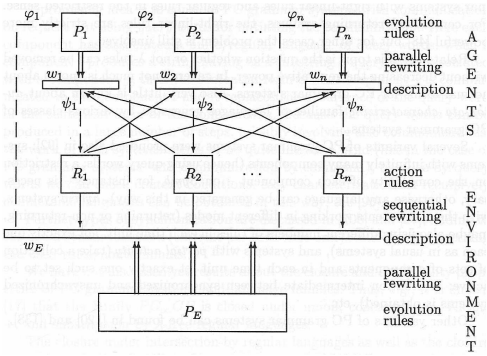
4 Příbuzné modely

Nyní se ještě ve zkratce podívejme na některé modely, které jsou příbuzné gramatickým systémům. Bude se jednat eco gramatiky, a pak také o test tube systémy.

4.1 Eco gramatiky

V případě EG[6] se jedná o relativně nedávno uvedený gramatický model, který je zaměřen na vztah mezi prostředím a agenty, kteří v něm působí. Takovýto systém se nazývá ekosystém. EG lze interpretovat jako zobecnění jak CD, tak i PC gramatických systémů. Hlavní myšlenkou při definici tohoto mod-

elu bylo zachycení popisu různých forem života tak, aby bylo možné tuto notaci použít při modelování multiagentních systémů, nebo jiných umělých forem života. Základním předpokladem tohoto modelu je fakt, že komponenty tohoto systému mohou být popsány řetězci nad danými abecedami. Schéma gramatického systému je zachyceno na obrázku 2.1.



Celý systém je ve své podstatě rozdělen na dvě části. Rozlišujeme prostředí, které je popsáno řetězci w_E nad abecedou V_E. Prostředí se vyvíjí podle množiny pravidel P_E, která jsou typu 0. Na druhou stranu agenti, kterých může být obecně n, jsou jsou popsáni řetězci w_i nad abecedou V_i a vyvíjí se podle pravidel P_i, která jsou rovněž typu 0. Formálně je tedy EG systém popsán následovně:

$$\Sigma = (E, A_1, A_2, ..., A_n),$$

kde

$$E = (V_E, P_E) \text{ je prostředí,}$$
$$A_i = (V_i, P_i, R_i, \varphi_i, \psi_i), 1 \leq i \leq n, \text{ jsou jednotliví agenti.}$$

Konvence pro popis je následující:

$$V_E, V_i \text{ jsou abecedy } (V_E \cap V_i = \emptyset), 1 \leq i \leq n,$$
$$P_E \text{ konečná množina pravidel typu 0 nad } V_E,$$
$$P_i \text{ konečná množina pravidel typu 0 nad } V_i, 1 \leq i \leq n,$$
$$R_i \text{ konečná množina přepisovacích pravidel nad } V_E \text{ nebo } \bigcup_{j \neq i} V_j, 1 \leq i \leq n,$$
$$\varphi_i : V_E^* \longrightarrow 2^{P_i}, 1 \leq i \leq n,$$
$$\psi_i : V_i^* \longrightarrow 2^{R_i}, 1 \leq i \leq n.$$

Aplikace v praxi

Tyto gramatiky jsou vhodné zejména pro popis agentních a multiagentních systémů. Bohužel se mi nepodařilo najít žádné informace o převedení tohoto modelu do praxe.

4.2 Test tube systémy

Posledním modelem, který bude nastíněn jsou test tube systémy[4]. Jedná se o mechanismus zpracování symbolů, který má architekturu PC gramatického systému komunikujícího na příkaz. Komunikace v rámci systému je prováděna pomocí redistribuce obsahu jednotlivých rour(tubes). Takovéto systémy se ukázaly výpočetně úplné. Mají tedy sílu vyjádřit rekurzivně vyčíslitelné jazyky.

Tato krátká kapitolka měla za úkol představit některé další modely, které jsou příbuzné gramatickým systémům. Byly prezentovány EG systémy a test tube systémy.

5 Závěr

Cílem této práce bylo obeznámit se s gramatickými systémy jak z teoretického tak i z praktického hlediska. Bylo provedeno rozdělení gramatických systémů na tři kategorie. CD gramatické systémy, PC gramatické systémy a příbuzné modely. CD gramatické systémy pracují sekvenčně. Nicméně i tak mají velkou sílu, neboť i s bezkontextovými pravidly lze generovat jazyky typu 0. Obvykle pracují všechny komponenty ve stejném módu což však neplatí o hybridních CD gramatických systémech, kde je každé komponentě přiřazen libovolný mód. Změny derivačního módu nijak neovlivní jejich sílu. Hlavní uplatnění těchto systémů je v překladu a interpretaci jazyků.

Další probranou kategorií jsou PC gramatické systémy. Jejich největší výhodou je fakt, že jsou paralelní. Každá komponenta tedy může pracovat samostatně, dokud není třeba kooperace s ostatními pomocí dotazovacích symbolů. PC gramatické systémy dělíme na returning a non-returning podle toho, zda se vracejí po komunikačním kroku zpět do počátečního stavu. Existují i speciální verze PC gramatických systémů jako například nesynchronizované PC gramatické systémy a jiné. Jejich aplikace je vhodná a možná pro popis jakýchkoli paralelních systémů např. v kryptografii a podobně.

Závěrem byly zmíněny i příbuzné modely. Jmenovitě celkem zajímavý koncept ekogramatik, za jejichž vznikem stojí snaha o stvoení formalismu pro popis živých struktur a také test tube systémy.

References

[1] Csuhaj-Varju, E. a. k.: *Grammar systems: a grammatical approach to distribution and cooperation*. OPA (Amsterdam) B.V., 1994.

[2] Csuhaj-Varju, E. a. k.: *Grammar systems with WAVE-like communication*. Computers and AI, 1996.

[3] Csuhaj-Varju, E. a. k.: *Grammar Systems. A gramatical approach to distribution and cooperation*. Gordon and Breach, London, 2004.

[4] Freund, R. a. k.: Test tube systems with cutting/recombination operations. [Online;accessed 16-May-2009].
URL `http://helix-web.stanford.edu/psb97/freund.pdf`

[5] Hillis, W.: *The connection machine*. The MIT press, Cambridge, Mass., 1985.

[6] Rozenberg, G.; Salomaa, A. (editoři): *Handbook of formal languages Vol. 2 Linear Modeling: Background and application*. Springer, Berlin, ISBN 3-540-60648-3, str. 528.

Seznam zkratek a symbolů

PC - parallel communicating
CD - cooperating distributed
EG - eco gramatiky