Algoritmy syntaktické analýzy (pomocí CFG)

Vladimír Kadlec, Aleš Horák

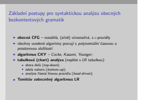
E-mail: `hales@fi.muni.cz`
`http://nlp.fi.muni.cz/poc_lingv/`

Obsah:

Základní postupy pro syntaktickou analýzu obecných bezkontextových gramatik

- **obecná CFG** – rozsáhlá, (silně) víceznačná, s ϵ-pravidly
- všechny uvedené algoritmy pracují s *polynomiální časovou a prostorovou složitostí*
- **algoritmus CKY** – *Cocke*, *Kasami*, *Younger*;
- **tabulková (chart) analýza** (neplést s LR tabulkou):
  - shora dolů (*top-down*);
  - zdola nahoru (*bottom-up*);
  - analýza řízená hlavou pravidla (*head-driven*);
- **Tomitův zobecněný algoritmus LR**

Základní postupy pro syntaktickou analýzu obecných bezkontextových gramatik



tabulková (chart) analýza – neplést s LR tabulkou → u chartu se žádná tabulka dopředu nepředpočítává
Tomitův zobecněný algoritmus LR - zobecňuje LR(0) analýzu tak, že se dá použít pro obecnou CFG.
A Grammar is said to be LR(1) if there is no conflict present in any of its LR(1) canonical sets I.e. if no state prompts the parser to perform more than one action on some input symbol.
LA LR – Look Ahead LR grammar.

Syntaktická analýza

- Vstupy:
  - řetězec lexikálních kategorií (preterminálních symbolů) $a_1\ a_2\ \dots\ a_n$

    např.: ADJ CONJ ADJ N V PREP N '.'
  - bezkontextová gramatika $G = \langle N, \Sigma, P, S \rangle$.
- Výstup:
  - efektivní reprezentace derivačních stromů.

Algoritmus CKY

- Gramatika musí být v Chomského normální formě.

  CNF (každá CFG jde do ní převést): $A \to BC$
  $D \to$ 'd'

- Pro daný vstup délky n derivujeme podřetězce symbolů délky q na pozici p, značíme $w_{p,q}$, $1 \le p, q \le n$.
- Derivace řetězců délky 1, $A \Rightarrow w_{p,1}$, je prováděno prohledáváním terminálních pravidel.
- Derivace delších řetězců $A \Rightarrow^* w_{p,q}$, $q \ge 2$ vyžaduje aby platilo $A \Rightarrow BC \Rightarrow^* w_{p,q}$. Tedy z B derivujeme řetězec délky k, $1 \le k \le q$, a z C derivujeme zbytek, řetězec délky $q - k$. Tzn. $B \Rightarrow^* w_{p,k}$ a $C \Rightarrow^* w_{p+k,q-k}$. Kratší řetězce máme tedy vždy "předpočítané."

CNF – jeste navic moznost pravidla "kořenový neterminál $\to \epsilon$," ale takové pravidlo v našich gramatikách není.

Algoritmus CKY pokrač.

program CKY Parser;
begin
  for $p := 1$ **to** n **do** $V[p,1] := \{A | A \to a_p \in P\}$;
  for $q := 2$ **to** n **do**
    for $p := 1$ **to** $n - q + 1$ **do**
      $V[p,q] = \emptyset$;
      for $k := 1$ **to** $q - 1$ **do**
        $V[p,q] =$
          $V[p,q] \cup$
          $\cup \{A | A \to BC \in P, B \in V[p,k], C \in V[p+k, q-k]\}$
      od
    od
  od
end

složitost CKY je $O(n^3)$

počítá se (trojúhelníková) matice V:

- sloupce = pozice ve vstupní větě
- řádky = délky (pod)řetězců vstupní věty
- prvky = množiny neterminálů, které pokrývají odpovídající část vstupní vět

první cyklus naplní první řádek matice
ve vnitřním cyklu se B a C vybírají vždy z už hotových políček matice (menší řetězce) – tj. od 2. řádku už vůbec nepracujeme se vstupní větou, jen s předchozími řádky
neznámé terminály na vstupu se *ignorují*

Algoritmus CKY, příklad – zadání

- vstupní gramatika je:

$S \rightarrow AA|BB|AX|BY|a|b$
$X \rightarrow SA$
$Y \rightarrow SB$
$A \rightarrow a$
$B \rightarrow b$

- vstupní řetězec je $w = abaaba$.

Algoritmus CKY, příklad – řešení (matice V)

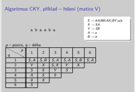
a b a a b a

$S \rightarrow AA|BB|AX|BY|a|b$
$X \rightarrow SA$
$Y \rightarrow SB$
$A \rightarrow a$
$B \rightarrow b$

p – pozice, q – délka

q \ p	1	2	3	4	5	6
1	S, A	S, B	S, A	S, A	S, B	S, A
2	Y	X	S, X	Y	X	
3	S	$\emptyset$	Y	S		
4	X	S	$\emptyset$			
5	$\emptyset$	X				
6	S					

Algoritmus CKY, příklad – řešení (matice V)



nechat počítat na tabuli studenty – políčka v prvních řádcích jdou rychleji napsat na tabuli prázdnou matici V a do ní doplňovat.
postup: např. 2. řádek, políčko [1,2] vzniká z [1,1] a [2,1] – 4 kombinace $SS, SB, AS, AB \rightarrow$ v gramatice je jen SB, tj. Y. kombinace se vždycky počítají ze dvou políček, které se pohybují ve "véčku" nad počítaným polem.
$\emptyset$ v políčku znamená, že příslušný podřetězec nejde vygenerovat z žádného pravidla gramatiky.
složitost CKY je vždycky $O(n^3)$ na rozdíl od ostatních, kde je jen $\Omega(n^3)$
výsledek = *true*/*false* podle toho, jestli je v políčku dole kořen. pro generování stromů z CKY tabulky bychom si museli pamatovat v každém políčku, z jakých políček vznikl který neterminál.

Tabulkové (chart) analyzátory

- Rozlišujeme tři základní typy **tabulkových analyzátorů**:
  - shora dolů;
  - zdola nahoru;
  - analýza řízená hlavou pravidla.
- Mnoho dalších variant je popsáno v:
  Sikkel Klaas: **Parsing Schemata: A Framework for Specification and Analysis of Parsing Algorithm**, 1997.
- Neklade se žádné omezení na gramatiku.
- Analyzátory typu "chart" v sobě většinou obsahují dvě datové struktury **chart** a **agendu**. Chart a agenda obsahují tzv. *hrany*.
- **Hrana** je trojice $[A \rightarrow \alpha_\bullet\beta,\ i,\ j]$, kde:
  - i, j jsou celá čísla, $0 \leq i \leq j \leq n$ pro n slov ve vstupní větě
  - a $A \rightarrow \alpha\beta$ je pravidlem vstupní gramatiky.

$[A \rightarrow BC \bullet DE, 0, 3]$

$_0$ a $_1$ b $_2$ a $_3$ a $_4$ b $_5$ a $_6$

Obecný analyzátor typu "chart"

```
program Chart Parser;
begin
    inicializuj (CHART);
    inicializuj (AGENDA);
    while (AGENDA není prázdná) do
        E := vezmi hranu z AGENDA;
        for each (hrana F, která může být vytvořena pomocí
          hrany E a nějaké jiné hrany z CHART) do
          if ((F není v AGENDA) and (F není v CHART) and
             (F je různá od E)
             then přidej F do AGENDA;
          fi;
        od;
        přidej E do CHART;
    od;
end;
```


tato struktura programu je společná všem typům chart parserů. ty se navzájem liší v:

1. jak se *inicializuje*
2. jak se vybírá F

dá se to udělat i jinak (bez agendy), ale tato metoda je nejčastější proč se nezacyklí:

1. hran je konečný počet
2. každou hranu projde maximálně jednou

Varianta shora dolů

Inicializace:

- $\forall\ p \in P \mid p = S \rightarrow \alpha$ přidej hranu $[S \rightarrow {}_{\bullet}\alpha,\ 0,\ 0]$ do agendy.
- počáteční chart je prázdný.

Iterace – vezmi hranu E z agendy a pak:

a) (*fundamentální pravidlo*) pokud je E ve tvaru $[A \rightarrow \alpha_{\bullet},\ j,\ k]$, potom pro každou hranu $[B \rightarrow \gamma_{\bullet}\ A\ \beta,\ i,\ j]$ v chartu vytvoř hranu $[B \rightarrow \gamma\ A\ {}_{\bullet}\beta,\ i,\ k]$.

b) (*uzavřené hrany*) pokud je E ve tvaru $[B \rightarrow \gamma_{\bullet}\ A\ \beta,\ i,\ j]$, potom pro každou hranu $[A \rightarrow \alpha_{\bullet},\ j,\ k]$ v chartu vytvoř hranu $[B \rightarrow\ \gamma\ A\ {}_{\bullet}\beta,\ i,\ k]$.

c) (*terminál na vstupu*) pokud je E ve tvaru $[A \rightarrow \alpha_{\bullet} a_{j+1} \beta,\ i,\ j]$, vytvoř hranu $[A \rightarrow\ \alpha\ a_{j+1\bullet}\beta,\ i,\ j{+}1]$.

d) (*predikce*) pokud je E ve tvaru $[A \rightarrow \alpha_{\bullet}\ B\ \beta,\ i,\ j]$ potom pro každé pravidlo $B \rightarrow \gamma \in P$, vytvoř hranu $[B \rightarrow {}_{\bullet}\ \gamma,\ i,\ i]$.

Příklad – tabulkové analýzy (typu chart)

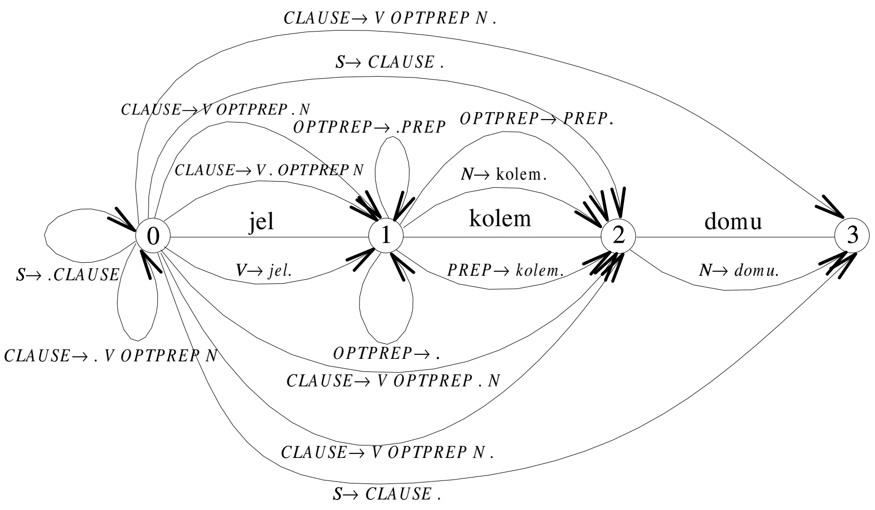
Gramatika:

$$
\begin{array}{rcl}
S & \rightarrow & CLAUSE \\
CLAUSE & \rightarrow & V\ OPTPREP\ N \\
OPTPREP & \rightarrow & \epsilon \\
OPTPREP & \rightarrow & PREP \\
V & \rightarrow & jel \\
PREP & \rightarrow & kolem \\
N & \rightarrow & domu \\
N & \rightarrow & kolem
\end{array}
$$

Věta:

"jel kolem domu" (a_1=*jel*, a_2=*kolem*, a_3=*domu*).

Příklad – chart po analýze shora dolů



1. inicializace
2. predikce – aplikace d)
3. predikce – aplikace d)
4. terminal – aplikace c)
5. uzavrena hrana – aplikace a)
6. ...
7. v posledním – vynechané ϵ-hrany (nevešly by se)

složitost – počet pravidel bereme jako konstantu $\rightarrow$ pak máme podle délky vstupu n celkem n^2 možných hran a v každém kroku zpracujem až n hran $\rightarrow O(n^3)$.

Varianta zdola nahoru

Inicializace:

- $\forall\ p \in P \mid p = A\rightarrow \epsilon$ přidej hrany $[A\rightarrow \bullet$, 0, 0], $[A\rightarrow \bullet$, 1, 1], ..., $[A\rightarrow \bullet$, n, n] do agendy.
- $\forall\ p \in P \mid p = A\rightarrow a_i\alpha$ přidej hranu $[A\rightarrow {\bullet}a_i\alpha$, i-1, i-1] do agendy.
- počáteční chart je prázdný.

Iterace – vezmi hranu E z agendy a pak:

a) (*fundamentální pravidlo*) pokud je E ve tvaru $[A\rightarrow \alpha_\bullet, j, k]$, potom pro každou hranu $[B\rightarrow \gamma_\bullet\ A\ \beta, i, j]$ v chartu vytvoř hranu $[B\rightarrow \gamma\ A\ {\bullet}\beta, i, k]$.

b) (*uzavřené hrany*) pokud je E ve tvaru $[B\rightarrow \gamma_\bullet\ A\ \beta, i, j]$, potom pro každou hranu $[A\rightarrow \alpha_\bullet, j, k]$ v chartu vytvoř hranu $[B\rightarrow\ \gamma\ A\ {\bullet}\beta, i, k]$.

c) (*terminál na vstupu*) pokud je E ve tvaru $[A\rightarrow \alpha_\bullet a_{j+1}\beta, i, j]$, potom vytvoř hranu $[A\rightarrow\ \alpha\ a_{j+1\bullet}\beta, i, j+1]$.

d) (*predikce*) pokud je E ve tvaru $[A\rightarrow \alpha_\bullet, i, j]$, potom pro každé pravidlo $B{\rightarrow}A\gamma$ vstupní gramatiky vytvoř hranu $[B\rightarrow {\bullet}A\gamma, i, i]$.

a), b) a c) jsou stejné jako u shora dolů, liší se jen v d).
většinou vytváří víc nadbytečných hran.

Analýza řízená hlavou pravidla

- *head-driven chart parsing*
- **Hlava pravidla** – libovolný (určený) symbol z pravé strany pravidla.

  Například pravidlo *CLAUSE* → *V* *<u>OPTPREP</u>* *N* může mít hlavy *V*, *OPTPREP*, *N*.
- Epsilon pravidlo má hlavu ϵ.
- Hrana v analyzátoru řízeném hlavou pravidla – trojice $[A \rightarrow \alpha_\bullet \beta_\bullet \gamma,\ i,\ j]$, kde *i, j* jsou celá čísla, $0 \leq i \leq j \leq n$ pro *n* slov ve vstupní větě a $A \rightarrow \alpha\beta\gamma$ je pravidlo vstupní gramatiky a hlava je v β.
- Algoritmus vlastní analýzy (varianta zdola nahoru) je podobný jednoduchému přístupu. Analýza neprobíhá zleva doprava, ale začíná na hlavě daného pravidla.

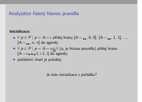
Analyzátor řízený hlavou pravidla

Inicializace:

- $\forall\ p \in P \mid p = A \rightarrow \epsilon$ přidej hrany $[A \rightarrow \bullet\bullet,\ 0,\ 0]$, $[A \rightarrow \bullet\bullet,\ 1,\ 1]$, ..., $[A \rightarrow \bullet\bullet,\ n,\ n]$ do agendy.
- $\forall\ p \in P \mid p = A \rightarrow \alpha \underline{a_i} \beta$ (a_i je hlavou pravidla) přidej hranu $[A \rightarrow \alpha_\bullet a_{i\bullet} \beta,\ i\text{-}1,\ i]$ do agendy.
- počáteční chart je prázdný.

Je tato inicializace v pořádku?

Analyzátor řízený hlavou pravidla



Možný problém: *co když inicializace nic nepřidá?* (žádné ϵ ani žádný terminál jako hlava)

Odpověď: taková gramatika by generovala prázdný jazyk.

Analyzátor řízený hlavou pravidla pokrač.

Iterace – vezmi hranu *E* z agendy a pak:

a_1) pokud je E ve tvaru $[A \rightarrow {}_\bullet\alpha_\bullet,\ j,\ k]$, potom pro každou hranu: $[B \rightarrow \beta_\bullet \gamma_\bullet A\delta,\ i,\ j]$ v chartu vytvoř hranu $[B \rightarrow \beta_\bullet \gamma A_\bullet \delta,\ i,\ k]$.

a_2) $[B \rightarrow \beta A_\bullet \gamma_\bullet \delta,\ k,\ l]$ v chartu vytvoř hranu $[B \rightarrow \beta_\bullet A \gamma_\bullet \delta,\ j,\ l]$.

b_1) pokud je E ve tvaru $[B \rightarrow \beta_\bullet \gamma_\bullet A\delta,\ i,\ j]$, potom pro každou hranu $[A \rightarrow {}_\bullet\alpha_\bullet,\ j,\ k]$ v chartu vytvoř hranu $[B \rightarrow \beta_\bullet \gamma A_\bullet \delta,\ i,\ k]$.

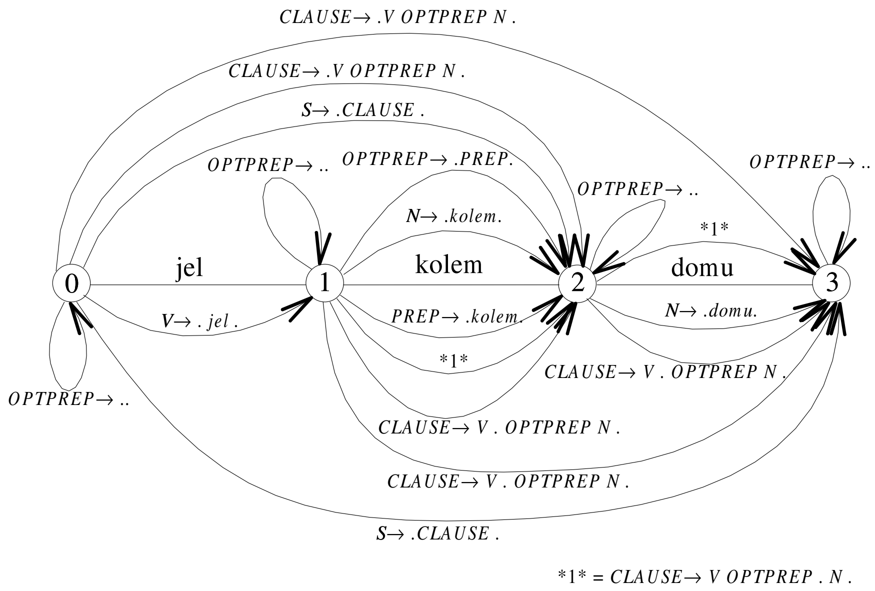
b_2) pokud je E ve tvaru $[B \rightarrow \beta A_\bullet \gamma_\bullet \delta,\ k,\ l]$, potom pro každou hranu $[A \rightarrow {}_\bullet\alpha_\bullet,\ j,\ k]$ v chartu vytvoř hranu $[B \rightarrow \beta_\bullet A \gamma_\bullet \delta,\ j,\ l]$.

c_1) pokud je E ve tvaru $[A \rightarrow \beta a_{i\bullet} \gamma_\bullet \delta,\ i,\ j]$, potom vytvoř hranu $[A \rightarrow \beta_\bullet a_i \gamma_\bullet \delta,\ i\text{-}1,\ j]$.

c_2) pokud je E ve tvaru $[A \rightarrow \beta_\bullet \gamma_\bullet a_{j+1} \delta,\ i,\ j]$, potom vytvoř hranu $[A \rightarrow \beta_\bullet \gamma a_{j+1\bullet} \delta,\ i,\ j{+}1]$.

d) pokud je E ve tvaru $[A \rightarrow {}_\bullet\alpha_\bullet,\ i,\ j]$, potom pro každé pravidlo $B \rightarrow \beta\ \underline{A}\ \gamma$ ve vstupní gramatice vytvoř hranu $[B \rightarrow \beta_\bullet A_\bullet \gamma,\ i,\ j]$ (symbol *A* je hlavou pravidla).

Příklad – chart po analýze řízené hlavou pravidla



DEMO NLTK – `chart`

`cd ~/text/poc_lingvistika/nltk/`

`./ukazky.sh` → `chart`

Tomitův zobecněný analyzátor LR

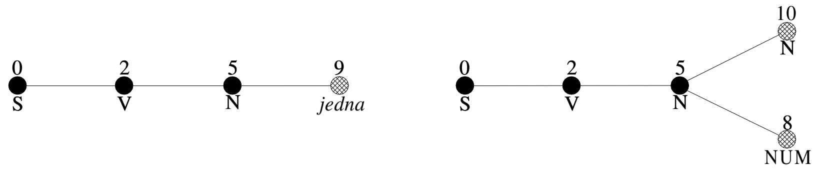
- *generalized LR parser (GLR)*
- Masaru Tomita: Efficient parsing for natural language, 1986
- standardní LR tabulka, která může obsahovat konflikty;
- zásobník je reprezentován acyklickým orientovaným grafem (*DAG*);
- derivační stromy jsou uloženy ve sbaleném "lese" stromů.
- v podstatě stejný, jako algoritmus LR;
- udržujeme si seznam aktivních uzlů zásobníku (grafu);
- akce redukce provádíme vždy před akcemi čtení;
- akci čtení provádíme pro všechny aktivní uzly najednou;
- kde je to možné, tam uzly slučujeme.

standardní LR tabulka, která může obsahovat konflikty – to normálně nemůže
algoritmus s sebou nese všechny nevýhody LR – předpočítávání (velké) tabulky – časově je pomalejší než chart analýza. je vhodný pro skorojednoznačné gramatiky, pro které jde líp použít předpočítaná LR tabulka.
akce redukce provádíme vždy před akcemi čtení – tohle je důležité, aby nenarůstal graf donekonečna – provádíme tady to spojování. počet uzlů v zásobníku je tak polynomiální.
DEMO NLTK – `srparser`

`cd ~/text/poc_lingvistika/nltk/`

`./ukazky.sh` → `srparser`

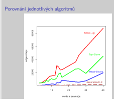
Příklad konfliktu redukce/redukce



stav	položka	akce	symbol	další stav
5	*CLAUSE*→*V N*•*NUM*	shift	*NUM*	8
	NN→ *N*•*N*		*N*	10
	NUM→ •*jedna*		*jedna*	9
	N→ •*tramvaj*		*tramvaj*	7
	N→ •*jedna*			
9	*NUM*→*jedna*•	reduce (6)		
	N→*jedna*•	reduce (5)		

Porovnání jednotlivých algoritmů

Porovnání jednotlivých algoritmů



měřeno na 10.000 větách z korpusu.

tady porovnáváme počty hran a uzlů GLR zásobníku, takže GLR nelze s chartem takto přesně srovnat. skutečný čas GLR leží někde mezi top-down a bottom-up.

u head-driven hodně záleží na volbě hlav.

Porovnání jednotlivých algoritmů