

CS344 - CNF & CYK

Note Title

2/1/2012

Announcements

- HW 2, due Friday
(new version / posted)
- HW 3 is up

Context Free Languages - CFLs

Described in terms of productions
(Called Backus-Naur Form, or BNF)

- A set of terminals T
- A set of non-terminals N
- A start symbol S (a non-terminal)
- A set of productions

Ex:

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow \underline{b} \mid \epsilon$

terminal

non terminals

Chomsky Normal Forms (CNF)

Each rule in the grammar is either:

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

Why CNF?

Parsing: building those parse trees
we saw

In general, there are an exponential
number of parse trees
for a given input.

Given CNF, can get a polynomial
time algorithm to generate a
valid parse tree.

Thm: All grammars can be converted to CNF.

Procedure:

First eliminate useless rules.

- Start from the start state, & expand set of "reachable states"

- start from terminating rules & work backwards

① Introduce "dummy" start state

Ex: $S \rightarrow ASA \mid aB$

$\{S, A, B\}$ ✓

$A \rightarrow B \mid S$

$\{B, A, S\}$



$\underline{B} \rightarrow \underline{b} \mid \epsilon$

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \epsilon$

② Nullable variables: $B \rightarrow \epsilon$
(only allowed for $B = \text{start state in CNF}$)

Remove all ϵ productions:

$$S_0 \rightarrow S$$

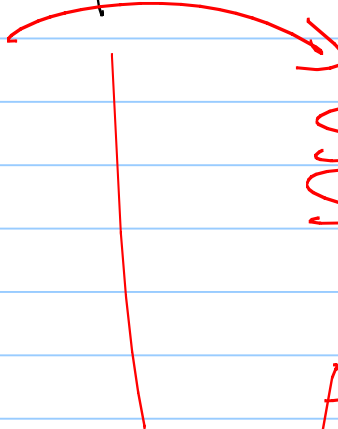
$$S \rightarrow ASA \mid aB$$

$$A \rightarrow B$$

$$A \rightarrow S$$

$$B \rightarrow b$$

~~$$B \rightarrow \epsilon$$~~


$$S_0 \rightarrow S$$
$$S \rightarrow ASA \mid aB \mid a$$
$$S \mid AS \mid SA \mid a$$

$$A \rightarrow B$$

$$A \rightarrow S$$

$$B \rightarrow b$$

③ Remove unit rules:

$$\underline{A \rightarrow B}$$

One idea: if $A \rightarrow B$ and $B \rightarrow w$,
remove $A \rightarrow B$ + replace
with $A \rightarrow w$

Will work, but 1 problem:

$$\begin{array}{l} A \rightarrow B \\ B \rightarrow C \mid b \\ C \rightarrow A \mid c \end{array}$$

Detecting unit pairs:

How? Must have:

$$X \rightarrow z_1, z_1 \rightarrow z_2, \dots, z_k \rightarrow Y$$

(since we removed ϵ -transitions in (2))

Compute Unit Pairs (Rules in CFG)

NewRules $\leftarrow \{ (x \rightarrow Y) \text{ in rules} \}$

do:

OldRules \leftarrow NewRules

for $(x \rightarrow Y)$ in NewRules

for $(Y \rightarrow z)$ in NewRules

NewRules \leftarrow NewRules $\cup (x \rightarrow z)$

while (NewRules \neq OldRules)

$O(n^2)$

$$\underline{A \rightarrow BC}$$

Now remove all unit rules $A \rightarrow B$.

For any unit pair (X, Y) & rule $Y \rightarrow w$,
add $X \rightarrow w$ to the transitions

Unit Pair: $A \Rightarrow B$
only since (non terminal) replacements:

$$\underline{A} \rightarrow X_1, X_1 \rightarrow X_2, X_2 \rightarrow X_3, \dots, X_k \rightarrow \underline{B}$$

↑
non terminals

Example: ~~$S_0 \rightarrow S$~~
 $S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS \leftarrow$
 $\rightarrow A \rightarrow \cancel{B} \mid \cancel{S}$
 $B \rightarrow \underline{b}$

Unit Pairs: $(S'_0 \rightarrow S)$ $(A' \rightarrow B)$ $(A' \rightarrow S)$

$S_0 \rightarrow \overbrace{ASA \mid aB} \mid a \mid SA \mid AS$
 $S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$
 $A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$
 $B \rightarrow b$

④ Get rid of "long" righthand sides.

Recall goal of CNF:

$A \rightarrow BC$ ← nonterminals

Not OK: $A \rightarrow BCD$ ←

$A \rightarrow bB$ ←

4a: Create $V_c \rightarrow c$ for every character.

Replace c with V_c everywhere.

Now all rules are either:

$A \rightarrow CDEF$

or

$V_c \rightarrow c$.

4b: $A \rightarrow B_1 B_2 B_3 \dots B_k$

How to replace with only 2 nonterminals on the right?
right?

$$A \rightarrow B_1 X_1$$

$$X_1 \rightarrow B_2 X_2$$

$$X_2 \rightarrow B_3 X_3$$

$$X_{k-1} \rightarrow B_{k-1} B_k$$

Ex:

$S_0 \rightarrow A \cancel{SA}^z | UB | a | SA | AS$
 $S_1 \rightarrow A \cancel{SA}^z | UB | a | SA | AS$
 $A \rightarrow b | A \cancel{SA}^z | UB | a | SA | AS$
 $B \rightarrow b$
 $U \rightarrow a$

$z \rightarrow SA$

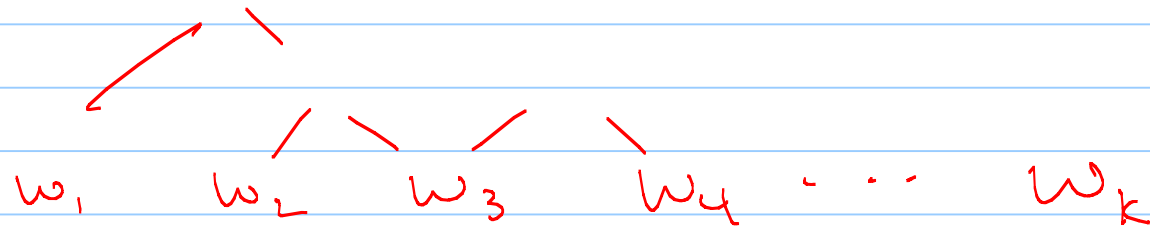
Done!

Why CNF?

In general, there are an exponential number of parse trees for a given input.

So how to check quickly?

Even in CNF might be 2^n possible parse trees:



Solution: Dynamic Programming

CYK Algorithm (Cocke-Younger-Kasami '65, '67)

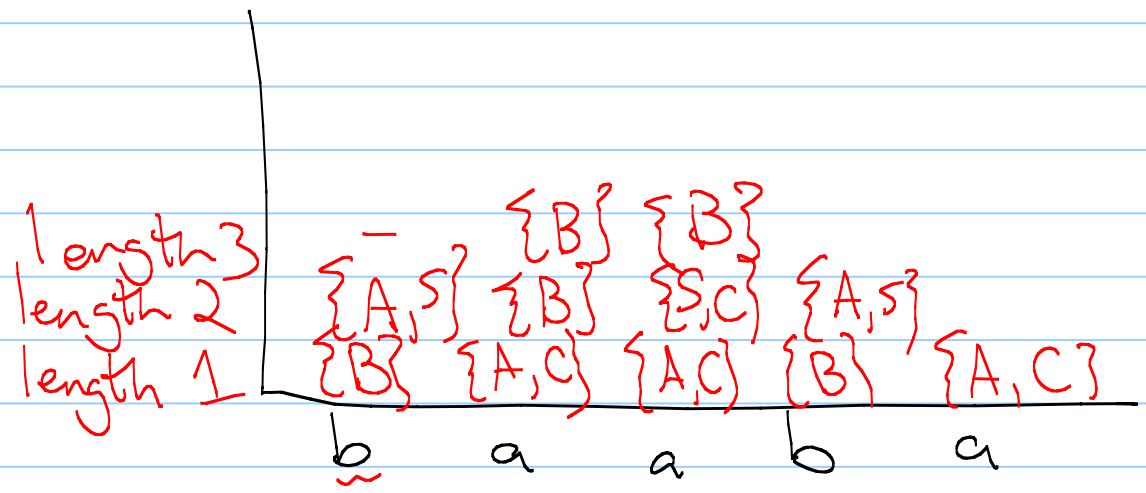
Given CNF grammar

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

We'll build a table from the bottom up.

Ex: $S \rightarrow AB \mid BC$
 $A \rightarrow BA \mid a$
 $B \rightarrow CC \mid b$
 $C \rightarrow AB \mid a$

Test if 'baaba' is in the language



Ex (cont)

Running times:

Say we have n rules.

Converting to CNF:

Running CYK: