VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

SYNTAKTICKÁ ANALÝZA ZALOŽENÁ NA NĚKOLIKA GRAMATIKÁCH

PARSING BASED ON SEVERAL GRAMMARS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ONDŘEJ KOUMAR**
AUTHOR

VEDOUCÍ PRÁCE **prof. RNDr. ALEXANDR MEDUNA, CSc.**
SUPERVISOR

BRNO 2024

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Citace

KOUMAR, Ondřej. *Syntaktická analýza založená na několika gramatikách.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. RNDr. Alexandr Meduna, CSc.

Syntaktická analýza založená na několika gramatikách

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondřej Koumar
11. dubna 2024

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Obsah

Seznam obrázků

Dotazy

Kapitola 1

Úvod

Kapitola 2

Základy teorie formálních jazyků

Základní přehled matematických znalostí potřebných k pochopení této kapitoly (a tím i celé této práce) je popsán například v [8], druhá kapitola.

2.1 Abeceda, řetězec a jazyk

2.2 Chomského hierarchie

2.3 Bezkontextová gramatika

Následující definice převzaty z [11].

Definice 2.3.1. *Bezkontextová gramatika* je čtveřice $G = (N, T, P, S)$, kde:

- N je konečná množina neterminálních symbolů,
- T je konečná množina terminálních symbolů,
- P je konečná množina přepisovacích pravidel ve tvaru $A \to x$, $A \in N$ a $x \in (N \cup T)^*$,
  - formálně vyjádřeno, množina P přepisovacích pravidel pro bezkontextové gramatiky je podmnožinou kartézského součinu $P \subseteq N \times (N \cup T)^*$.
- $S \in N$ je výchozí symbol gramatiky.

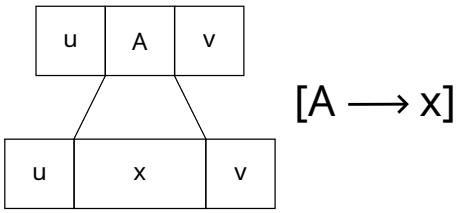
Konvence 2.3.1. Pro označení bezkontextových gramatik bude v textu dále využívána zkratka BKG. Neterminální symboly budou označovány jako neterminály, terminální symboly jako terminály.

Derivační krok

Myšlenkou derivačního kroku je přepsat aktuální řetězec na řetězec nový za použití pravidla BKG.

Definice 2.3.2. Nechť $G = (N, T, P, S)$ je BKG, $\alpha, \beta \in (N \cup T)^*$ a $p = A \to x \in P$.
Mezi řetězci α a β platí binární relace $\underset{G}{\Rightarrow}$, nazývaná *přímá derivace*, jestliže můžeme řetězce α a β vyjádřit ve tvaru

$$\alpha = uAv$$
$$\beta = uxv$$



Obrázek 2.1: Ilustrace derivačního kroku za použití pravidla $A \to x$.

kde u a v jsou libovolné řetězce z $(N \cup T)^*$. Říká se také, že gramatika G provádí *derivační krok* z uAv do uxv, respektive z α do β.

To znamená, že pokud lze přepsat řetězec $\alpha = uAv$ na řetězec $\beta = uxv$ za použití pravidla $A \to x$, potom mezi řetězci α a β je relace přímé derivace, zapsáno $\alpha \Rightarrow \beta$.

Sekvence derivačních kroků

Definice 2.3.3. Nechť $G = (N, T, P, S)$ je BKG a $\alpha, \beta \in (N \cup T)^*$.
Mezi řetězci α a β platí relace $\Rightarrow^+$ nazývaná *derivace*, jestliže existuje posloupnost přímých derivací $v_{i-1} \Rightarrow v_i, i \in \{1, \ldots, n\}, n \geq 1$ taková, že platí

$$\alpha = v_0 \Rightarrow v_1 \Rightarrow \ldots \Rightarrow v_{n-1} \Rightarrow v_n = \beta.$$

Tato posloupnost se nazývá *derivace délky n*. Platí-li $\alpha \Rightarrow \beta$, pak řetězec β lze *generovat* z řetězce α, nebo také β je *derivovatelný* z α v gramatice G. Relace $\Rightarrow^+$ je tranzitivním uzávěrem relace přímé derivace $\Rightarrow$. Symbolem $\Rightarrow^n$ se značí n-tá mocnina přímé derivace $\Rightarrow$.

Jinými slovy, pokud řetězec α derivuje řetězec β v nenulovém počtu kroků a zároveň β derivuje řetězec γ v nenulovém počtu kroků, pak je zřejmé, že α derivuje γ v nenulovém počtu kroků, zapsáno $\alpha \Rightarrow^+ \gamma$.

Definice 2.3.4. Nechť $G = (N, T, P, S)$ je BKG a $\alpha, \beta \in (N \cup T)^*$.
Jestliže v G platí pro řetězce α a β relace $\alpha \Rightarrow^+ \beta$ nebo identita $\alpha = \beta$, pak $\alpha \Rightarrow^* \beta$. Relace $\Rightarrow^*$ je reflexivním a tranzitivním uzávěrem relace přímé derivace $\Rightarrow$.

Reflexivní uzávěr relace přímé derivace $\Rightarrow$ znamená, že řetězec přímo derivuje sám sebe v nula krocích. Také je možné říci, že se nepoužije žádné pravidlo k přepsání řetězce na sebe sama, zapsáno $\alpha \Rightarrow^0 \alpha$.

2.4 Konečný automat

Definice související s konečnými automaty převzaty z [5], není-li řečeno jinak.

Definice 2.4.1. *Konečný automat* je pětice

$$M = (Q, \Sigma, R, s, F),$$

kde

- Q je konečná množina stavů,
- Σ je konečná vstupní abeceda, $Q \cap \Sigma = \emptyset$,

- R je konečná relace, $R \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$,
  - je nazývána *množinou pravidel* ve tvaru $qa \to p,\ q, p \in Q,\ a \in \Sigma \cup \{\varepsilon\}$. Jakákoli uspořádaná trojice $(q, a, p) \in R$ je pravidlem, zapsáno $qa \to p$.
  - Další možnost zápisu je zobrazení (*přechodová funkce*) $R : Q \times \Sigma \to 2^Q$ [11].
- $s \in Q$ je počáteční stav automatu,
- $F \subseteq Q$ je konečná množina koncových stavů.

Konvence 2.4.1. Pro označení konečných automatů bude dále v textu použito zkrácené KA.

U konečných automatů popisujeme jejich *konfiguraci* – ve kterém stavu se nachází a jaký řetězec na vstupní pásce.

Definice 2.4.2. Nechť $M = (Q, \Sigma, R, s, F)$ je KA. Dle autorů v [11] je konfigurace M uspořádaná dvojice

$$(q, w) \in Q \times \Sigma^*.$$

Nechť X_M značí množinu všech konfigurací M.

Pomocí konfigurací můžeme definovat *přechody* KA, které reprezentují jejich výpočetní kroky. Výpočetní krok znamená přechod z jedné konfigurace do druhé za přečtení symbolu ze vstupní pásky.

Definice 2.4.3. Nechť $M = (Q, \Sigma, R, s, F)$ je KA.
Binární relace $\vdash_M$, nazývána přechodem KA M, je

$$\beta \vdash_M \chi,$$

kde $\beta = (q, ax)$, $\chi = (p, x) \in X_M$, a $qa \to p \in R$. Je to binární relace na množině konfigurací – pokud $\beta \vdash_M \chi$, pak $(\beta, \chi) \in X_M \times X_M$. Pokud $a = \varepsilon$, není ze vstupní pásky přečten žádný symbol.

Symboly $\vdash_M^n, \vdash_M^+$ a $\vdash_M^*$ nechť značí příslušně n-tou mocninu pro $n \geq 0$, tranzitivní a reflexivně-tranzitivní uzávěr relace $\vdash_M$ podobně jako u derivačního kroku BKG (2.3.2), čímž reprezentují *sekvenci přechodů* KA M. Například pro konfigurace β a χ platí $\beta \vdash_M^+ \chi$ právě tehdy, když

$$\beta = c_0 \vdash_M c_1 \vdash_M \ldots \vdash_M c_{n-1} \vdash_M c_n = \chi,$$

pokud $\beta, \chi, c_1, \ldots, c_{n-1} \in X_M,\ n \geq 1$.

Konvence 2.4.2. Bude-li z kontextu jasné, že se jedná o přechod automatu M, pak bude relace přechodu $\vdash$ psána bez indexu M.

Definice 2.4.4. Nechť $M = (Q, \Sigma, R, s, F)$ je KA.
Jazyk přijímaný M, značen $L(M)$, je

$$L(M) = \{w \in \Sigma^* : sw \vdash^* f,\ f \in F\}.$$

Jsou to řetězce takové, po jejichž zpracování skončí M v koncovém stavu.

Definice 2.4.5. Nechť $M = (Q, \Sigma, R, s, F)$ je KA.
M je *KA bez ε-přechodů*, pokud pro každé pravidlo $qa \to p \in R$ platí, že $a \neq \varepsilon$.

Definice 2.4.6. Nechť $M = (Q, \Sigma, R, s, F)$ je KA bez ε-přechodů.
M je *deterministický KA* právě tehdy, když pro každé $q \in Q$ a každé $a \in \Sigma$ neexistuje více než jedno $p \in Q$ takové, že $qa \to p \in R$. Jinými slovy, z jednoho stavu není možné přejít do několika stavů přečtením stejného symbolu.

2.5 Zásobníkový automat

Zásobníkový automat je rozšíření konečného automatu, popsaného v definici 2.4.1, o zásobník. Zásobník slouží jako paměťové médium, na které si automat může ukládat informace v podobě symbolů zásobníkové abecedy. Následující definice převzaty z [5, 11].

Definice 2.5.1. *Zásobníkový automat* (ZA) je sedmice

$$M = (Q, \Sigma, \Gamma, R, s, S, F),$$

kde:

- Q, Σ, s, F jsou definovány stejně jako v definici 2.4.1,
- Γ je konečná zásobníková abeceda,
- R je konečná relace $R \subseteq \Gamma \times Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \times Q$ nazývána množinou pravidel ZA,
  - každé pravidlo (A, q, a, w, p) může být zapsáno ve tvaru $Aqa \to wp$, kde $A \in \Gamma$, $q, p \in Q$, $a \in \Sigma \cup \{\varepsilon\}$, $w \in \Gamma^*$ [7],
- $S \in \Gamma$ je počáteční symbol na zásobníku.

Konvence 2.5.1. Zásobníkové automaty budou dále značeny zkráceně ZA.

Definice 2.5.2. Nechť $M = (Q, \Sigma, \Gamma, R, s, S, F)$ je ZA.
Konfigurací ZA nazveme trojici

$$(\alpha, q, w) \in \Gamma^* \times Q \times \Sigma^*,$$

kde

- α je obsah zásobníku,
- q je aktuální stav automatu (*řídící jednotky*),
- w je doposud nepřečtená část vstupního řetězce, jehož první symbol je aktuálně pod čtecí hlavou. Pokud $w = \varepsilon$, všechny symboly byly ze vstupní pásky již přečteny.

Nechť X_M reprezentuje množinu všech konfigurací ZA M.

Definice 2.5.3. Nechť $M = (Q, \Sigma, \Gamma, R, s, S, F)$ je ZA.
Přechod ZA je binární relace definována nad množinou konfigurací (podobně jako v definici 2.4.3) v podobě

$$\beta \vdash \chi,$$

kde $\beta = (uA, q, av)$, $\chi = (uw, p, v)$ a $Aqa \to wp \in R$. Symbolem A je reprezentován vrchol zásobníku a symbol a nechť reprezentuje aktuální symbol pod čtecí hlavou. Pokud $w = \varepsilon$, pak se pouze vyjme A ze zásobníku bez náhrady. Relace $\vdash^n$, $\vdash^+$, $\vdash^*$ jsou opět n-tou mocninou relace, tranzitivním a reflexivně-tranzitivním uzávěrem.

Oproti klasickým KA se při přechodu musí pracovat se zásobníkem, ze kterého se vyjme symbol A, namísto něj se vloží symbol w.

Automat M přijímá řetězec právě tehdy, když $Ssw \vdash^* uf$ v M; $u \in \Gamma^*, f \in F$.

Definice 2.5.4. Nechť $M = (Q, \Sigma, \Gamma, R, s, S, F)$ je ZA.
Jazyk přijímaný M je

$$L(M) = \{w : w \in \Sigma^*,\ Ssw \vdash^* f,\ f \in F\}.$$

Příklad 2.5.1. Nechť $M = (\{s, f\}, \{a, b\}, \{S, a\}, R, s, S, \{f\})$ je ZA a

$$R = \{Ssa \to as,\ asas \to aas,\ asb \to f,\ afb \to f\}.$$

Počáteční konfigurace nechť je $(S, s, aaabbb)$. Přechody M budou vypadat následovně:

$$\begin{aligned}
(S, s, aaabbb) &\vdash (a, s, aabbb) && [Ssa \to as] \\
&\vdash (aa, s, abbb) && [asa \to aas] \\
&\vdash (aaa, s, bbb) && [asa \to aas] \\
&\vdash (aa, f, bb) && [asb \to f] \\
&\vdash (a, f, b) && [afb \to f] \\
&\vdash (\varepsilon, f, \varepsilon) && [afb \to f]
\end{aligned}$$

Rozšířený zásobníkový automat

Rozšířené zásobníkové automaty reprezentují přirozené rozšíření klasických ZA, které na zásobníku mohou pracovat pouze s jeho vrcholem. Rozšířené zásobníkové automaty mohou provádět expanzi symbolů na zásobníku v libovolné hloubce, jinak pracují indenticky. Text a definice v této kapitole převzaty z [5, 8], není-li řečeno jinak.

Definice 2.5.5. *Rozšířený zásobníkový automat* (RZA) je sedmice

$$M = (Q, \Sigma, \Gamma, R, s, S, F),$$

kde:

- Q, Σ, s, S, F jsou definovány stejně jako u klasických ZA (2.5.1),
- Γ je zásobníková abeceda, $\mathbb{N},\ Q$, a Γ jsou navzájem disjunktní, $\Sigma \subseteq \Gamma$[[**opravdu $\subseteq$ a ne $\subset$?**]]

  a $\Gamma \setminus \Sigma$ obsahuje speciální symbol $\#$ (*spodní symbol*), který je považován za dno zásobníku,
- R je konečná relace

  $$\begin{aligned}
  &(\mathbb{N} \times Q \times (\Gamma \setminus (\Sigma \cup \{\#\})) \times Q \times (\Gamma \setminus \{\#\})^+) \cup \\
  &(\mathbb{N} \times Q \times \{\#\} \times Q \times (\Gamma \setminus \{\#\})^* \{\#\}),
  \end{aligned}$$

  místo uspořádané pětice $(m, q, A, p, v) \in R$ píšeme $mqA \to pv \in R$ a R nazýváme množinou pravidel.

  - Z definice R lze vidět, že jsou dva typy přechodových pravidel, dají se zapsat ve tvaru

    $$\begin{aligned}
    &mqA \to pv, \\
    &mq\# \to pv\# \text{ [6]}.
    \end{aligned}$$

    O jejich použití a rozdílech je psáno v definici 2.5.7.

Konvence 2.5.2. Rozšířené zásobníkové automaty budou dále označeny zkratkou RZA.

Přechody RZA pracují, stejně jako přechody ZA, nad konfiguracemi. Konfigurace jsou podobné těm u klasických ZA, nicméně stav zásobníku musí končit symbolem # a zbytek řetězce na zásobníku jej nesmí obsahovat.

Definice 2.5.6. Nechť $M = (Q, \Sigma, \Gamma, R, s, S, F)$ je RZA.
Konfigurací RZA nazveme trojici

$$(q, w, \alpha) \in Q \times \Sigma^* \times (\Gamma \setminus \{\#\})^*\{\#\}.$$

Nechť X_M reprezentuje množinu všech konfigurací RZA M.

Další podobnost je práce s řetězci abedecedy Σ a s pravidly při přechodech. Jediná změna od klasických ZA již byla zmíněna na začátku této kapitoly – při přechodech se na vrcholu zásobníku mohou měnit celé řetězce.

Definice 2.5.7. Nechť $M = (Q, \Sigma, \Gamma, R, s, S, F)$ je RZA a $\beta,\ \chi \in X_M$ jeho konfigurace.
M vyjme (anglicky *pops*) ze zásobníku symbol a přejde z konfigurace β do konfigurace χ za přečtení symbolu vstupní pásky, symbolicky zapsáno

$$\beta\ {}_p\vdash \chi,$$

pokud konfigurace jsou ve tvaru $\beta = (q, au, az),\ \chi = (q, u, z),\ a \in \Sigma,\ u \in \Sigma^*,\ z \in \Gamma^*$. M rozšíří (anglicky *expands*) symbol na zásobníku a přejde z konfigurace β do konfigurace χ za přečtení symbolu ze vstupní pásky, symbolicky zapsáno

$$\beta\ {}_e\vdash \chi,$$

pokud konfigurace jsou ve tvaru $\beta = (q, w, uAz),\ \chi = (p, w, uvz)$ a zároveň $mqA \to pv \in R$; $q, p \in Q,\ w \in \Sigma^*,\ A \in \Gamma,\ u, v, z \in \Gamma^*$; m je hloubka symbolu A v zásobníku, respektive u obsahuje $m - 1$ *nevstupních* symbolů (symboly a, pro které platí $\{a\ :\ a \in \Gamma \setminus \Sigma\}$). Pro ilustraci, že automat udělá přechod $\beta\ {}_e\vdash \chi$ podle pravidla $mqA \to pv$, píšeme

$$\beta\ {}_e\vdash \chi\ \ [mqA \to pv].$$

M udělá *přechod* z β do χ, psáno

$$\beta \vdash \chi$$

právě tehdy, když M udělá vyjmutí symbolu $\beta\ {}_p\vdash \chi$ nebo expanzi symbolu $\beta\ {}_e\vdash \chi$. Tranzitivní uzávěry ${}_p\vdash^+, {}_e\vdash^+$, reflexivně-tranzitivní uzávěry ${}_p\vdash^*, {}_e\vdash^*$ a n-té mocniny ${}_p\vdash^n, {}_e\vdash^n$ jsou definovány standardně. Alternativní definice přechodu je popsána v [11].

Obrázek 2.2: [[obrazek p-prechodu a e-prechodu rza]]

Pokud existuje $n \in \mathbb{N}$ takové, že pro všechna pravidla M platí, že jejich hloubka je maximálně rovna n, pak říkáme, že M je hloubky maximálně n, zapsáno ${}_nM$.

Definice 2.5.8. Nechť ${}_nM = (Q, \Sigma, \Gamma, R, s, S, F)$ je RZA hloubky $n \in \mathbb{N}$.
Jazyk přijímaný automatem M je

$$L({}_nM) = \{w \in \Sigma^* : (s, w, S\#) \vdash^* (f, \varepsilon, \#) \text{ v } {}_nM,\ f \in F\}.$$

Tyto jazyky jsou přijímané vyprázdněním zásobníku a zároveň přechodem do koncového stavu automatu. Dále existují jazyky přijímané RZA pouze vyprázdněním zásobníku, přičemž automat se nemusí dostat do koncového stavu.

Definice 2.5.9. Nechť ${}_nM = (Q, \Sigma, \Gamma, R, s, S, F)$ je RZA hloubky $n \in \mathbb{N}$.
Jazyk přijímaný automatem M vyprázdněním zásobníku je

$$E({}_nM) = \{w \in \Sigma^* : (s, w, S\#) \vdash^* (q, \varepsilon, \#) \text{ v } {}_nM,\ q \in Q\}.$$

Příklad 2.5.2. Nechť ${}_2M = (\{s, q, p, f\}, \{a, b, c\}, \Sigma \cup \{A, S, \#\}, R, s, S, \{f\})$, kde R je

$$\begin{array}{lll} 1sS \to qAA, & 1qA \to fab, & 1fA \to fc, \\ 1qA \to paAb, & 2pA \to qAc. & \end{array}$$

Mějme počáteční konfiguraci $(s, aabbcc, S\#)$. M udělá nejdříve dvakrát expanzi – jednou expanduje startovací symbol na AA, přičemž A není vstupní symbol, takže expanze proběhne podruhé.

$$\begin{array}{rl} (s, aabbcc, S\#) \ {}_e\vdash\ (q, aabbcc, AA\#) & [1sS \to qAA] \\ {}_e\vdash\ (p, aabbcc, aAbA\#) & [1qA \to paAb] \end{array}$$

Na aktuální konfiguraci je vidět, že na vrcholu zásobníku i aktuálně čtený symbol je a. M tento symbol vyjme.

$$(p, aabbcc, aAbA\#) \ {}_p\vdash\ (p, abbcc, AbA\#)$$

Následující krok bude znovu expanze, tentokrát hlouběji v zásobníku.

$$(p, abbcc, AbA\#) \ {}_e\vdash\ (q, abbcc, AbAc\#) \quad [2pA \to qAc]$$

Dále přijímání řetězce probíhá stejným způsobem jako v předešlých krocích. **[[Příklad v knize - přeskočení kroku z 5 na 6, z kroku 7 na 8 je psána expanze, ale nic se v konfiguraci nezměnilo - vysvětlím osobně]]**

$$\begin{array}{rll} (q, abbcc, AbAc\#) \ {}_e\vdash\ (q, abbcc, abbAc\#) & [1qA \to fab] \\ {}_p\vdash\ (f, bbcc, bbAc\#) & \\ {}_p\vdash\ (f, bcc, bAc\#) & \\ {}_p\vdash\ (f, cc, Ac\#) & \\ {}_e\vdash\ (f, cc, cc\#) & [1fA \to fc] \\ {}_p\vdash\ (f, c, c\#) & \\ {}_p\vdash\ (f, \varepsilon, \#) & \end{array}$$

Kapitola 3

Důležité pojmy syntaktické analýzy

Kapitola 4

Gramatické systémy

Veškeré definice a znalosti použité v této kapitole převzaty z [2, 3, 9], není-li řečeno jinak. [[**Jak uvést tuto kapitolu?**]]

4.1 Kooperující distribuované gramatické systémy

Kooperující distribuovaný (*cooperating distributed*) gramatický systém stupně n je systém gramatik, které mezi sebou sdílejí množinu neterminálů i terminálů a startovací symbol. Spolupracují mezi sebou předáváním řízení derivace aktuálně zpracovávaného řetězce dle pravidel nastaveného *derivačního režimu*.

Definice 4.1.1. Kooperující distribuovaný gramatický systém je $(n+3)$-tice

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

kde:

- N, T, a S jsou definovány stejně jako v definici 2.3.1,
- P_i je konečná množina pravidel ve tvaru $A \to x$, kde pravidla jsou definována stejně jako v definici 2.3.1, nazývaná *komponentou* systému, $i \in \{1, \ldots, n\}$,
- i-tá gramatika systému se zapisuje jako $G_i = (N, T, S, P_i)$

Alternativní definice pro CDGS je $\Gamma = ((N, T, S, P_1), \ldots, (N, T, S, P_n))$.

Konvence 4.1.1. Pro označení kooperujících distribuovaných gramatických systémů bude dále využito zkrácené *CDGS*, případně *CD gramatické systémy*.

Derivační krok v CDGS

Notace derivačního kroku v CDGS je

$$x \,{}_i\Rightarrow^f y.$$

Tento zápis znamená, že řetězec $x \in (N \cup T)^*$ derivuje řetězec $y \in (N \cup T)^*$ v i-té komponentě za použití *derivačního režimu* f.

Derivační režimy

Prvním příkladem je režim $*$, který byl v definici 2.3.4 uveden pro bezkontextovou gramatiku a princip v CD gramatických systémech je podobný. Při použití tohoto derivačního režimu řetězec $x \in (N \cup T)^*$ derivuje řetězec $y \in (N \cup T)^*$ v libovolném počtu kroků v i-té komponentě, zapsáno $x \,{}_i\Rightarrow^* y$. Je možné předat řízení derivace jiné komponentě i v případě, že ve stejné komponentě lze s derivací stejného řetězce pokračovat.

Podobným příkladem je režim *ukončovací*, který spočívá v nutné derivaci řetězce v dané komponentě, dokud je to možné. Značí se písmenem t. Jsou dvě nutné podmínky, aby y bylo derivovatelné z x v komponentě G_i režimem t.

1. $x \,{}_i\Rightarrow^* y$ – v dané komponentě lze posloupností derivačních kroků získat řetězec y z řetězce x,
2. $y \,{}_i\not\Rightarrow z$ pro všechna $z \in (N \cup T)^*$ – ve stejné komponentě nelze nalézt další pravidlo, které by derivovalo y.

Jinými slovy, pokud můžeme z aktuálního řetězce v aktuální komponentě odvodit řetězec nový, *musí* se s derivací v této komponentě pokračovat.

Další derivační režimy:

- alespoň k derivací, tedy $x \,{}_i\Rightarrow^{\geq k} y$,
- nejvíce k derivací, tedy $x \,{}_i\Rightarrow^{\leq k} y$,
- právě k derivací, tedy $x \,{}_i\Rightarrow^{=k} y$,

kde $k \in \mathbb{N} \cup \{0\}$ a i symbolizuje i-tou komponentu gramatického systému.

Derivačních režimů existuje mnohem více**[[Zdroj?? - Mluvili jsme o tom na konzultaci a říkáte to i v záznamu přednášky TID. Stačil by asi klidně jen nějaký odkaz na výčet.]]**

, nejsou ale pro tuto práci podstatné. Tyto režimy mohou být reprezentovány jako množina, což pomůže definovat další pojmy v následující podkapitole o generovaném jazyce.

Definice 4.1.2. Nechť $k \in \mathbb{N}$ a $*$, t představují derivační režimy.
Potom množina

$$D = \{*, t\} \cup \{\leq k, \geq k, = k\}$$

reprezentuje derivační režimy použitelné v CD gramatických systémech.

Jazyk generovaný CD gramatickým systémem

Než bude definován samotný jazyk, je vhodné definovat pomocnou množinu, která reprezentuje *možné derivace* z řetězců.

Definice 4.1.3. Nechť $\Gamma = (N, T, S, P_1, \ldots, P_n)$.
Potom

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

je množina všech řetězců v derivovatelných z u v j-té komponentě za použití derivačního režimu f.

Definice 4.1.4. Nechť $\Gamma = (N, T, S, P_1, \ldots, P_n)$.
Jazyk generovaný systémem Γ za derivačního režimu f je

$$L_f(\Gamma) = \{w \in T^* : \text{ existují } v_0, v_1, \ldots, v_m \text{ takové, že } v_k \in F(G_{j_k}, v_{k-1}, f), \\ k \in \{1, \ldots, m\},\ j_k \in \{1, \ldots, n\},\ v_0 = S,\ v_m = w \text{ pro } m \geq 1\}.$$

Výsledný řetězec w, který vznikl postupnou derivací startovacího symbolu v_0. Měl několik mezikroků, které jsou reprezentovány řetězci $v_1, \ldots, v_{m-1}$. Každý řetězec v_k, kde $k \in \{1, \ldots, m\}$ byl zderivován z řetězce v_{k-1} v komponentě G_{j_k}, kde $j_k \in \{1, \ldots, n\}$, za derivačního režimu f.

Příklad 4.1.1. Nechť $\Gamma = (\{S, A\}, \{a\}, S, P_1, P_2, P_3)$, kde $P_1 = \{S \to AA\}$, $P_2 = \{A \to S\}$, $P_3 = \{A \to a\}$. Nechť $f = t$, Γ pracuje na ukončovacím derivačním režimu.

Počáteční řetězec je počáteční symbol, tedy S. Je pouze jedna možnost, jak S přepsat, a to použitím pravidla z P_1, $S \to AA$.

$$S \to AA \quad [S \to AA]$$

Díky tomu, že v komponentě P_1 už neexistuje pravidlo, kterým by mohla derivace dále pokračovat, řízení derivace je předáno komponentě jiné. Aktuálně zpracovávaný řetězec je AA. Komponenty P_1 a P_2 mají pravidla pro přepsání neterminálu A. Při použití derivačního režimu t je zřejmé, že celý řetězec bude přepisovat pouze jedna komponenta. Při předání komponentě P_3 je řetězec přepsán na aa,

$$AA \to aA \quad [A \to a]$$
$$aA \to aa \quad [A \to a]$$

komponenta P_2 stejný řetězec přepíše na SS.

$$AA \to SA \quad [A \to S]$$
$$SA \to SS \quad [A \to S]$$

$$SS \to AAS \quad [S \to AA]$$
$$AAS \to AAAA \quad [S \to AA]$$

Ze startovacího symbolu dostaneme buď terminál a nebo dva startovací symboly. To znamená, že pokaždé, kdy se AA přepíše na SS, počet terminálů a se zdvojnásobí. Jazyk generovaný systémem Γ za derivačního režimu t, $L(\Gamma)_t = \{a^{2^n},\ n \geq 1\}$.

Klasifikace skupin CD jazyků

CD jazyky jsou jazyky, které jsou generovány CD gramatickými systémy. Tyto jazykové rodiny se rozdělují podle různých typů použitých CDGS. Jejich označení je

$$CD_x^y(f),$$

kde:

- y určuje použití ε-pravidel:
  - $y = \varepsilon$ – ε-pravidla v komponentách jsou povolena,

- y chybí – ε-pravidla se v komponentách nemohou vyskytovat,

- x je stupeň gramatického systému:
  - $n, n \geq 1$ – gramatický systém může obsahovat maximálně n komponent,
  - ∞ – gramatický systém nemá omezen počet komponent,

- f je derivační režim, $f \in D$.

Příklad 4.1.2. $CD_6^{\varepsilon}(t)$ je rodina jazyků generována CD gramatickými systémy s maximálně šesti komponentami, povolenými ε-pravidly a pracujícími nad ukončovacím režimem.

Hybridní CD gramatické systémy

Hybridní CD gramatické systémy nabízí možnost definovat derivační režim samostatně pro jednotlivé komponenty systému.

Definice 4.1.5. *Hybridní CDGS* je n-tice $\Gamma = (N, T, S, (P_1, f_1), \ldots, (P_n, f_n))$, kde:

- N, T, P_i, S jsou definovány stejně jako v klasických CDGS (4.1.1),

- f_i je derivační režim i-té komponenty, $f_i \in D$ pro všechna $i \in \{1, \ldots, n\}$.

Jazyk generovaný těmito gramatickými systémy je velmi podobný jazykům generovaným klasickými CDGS. Jediný rozdíl je v použití derivačního režimu příslušné komponenty, který je v definici označen jako f_{j_k}, místo společného derivačního režimu pro celý gramatický systém.

Definice 4.1.6. Nechť $\Gamma = (N, T, S, (P_1, f_1), \ldots, (P_n, f_n))$.
Jazyk generovaný systémem Γ,

$$L(\Gamma) = \{w \in T^* : \text{existují } v_0, v_1, \ldots, v_n \text{ takové, že } v_k \in F(G_{j_k}, v_{k-1}, f_{j_k}),$$
$$k \in \{1, \ldots, m\}, j_k \in \{1, \ldots, n\}, v_0 = S, v_m = w \text{ pro } m \geq 1\}.$$

Zápis jazykových skupin generovaných hybridními CDGS je

$$X\, CD_{x,v}^{y}(f),$$

kde:

- x, y, f jsou definovány stejně jako u nehybridních CDGS,

- v je nepovinné omezení počtu pravidel v komponentách:
  - m – každá komponenta $P_i, i \in \{1, \ldots, n\}$ obsahuje nejvíce m pravidel, $m \geq 1$,
  - ∞, chybí – maximální počet pravidel pro komponenty není omezen,

- X určuje determinismus gramatického systému:
  - D – gramatický systém je deterministický, tedy pro každé $A \to u, A \to w \in P_i$, $i \in \{1, \ldots, n\}$ platí, že $u = w$. Jinými slovy, pro všechny neterminály platí, že pro jejich přepis neexistují pravidla (ve *všech* komponentách), která by měla různé pravé strany.
  - *nic* – gramatický systém je nedeterministický. To znamená, že v *každé* komponentě existuje pravidlo pro nějaký neterminál, které má různé pravé strany.
  - H – gramatický systém je hybridní a obsahuje alespoň jednu nedeterministickou komponentu a jednu deterministickou komponentu. Nezapisuje se derivační režim, který je určen samostatně pro každou komponentu.

4.2 Paralelní komunikující gramatické systémy

Paralelní komunikující (*parallel communicating*) – PC gramatický systém stupně n je systém gramatik, v němž každá začíná vlastním startovacím symbolem (*axiomem*), které sdílí množinu neterminálů i terminálů a nově množinu komunikačních symbolů. Komunikační symboly slouží ke komunikaci na vyžádání. Kdykoliv se vyskytnou ve větné formě libovolné komponenty, proběhne *komunikační krok* (také nazýván *c-derivační krok*).

Komunikačních symbolů je stejné množství jako komponent v PC gramatickém systému, $\{Q_1, \ldots, Q_n\}$, kde index symbolu Q_i odkazuje na komponentu P_i [4].

Definice 4.2.1. Paralelní komunikující gramatický systém stupně $n, n \geq 1$ je $(n+3)$-tice

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

kde:

- N, T jsou definovány stejně jako u bezkontextových gramatik (2.3.1),
- $K = \{Q_1, \ldots, Q_n\}$ je množina komunikačních symbolů (N, K, T jsou navzájem disjunktní); index i komunikačního symbolu Q_i koresponduje s indexem i-té komponenty,
- $P_i, i \in \{1, \ldots, n\}$ je množina pravidel nazývané komponenty, stejně jako u CD gramatických systémů (4.1.1),
- i-tá gramatika systému je konstrukt $G_i = (N \cup K, T, S_i, P_i), i \in \{1, \ldots, n\}$.

Konvence 4.2.1. Pro označení paralelních komunikujících gramatických systémů bude dále využito zkrácené *PCGS* nebo *PC gramatické systémy.*

Derivační kroky v PCGS

V PC gramatických systémech existují dva druhy derivačního kroku, a to g-derivační krok a c-derivační krok. První zmíněný slouží k přímé derivaci řetězce v rámci jedné komponenty bez zásahu komponent jiných a druhý slouží pro vzájemnou pomoc při derivaci řetězce. PC gramatický systém *vždy* preferuje c-derivační krok nad g-derivačním krokem. Ten se provede pokaždé, obsahuje-li alespoň jeden z řetězců $x_i, i \in \{1, \ldots, n\}$ alespoň jeden komunikační symbol.

Derivační kroky pracují nad *konfigurací* PCGS. Následující definice převzata z [10].

Definice 4.2.2. Nechť $\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n))$ je PC gramatický systém. Potom n-tice

$$(x_1, \ldots, x_n),\ x_i \in (N \cup K \cup T)^*,\ i \in \{1 \ldots, n\}$$

se nazývá *konfigurací* Γ. $(S_1, \ldots, S_n)$ je *počáteční konfigurací* Γ.

Nutná podmínka k proběhnutí libovolného derivačního kroku je $x_1 \notin T^*$. Pokud tato situace nastane, už je vygenerována věta jazyka definovaného PCGS a dále se s kroky nepokračuje. Více v definici 4.2.

g-derivační krok

Nechť $\Gamma = (N, K, T, (P_1, S_1), \ldots, (P_n, S_n))$ je PCGS a $(x_1, \ldots x_n)$, $(y_1, \ldots, y_n)$ jsou konfigurace Γ. Γ udělá g-derivační krok, formálně zapsáno

$$(x_1, \ldots, x_n) \,{}_g\Rightarrow (y_1, \ldots, y_n)$$

pokud $alph(x_i) \cap K = \emptyset$ a zároveň:

- $x_i \Rightarrow y_i$ v $G_i = (N, K, T, (S_i, P_i))$ – y_i je přímo derivován z x_i v gramatice G_i (komponentě P_i), nebo
- $x_i = y_i \in T^*$ – x_i již je řetězcem terminálů

pro všechna $i \in \{1, \ldots, n\}$.

c-derivační krok

Někdy taky nazýván *komunikační krok*. Tento koncept umožňuje gramatikám spolupracovat a vzájemně si mezi sebou měnit vygenerované řetězce pomocí komunikačních symbolů. Algoritmus ukazující princip komunikačního kroku je následující:

Algoritmus 1 c-derivační krok v PCGS

```
1: Vstup: konfigurace (x_1, ..., x_n)
2: Výstup: konfigurace (y_1, ..., y_n)
3:
4: for all i ∈ {1, ..., n} do
5:     z_i ← x_i
6: end for
7: for all i ∈ {1, ..., n} do
8:     if alph(x_i) ∩ K ≠ ∅ and foreach Q_j in x_i: alph(x_j) ∩ K = ∅ then
9:         for all Q_j in x_i do
10:            z_j ← S_j          ▷ vynecháno, pokud PCGS pracuje na nevracejícím se režimu
11:            zaměň Q_j za x_j v x_i
12:            z_i ← x_i                          ▷ x_i = řetězec, který vznikl o krok zpět
13:        end for
14:    end if
15: end for
16: proveď (x_1, ..., x_n) _c⇒ (y_1, ..., y_n) s y_i = z_i, i ∈ {1, ..., n}
```

Přímá derivace v PCGS

Definice 4.2.3. Konfigurace $(x_1, \ldots, x_n)$ přímo derivuje konfiguraci $(y_1, \ldots, y_n)$, zapsáno

$$(x_1, \ldots, x_n) \Rightarrow (y_1, \ldots, y_n)$$

právě tehdy, když

$$(x_1, \ldots, x_n) \;{}_g\Rightarrow (y_1, \ldots, y_n)$$

nebo

$$(x_1, \ldots, x_n) \;{}_c\Rightarrow (y_1, \ldots, y_n).$$

Jazyk generovaný PCGS

Generování větné formy končí v momentě, kdy první komponenta dosáhne řetězce terminálů a na řetězcích ostatních komponent nezáleží.

Definice 4.2.4. Nechť $\Gamma = (N, K, T, (P_1, S_1), \ldots, (P_n, S_n))$ je PCGS. Jazyk generovaný Γ je stejný jako jazyk generovaný jeho první komponentou:

$$L_f(\Gamma) = \{w \in T^* : (S_1, \ldots, S_n) \Rightarrow_f^* (w, \alpha_2, \ldots, \alpha_n), \\ for\ \alpha_i \in \{N \cup T \cup K\},\ i \in \{2, \ldots, n\},\ f \in \{r, nr\}\},$$

kde r a nr specifikuje *vracející* nebo *nevracející* PCGS.

Vracející a nevracející se režim

Pokud PC gramatický systém pracuje na vracejícím se režimu, potom komponenty, které v rámci komunikačního kroku poslaly svůj řetězec jiným komponentám, generují řetězec od svého axiomu. Při nevracejícím se režimu komponenty pokračují ve zpracovávání aktuálního řetězce.

Tato skutečnost se projeví na samotném komunikačním kroku, u kterého se vynechá přiřazení axiomu do řetězce z_j – neprovede se krok na řádku 10 algoritmu 1.

Centralizované PCGS

Centralizované PC gramatické systémy mají tu vlastnost, že pouze první komponenta (nazývaná *master*) systému může generovat komunikační symboly a tím žádat ostatní komponenty o řetězce. Řeší jeden z možných případů uváznutí, kdy komponenty v cyklu zavádějí komunikační symboly a donekonečna se provádí stejná sekvence komunikačních kroků.

Definice 4.2.5. Nechť $\Gamma = (N, K, T, (S_1, P_1), \ldots, (S_n, P_n))$ je PCGS. Pokud pouze P_1 může uvést komunikační symboly, formálně

$$P_i \subseteq (N \cup T)^* \times (N \cup T)^* \text{ pro } i \in \{2, \ldots, n\},$$

potom Γ je centralizovaný PC gramatický systém. Jinak je necentralizovaný.

Příklady

V prvním příkladu bude pouze demonstrován princip obou derivačních kroků na systému, který generuje jen velmi omezený jazyk.

Příklad 4.2.1. Nechť $\Gamma = (\{S_1, S_2, S_3\}, \{Q_3\}, \{a, b\}, (S_1, \{S_1 \to Q_3\}), (S_2, \{S_2 \to a\}), (S_3, \{S_3 \to b\}))$ je PCGS. Počáteční konfigurace Γ je zřejmě (S_1, S_2, S_3). Víme, že PC gramatické systémy *vždy* preferují c-krok nad g-krokem, je-li to možné. Nutná podmínka je, aby alespoň jeden z řetězců v aktuální konfiguraci obsahoval alespoň jeden komunikační symbol, což aktuálně není splněno. Γ tedy udělá g-krok.

$$(S_1, S_2, S_3) \ {}_g\Rightarrow\ (Q_3, a, b)$$

Nyní je v konfiguraci komunikační symbol, což indikuje, že bude následovat komunikační krok. Při postupu podle algoritmu 1 je postup následující:

1. Zavedení pomocných řetězců $z_1 = Q_3$, $z_2 = a$, $z_3 = b$ podle řádků 4 – 6.

2. Kontrola, zda řetězec x_i, $i \in \{1, \ldots, n\}$ z původní konfigurace obsahuje komunikační symboly (podmínka $alph(x_i) \cap K \neq \emptyset$).

   - V tomto příkladu splňuje podmínku pouze řetězec x_1, který obsahuje Q_3.

3. Pokud x_i obsahuje komunikační symboly Q_j, z každého se přečte index j a proběhne kontrola, zda všechny j-té řetězce konfigurace (x_j) *neobsahují* komunikační symboly. Tato část koresponduje s podmínkou **foreach** Q_j **in** $x_i : alph(x_i) \cap K = \emptyset$ na řádku 8.

   - Řetězec x_1 obsahuje jeden komunikační symbol Q_3, jehož index odkazuje na řetězec x_3 z aktuální konfigurace. Hodnota řetězce x_3 je b, ten žádné další komunikační symboly neobsahuje.

4. Každý Q_j v x_i se nahradí za x_j, pokud prošel podmínkou v předchozím kroku. Zároveň se z_j nastaví na počáteční symbol j-té komponenty. Tyto kroky jsou v algoritmu na řádcích 9 – 12.

   - Komunikační symbol Q_3 bude v x_1 nahrazen řetězcem b z x_3. Dále $x_3 \leftarrow S_3$ a $z_1 \leftarrow x_1$. Aktuálně $z_1 = b$, $z_2 = a$, $z_3 = S_3$.

5. Proveď $(x_1, \ldots, x_n) \ {}_c\Rightarrow (y_1, \ldots, y_n)$, kde $y_i = z_i$ pro $i \in \{1, \ldots, n\}$.

   - Nová konfigurace (y_1, y_2, y_3) bude stejná, jako pomocná konfigurace (z_1, z_2, z_3), a to (b, a, S_3).

Γ provede komunikační krok z (Q_3, a, b) do (b, a, S_3).

$$(Q_3, a, b) \ {}_c\Rightarrow (b, a, S_3)$$

Další kroky už Γ provádět nebude, protože řetězec generovaný první komponentou je již řetězec terminálních symbolů. V tomto příkladu byla použita sémantika vracejícího se PCGS, nicméně při použití nevracejícího by jazyk vypadal stejně, jen výsledná konfigurace by se lišila v řetězci x_3.

$$L(\Gamma)_r = L(\Gamma)_{nr} = \{b\}$$

Ve druhém příkladu bude demonstrováno několik derivačních kroků a bude se zkoumat výsledný generovaný jazyk.

Příklad 4.2.2. Nechť $\Gamma = (\{S_1, S_1', S_2, S_3\}, K, a, b, c, (S_1, P_1), (S_2, P_2), (S_3, P_3))$, kde:

$$\begin{aligned} P_1 = \{&S_1 \to abc, S_1 \to a^2b^2c^2, S_1 \to aS_1', S_1 \to a^3Q_2, \\ &S_1' \to aS_1', S_1' \to a^3Q_2, S_2 \to b^2Q_3, S_3 \to c\}, \\ P_2 = \{&S_2 \to bS_2\}, \\ P_3 = \{&S_3 \to cS_3\}. \end{aligned}$$

Komunikační symboly může uvést pouze komponenta P_1, a proto se jedná o *centralizovaný* PC gramatický systém.

Počáteční konfigurace je (S_1, S_2, S_3). Γ může generovat konfiguraci (aS_1', bS_2, cS_3) za použití pravidel $(S_1 \to aS_1', S_2 \to bS_2, S_3 \to cS_3)$.

$$(S_1, S_2, S_3) \Rightarrow (aS_1', bS_2, cS_3)$$

Zřejmě při použití pravidel $(S'_1 \to aS'_1, S_2 \to bS_2, S_3 \to cS_3)$ n-krát po sobě je možné generovat konfigurace $(a^nS'_1, b^nS_2, c^nS_3),\ n \geq 1$.

$$(aS'_1, bS_2, cS_3) \Rightarrow^* (a^nS'_1, b^nS_2, c^nS_3)$$

Je vidět, že komponenty P_2 a P_3 neustále používají ta samá pravidla, což je logické vzhledem k jejich kardinalitě. Pro zjednodušení zápisu, při každém g-derivačním kroku bude zmíněno pouze pravidlo komponenty P_1, pravidla komponent P_2 a P_3 budou vždy stejná.

Z konfigurace $(a^nS'_1, b^nS_2, c^nS_3)$ se na rozdílný výsledek dá dostat pouze za použití pravidla $S'_1 \to a^3Q_2$.

$$(a^nS'_1, b^nS_2, c^nS_3) \Rightarrow (a^{n+3}Q_2, b^{n+1}S_2, c^{n+1}S_3)$$

Provedeme c-derivační krok a zaměníme Q_2 za x_2, nový x_2 bude S_2, který je axiomem G_2.

$$(a^{n+3}Q_2, b^{n+1}S_2, c^{n+1}S_3) \Rightarrow (a^{n+3}b^{n+1}S_2, S_2, c^{n+1}S_3)$$

Jediné pravidlo pro symbol S_2 v komponentě P_1 je $S_2 \to b^2Q_3$.

$$(a^{n+3}b^{n+1}S_2, S_2, c^{n+1}S_3) \Rightarrow (a^{n+3}b^{n+3}Q_3, bS_2, c^{n+2}S_3)$$

Zaměníme Q_3 za x_3 nový x_3 bude S_3, který je axiomem G_3.

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

Pro S_3 je v P_1 také pouze jediné pravidlo, které se může aplikovat, a to $S_3 \to c$.

$$(a^{n+3}b^{n+3}c^{n+2}S_3, bS_2, S_3) \Rightarrow (a^{n+3}b^{n+3}c^{n+3}, bbS_2, cS_3)$$

x_1 je aktuálně řetězec terminálů, tudíž na něj nelze aplikovat žádné další pravidlo a neobsahuje komunikační symboly. Je tedy generovaným jazykem pro Γ ve vracejícím se režimu. Takto by vypadaly kroky pro nevracející se režim (konfigurace se začnou lišit od výskytu komunikačních symbolů):

$$\begin{aligned}(a^{n+3}Q_2, b^{n+1}S_2, c^{n+1}S_3) &\Rightarrow (a^{n+3}b^{n+1}S_2, b^{n+1}S_2, c^{n+1}S_3)\\ &\Rightarrow (a^{n+3}b^{n+3}Q_3, b^{n+2}S_2, c^{n+2}S_3)\\ &\Rightarrow (a^{n+3}b^{n+3}c^{n+2}S_3, b^{n+2}S_2, c^{n+2}S_3)\\ &\Rightarrow (a^{n+3}b^{n+3}c^{n+3}, b^{n+3}S_2, c^{n+3}S_3)\end{aligned}$$

Je tedy zřejmé, že:

$$L(\Gamma)_r = L(\Gamma)_{nr} = \{a^nb^nc^n,\ n \geq 1\}.$$

Kapitola 5

Použití gramatických systémů pro popis jazyka Koubp

[[základní popis použitého GS, jak je spojený s CDGS a potažmo PCGS]]
[[popsat možný deadlock mezi neterminály codeblock a statement]]
[[indexace neterminálů]]
[[ll tabulka, která obsahuje uspořádané dvojice]]

Kapitola 6

Implementace syntaktického analyzátoru pro jazyk Koubp [[rozdelit jinak sekce? co pripadne z implementace doplnit?]]

[[uvod do teto kapitoly - navod na pouziti a build system?]]

6.1 Přijímaný jazyk

Jazyk *Koubp* je založený na jazyce IFJ22, který je podmnožinou jazyka PHP 8, jenž byl specifikován v rámci zadání projektu do předmětu Formální jazyky a překladače v akademickém roce 2022/2023. **[[idealne citovat zadani projektu?]]**

Některé aspekty jazyků jsou společné. Oba dva jazyky jsou strukturované, podporují definici proměnných a funkcí. Hlavní tělo programu se skládá z prolínání sekvence příkazů a definic funkcí, které se mohou vzájemně rekurzivně volat. Vstupním bodem programu není funkce `main()`, jak lze nalézt například u jazyka C [1], analýza probíhá od začátku souboru. V uživatelem definovaných funkcích může být větvení, iterace a další běžné konstrukce. Veškeré proměnné jsou lokální, i v rámci hlavního těla programu. Soubory se zdrojovým kódem nelze slučovat a vytvářet tak jediný modul, který by bylo možné zkompilovat. **[[doplnit veci, ktere nejsou spolecne (nezabihat do detailu, vse bude specifikovano v podnadpisech)]]**

Deklarace a definice funkcí

Příkazy

- Přiřazení
- Větvení
- Cyklus while
- Cyklus for
- Volání funkcí

Výrazy

- Operátory
- Priorita operátorů
- Volání funkcí

6.2 Gramatický systém definujíci syntax jazyka Koubp

[[tady si nejsem uplne jisty - gramaticky system bych dal urcite do prilohy, ale rad bych zminil princip indexace + jak to vypada v ll tabulce a implementacne vyreseny deadlock.]]

Indexace neterminálů a význam pro implementaci

6.3 Návrh řešení syntaktického analyzátoru

Implementace syntaktické analýzy je silně objektově orientována, veškeré datové struktury jsou reprezentovány třídami. Třídy reprezentující neterminály a terminály mají společnou nadtřídu, díky čemuž je možné je ukládat do jednoho zásobníku za pomoci šablon. Jednu nadtřídu mají také gramatiky, jednotlivé instance jsou poté konstruovány pomocí tovární metody. Analyzátory jsou také reprezentovány třídami, společnou nadtřídu ale nemají.

Práce s gramatikami

Gramatiky jsou rozděleny tak, aby každá z nich tvořila ucelenou část jazyka Koubp. Mimo čtvrtou gramatiku jsou všechny využívány pouze pro prediktivní analýzu. Ačkoliv téma čtvrté gramatiky je analýza výrazů, je využívána jak precedenčním, tak prediktivním analyzátorem. Důvod je zvolený postup prediktivní analýzy volání funkcí – o samotné výrazy bez volání funkcí, tedy pouze s konstantami, proměnnými a podobně, se postará analýza precedenční.

Obrázek 6.1: **[[Tady bude obrázek znázorňující oba dva parsery a šest gramatik a který s čím pracuje.]]**

Předávání řízení prediktivní a precedenční analýzy

Při nutnosti předání řízení druhému analyzátoru je na místě vytvořena nová instance a okamžité zavolání metody `Parse()`.

Jedním z nich je neterminál `Expression` na vrcholu zásobníku pro prediktivní analýzu. V tomto případě je zřejmé, že musí být předáno řízení precedenční analýze, aby mohla být provedena analýza výrazu. Pokud precedenční analýza narazí na token `tFuncName`, reprezentující počátek volání funkce, musí opět předat řízení zpět prediktivní analýze.

[[presny popis zatim nemohu poskytnout, je to posledni vec, ktera mi chybi k dokonceni SA. nicmene bude doplnen.]]

Obrázek 6.2: **[[tady bude vytazek z kodu pro prepinani analyzatoru.]]**

Obrázek 6.3: **[[Tady bude obrazek, kde bude napsany jednoduchy kod volani funkce, jeho reprezentace v tokenech a navic vlozene pomocne tokeny, ktere parsery vyuzivaji. Bude tam zobrazene, v jakych mistech se provadi zmena analyz.]]**

6.4 Lexikální analýza a nástroj Flex

Pro usnadnění práce byl lexikální analyzátor automaticky vygenerován nástrojem *Flex*[1] ze souboru s lexémy popsanými regulárními výrazy. S každým úspěšně analyzovaným lexémem následuje vložení tokenu do vstupní pásky pro syntaktickou analýzu. Znamená to tedy, že se nejdříve v celém zdrojovém souboru ověří lexikální správnost, až poté správnost syntaktická. Při konstrukci tokenu se inicializuje i jeho datová část, do kterého patří datový typ a samotná data. Data jsou inicializována i pro tokeny, které žádná data v programu nereprezentují –

6.5 Abstraktní syntaktický strom

[1] Manuál k programu k dispozici na https://westes.github.io/flex/manual/.

Kapitola 7

Testování

Kapitola 8

Závěr

Literatura

[1] International Organization for Standardization (ISO). *ISO/IEC 9899:2018, Programming languages - C*. International Organization for Standardization (ISO), 2018. Dostupné z: https://www.open-std.org/jtc1/sc22/wg14/www/docs/n2310.pdf.

[2] Jiří Techet, T. M. a Meduna, A. *Cooperating Distributed Grammar Systems* [online]. 2007. Ústav Informačních Systémů, FIT VUT v Brně. Dostupné z: http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:09-cdgspres.pdf.

[3] Jiří Techet, T. M. a Meduna, A. *Parallel Communicating Grammar Systems* [online]. 2007. Ústav Informačních Systémů, FIT VUT v Brně. Dostupné z: http://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:tid:frvs:10-pcgspres.pdf.

[4] Kari, L. a Lutfiyya, H. CHAPTER 2 PARALLEL COMMUNICATING GRAMMAR SYSTEMS Bringing PC Grammar Systems Closer to Hoare ' s CSP ' s 1. In:. 2011. Dostupné z: https://api.semanticscholar.org/CorpusID:2392372.

[5] Kožár, T. a Meduna, A. *Automata: Theory, Trends, And Applications*. World Scientific Publishing Co Pte Ltd, 2023. 1–418 s. ISBN 978-981-1278-12-9. Dostupné z: https://www.fit.vut.cz/research/publication/13114.

[6] Meduna, A. *Deep Pushdown Automata* [online]. 2007. Ústav Informačních Systémů, FIT VUT v Brně. Dostupné z: https://www.fit.vutbr.cz/~meduna/work/lib/exe/fetch.php?media=lectures:phd:rgd:slides:deep_pda.pdf.

[7] Meduna, A. a Lukáš, R. *Modely bezkontextových jazyků* [online]. 2017. Ústav Informačních Systémů, FIT VUT v Brně. Dostupné z: https://www.fit.vutbr.cz/study/courses/IFJ/private/prednesy/Ifj08-cz.pdf.

[8] Meduna, A. a Zemek, P. *Regulated Grammars and Automata*. 1. vyd. New York, NY: Springer Science+Business Media, 2014. 694 s. ISBN 978-1-4939-0368-9, 978-1-4939-0369-6, 978-1-4939-4316-6.

[9] Rozenberg, G. a Salomaa, A., ed. *Handbook of Formal Languages: Linear Modeling: Background and Application*. 1. vyd. Berlin Heidelberg: Springer, 1997. ISBN 978-3-642-08230-6.

[10] Vaszil, G. Various communications in PC grammar systems. *Acta Cybernetica*. 1. vyd. Szeged, HUN: Institute of Informatics, University of Szeged. květen 1998, sv. 13, č. 2, s. 173–196. ISSN 0324-721X.

[11] Češka, M., Vojnar, T., Rogalewicz, A. a Smrčka, A. *Teoretická informatika: Studijní opora* [online]. Srpen 2020. Ústav Inteligentních Systémů, FIT VUT v Brně. Dostupné z: https://www.fit.vutbr.cz/study/courses/TIN/public/Texty/TIN-studijni-text.pdf.