CS344 - CYK Algorithm + Parsing

Note Title 2/3/2012

Announcements

- HW2 due now
- HW3 is up
  - ↳ feel free to use the web

    due on Sunday

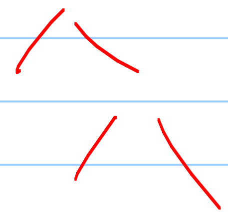
Recap: CFGs and CNF

Chomsky Normal Form:

Each rule in the grammar is either:

- $A \rightarrow BC$ 2 nonterminals
  where neither B or C is the start variable, & both are nonterminals
- $A \rightarrow a$
  where a is a terminal
- No useless symbols

Why do we care?



1. Makes structure of parsing nice.
2. Computable: $O(n^2)$ time to convert to CNF
3. Given CNF, can compute a parse tree in $O(n^3)$ time (using dynamic programming).

Ex: Simple CFG

$S_0 \rightarrow AB$

$A \rightarrow aA \mid \varepsilon$

$B \rightarrow bAbAB \mid \varepsilon$

tokens or terminals

$a^*(ba^*ba^*)^*$

Convert:

① useless states?

No

② Eliminate ε- transitions

$$S_0 \to AB$$

$$A \to aA \mid \cancel{\varepsilon}$$

$$B \to bAbAB \mid \cancel{\varepsilon}$$

$$S_0 \to AB \mid A \mid B \mid \varepsilon$$

$$A \to aA \mid a$$

$$B \to bAbAB \mid bAbA \mid bb \mid bbB$$
$$\mid bAb \mid bbA \mid bbAB \mid bAbB$$

$X \to Y$

$X \to Y \to Z$

3 Eliminate unit pairs : (S, A) + (S, B)

$S_0 \to AB \mid A \mid B \mid \varepsilon$

$A \to aA \mid a$

$B \to bAbAB \mid bAbA \mid bb \mid bbB$
$\quad bAb \mid bbA \mid bbAB \mid bAbB$

$S_0 \to AB \mid aA \mid a$
$\quad bAbAB \mid bAbA \mid bb \mid bbB$
$\quad bAb \mid bbA \mid bbAB \mid bAbB \mid \varepsilon$

$A \to aA \mid a$

$B \to bAbAB \mid bAbA \mid bb \mid bbB$
$\quad bAb \mid bbA \mid bbAB \mid bAbB$

4) a: Get rid of non-single terminals
(eg $X \to yz$ becomes $X \to YZ$ + $Y \to y$)

$S_0 \to AB \mid aA \mid a$
$\mid bAbAB \mid bAbA \mid bb \mid bbB$
$\mid bAb \mid bbA \mid bbAB \mid bAbB \mid \varepsilon$

$A \to XA \mid a$
$B \to YAYAB \mid YAYA \mid bb \mid bbB$
$\mid bAb \mid bbA \mid bbAB \mid bAbB$

$X \to a$
$Y \to b$

(replace any a with X
+ any ~~b~~ b with Y)

4b: Eliminate 3 or more non-terminal transitions

$B \to YAYAB$ (too long)

$$\begin{cases} B \to Y M_1 \\ M_1 \to A M_2 \\ M_2 \to Y M_3 \\ M_3 \to AB \end{cases}$$

$B \to M_1 M_2$

$M_1 \to YA$

$M_2 \to M_1 B$

Now, parsing. Consider abbaabab.

In language?

How to parse:

$$S_0 \longrightarrow AB$$

$$A \longrightarrow aA \mid \varepsilon$$

$$B \longrightarrow bAbAB \mid \varepsilon$$

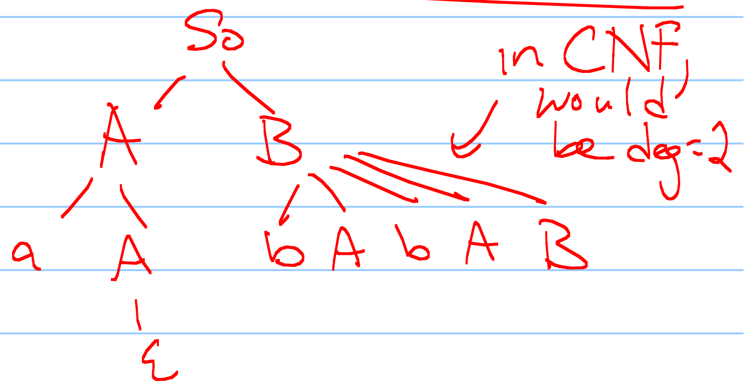
tokens

1 $S_0 \to AB \xrightarrow{(\text{do } A)} aAB$

$\to aB$

$\to abAbAB$

$\to abbAB$



Parse tree: (prev page)

CYK algorithm: build a table

Given a word $w = w_1 w_2 w_3 w_4 \cdots w_k$, we'll look at all possible substrings $w_i w_{i+1} \cdots w_{j-1} w_j$, and look at how they can be parsed.

We'll build a table from the bottom up.

Ex:

$$
\begin{aligned}
S &\to AB \mid BC \\
A &\to BA \mid a \\
B &\to CC \mid b \\
C &\to AB \mid a
\end{aligned}
\quad \Bigg\} \quad X \to x
$$

complexity n^3

Compute valid parse tree for 'baaba'

length 5					
length 4	–	–	{S,C,A}		
length 3	–	{B}	{B}		
length 2	{A,S}	{B}	{S,C}	{A,S}	
length 1	{B}	{A,C}	{A,C}	{B}	{A,C}
	b	a	a	b	a

Running times:

Say we have n rules.

Converting to CNF: $O(n^2)$

finding unit pairs

Running CYK: $O(n^3)$

Other parsing algorithms

CYK is still pretty slow, especially for large programming languages.

After it was developed, a lot of work was put into figuring out what grammars could have faster algorithms.

Two big (& useful) classes have linear time parsers: LL & LR.