

Today: EBNF $\rightarrow$ standard
 Bottom-up is top-down
 parsing

$A \rightarrow R$
 R is regexp over
 NUT

Ex:
 $E \rightarrow (T(''|')^*)T$
 $E \Rightarrow T \circ T \circ T \circ \dots \circ T$
 $T \rightarrow (F(''|')^*)F$
 $F \rightarrow c(''|')^*E'$

EBNF $\rightarrow$ 'Normal' grammar
 Recursive
 Replace
 $A \rightarrow R$
 (R is a regexp)
 into 1 or more productions
 of the form
 $A \rightarrow R'$
 R' is shorter (simpler) than R
 Apply local replacement repeatedly until
 normal grammar

Feb 14-11:02 AM

Base case:
 $A \rightarrow \emptyset$
 not production

Note: A parse tree for an
 EBNF grammar has internal
 nodes looking like

$A \rightarrow X_1 \dots X_k$ an
 grammar symbol
 where A is a nonterminal and
 $X_1 \dots X_k$ match some regexp R
 and $A \rightarrow R$ is a production.

$A \rightarrow \epsilon$ leave this alone
 $A \rightarrow X_1 X_2 \dots X_k$ ($k > 0$)
 where $X_1, \dots, X_k$ are grammar
 symbols: leave alone

$A \rightarrow R/S$ (R, S regexp)
 replace with 2 productions:
 $A \rightarrow R$
 $A \rightarrow S$

$A \rightarrow RS$ (R, S regexp)
 Introduce 2 new variables
 A_1, A_2 not appearing anywhere else
 Replace the prod above with:
 $A \rightarrow A_1 A_2$
 $A_1 \rightarrow R$
 $A_2 \rightarrow S$

$A \rightarrow R^*$ Replace with
 $A \rightarrow A' A$
 $A' \rightarrow \epsilon \mid R$
 End of transformation

Feb 14-11:11 AM

Top-down & bottom-up parsing

$S \rightarrow SS \mid (S) \mid [S] \mid \epsilon$

yield:
 $([()])()$ $\leftarrow$ parse only if
 it's right

Equivalent, combinatorial grammar
 $S \rightarrow (S)S \mid [S]S \mid \epsilon$

Bottom up: traverse children
 before parent

Top-down: visit parent & traverse
 children

labeled
 $([()])()$

LL(1) grammar: allow deterministic
 top-down parsing with no symbol of
 lookahead.

Expression grammar
 $E \rightarrow E + T \mid T$ not LL(1)
 $T \rightarrow T * F \mid F$
 $F \rightarrow c \mid (E)$

Feb 14-11:30 AM

$c * c + (c)$

The grammar
 is suitable for
 bottom-up parsing: LR(1)
 rightmost derive

bottom up:
 $c * c + (c)$

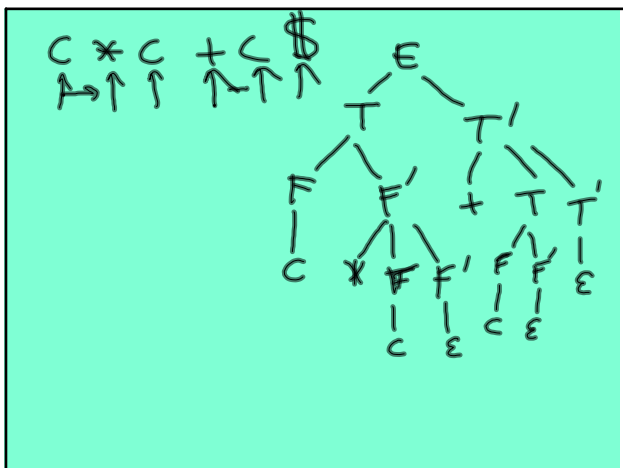
LL(1) grammars vs LR(1) grammars
 $E \rightarrow E + T \mid T$ LR(1)
 $T \rightarrow T * F \mid F$ not LL(1)
 $F \rightarrow c \mid (E)$

Left recursion:
 $A \rightarrow A\alpha \mid \beta$
 Left recursion
 prevents top-down parsing
 -good for bottom-up

LL(1) grammar for
 arithmetic exps

$E \rightarrow TT'$
 $T' \rightarrow E \mid +TT'$
 $T \rightarrow FT'$
 $F' \rightarrow E \mid *FF'$ $F \rightarrow c \mid (E)$

Feb 14-11:55 AM



Feb 14-12:11 PM