Context Free Grammars

$G = (V, \Sigma, R, S)$

- V: Finite Set of Variables (Non-Terminals)
- Σ: Finite Alphabet of Terminals
- R: Rules, each of Form
- S: Start Symbol $S \in V$

$$A \to \alpha \qquad A \in V,\ \alpha \in (V \cup \Sigma)^*$$

A-Rules can be listed on separate lines or using | to indicate alternatives, e.g.

$$S \to aSb \mid \lambda \qquad \{a^n b^n \mid n \geq 0\}$$

OR

$$S \to aSb$$
$$S \to \lambda$$

OR

$$S \to aSb$$
$$\mid \lambda$$

Some Example CFLs

$L_1 = \{a^n b^n \mid n \geq 0\}$

$G_1 = (\{S\}, \{a,b\}, R, S)$

R: $S \to aSb \mid \lambda$

$L_2 = \{a^n b^m \mid m \geq n\}$

$S \to aSb \mid Sb \mid \lambda$

$L_3 = \{a^n b^m \mid m > n\}$

$S \to aSb \mid Sb \mid b$

$L_4 = \{w w^R \mid w \in \{a,b\}^*\}$

$S \to aSa \mid bSb \mid \lambda$

NOTE $\{ww \mid w \in \{a,b\}^*\}$

IS NOT A CFL

$L_5 = \{w \mid \#a\text{'s in } w = \#b\text{'s in } w\}$

$S \to SaSbS \mid SbSaS \mid \lambda$

CAN OMIT

Proof That a Grammar Generates Desired Language

$L = \{ w \mid w \in \{a,b\}^* \text{ and } w_a = w_b \}$

Here w_a is # of a's in w
w_b is # of b's in w

$G = (\{S\}, \{a,b\}, R, S)$

R: $S \to aSbS \mid bSaS \mid \lambda$

Prove $\mathcal{L}(G) = L$

1. First $\mathcal{L}(G) \subseteq L$
   As all RHS of S have equal # of a's & b's
   $w \in \mathcal{L}(G) \Rightarrow w \in L$
2. Now show $L \subseteq \mathcal{L}(G)$
   Do by induction on $|w|$ to show
   $w \in L \Rightarrow w \in \mathcal{L}(G)$

INDUCTIVE PROOF

BASIS: $|w|=0$. SHOW $\lambda \in \mathcal{L}(G)$

AS $S \to \lambda \in R$ THEN $S \Rightarrow \lambda$ AND λ IS ONLY STRING OF LENGTH 0.

IH: FOR ALL $i \leq k$, $k \geq 0$,

IF $w \in L$, $|w| = 2i$ THEN $w \in \mathcal{L}(G)$

NOTE: ALL ELEMENTS OF L ARE OF EVEN LENGTH

IS: SHOW FOR $|w| = 2k+2$

ASSUME w STARTS WITH AN 'a'

THEN $w = a\alpha b\beta$

WHERE $\alpha \in L$, $\beta \in L$ AND α IS SHORTEST STRING LEADING TO MATCHING 'b' FOR FIRST 'a'

AS $|\alpha| \leq 2k$ AND $|\beta| \leq 2k$

$S \overset{*}{\Rightarrow} \alpha$, $S \overset{*}{\Rightarrow} \beta$ BY IH

SINCE $S \to aSbS \in R$

$S \Rightarrow aSbS \overset{*}{\Rightarrow} a\alpha bS \overset{*}{\Rightarrow} a\alpha b\beta$

AND SO $S \overset{*}{\Rightarrow} w$ AND $w \in \mathcal{L}(G)$

CAN REDO FOR STARTING 'b'

THUS, $w \in L \Rightarrow w \in \mathcal{L}(G)$

COMBINING (1) AND (2)

$w \in L \Leftrightarrow w \in \mathcal{L}(G)$

ALSO

$L = \mathcal{L}(G)$

ONE MORE PROOF WRT CFGs, $G=(\{P\},\{0,1\},R,P)$

R: $P \to \lambda \mid 0 \mid 1 \mid 0P0 \mid 1P1$

$\mathcal{L}(G) = \{w \mid w \text{ IS A BINARY PALINDROME}\}$

$= \{w \mid w \in \{0,1\}^* \text{ \& } w = w^R\}$

PROOF: 1. ALL BINARY PALINDROMES ARE IN $\mathcal{L}(G)$

BASIS: $|w|=0$ or $|w|=1$

$P \to \lambda$ SO $P \overset{*}{\Rightarrow} \lambda \Rightarrow \lambda \in \mathcal{L}(G)$

$P \to 0$ SO $P \overset{*}{\Rightarrow} 0 \Rightarrow 0 \in \mathcal{L}(G)$

$P \to 1$ SO $P \overset{*}{\Rightarrow} 1 \Rightarrow 1 \in \mathcal{L}(G)$

AND SO BASIS IS SHOWN

IH: ASSUME FOR ALL w, $|w| \leq k, k \geq 1$,

$P \overset{*}{\Rightarrow} w$ WHENEVER $w = w^R$

IS: LET $|w| = k+1$. SINCE $k \geq 1$, $|w| \geq 2$ & SO

$w = 0x0$ OR $w = 1x1$ for $|x| = k-1$ AND $x = x^R$

BUT THEN

$P \Rightarrow 0P0 \overset{*}{\Rightarrow} 0x0$ } IND. HYP.

OR $P \Rightarrow 1P1 \overset{*}{\Rightarrow} 1x1$

& SO $P \overset{*}{\Rightarrow} w$

PROOF: 2 SHOW: ... $\mathcal{L}(G)$ CONTAINS ONLY PALINDROMES

DO INDUCTION OF # OF STEPS IN DERIVATION

BASIS: $k=1$

EITHER $P \Rightarrow \lambda$ OR $P \Rightarrow 0$ OR $P \Rightarrow 1$ AND ALL ARE PALINDROMES

IH: ASSUME FOR ALL $i \leq k, k \geq 1$, IF $P \overset{i}{\Rightarrow} X$ AND $X \in \{0,1\}^*$ THEN X IS AN PALINDROME

IS: $P \Rightarrow 0P0 \overset{k}{\Rightarrow} 0X0$ } $X \in \{0,1\}^*$

OR $P \Rightarrow 1P1 \overset{k}{\Rightarrow} 1X1$

ARE ONLY $k+1$ LENGTH DERIVATIONS OF TERMINAL STRINGS.

SINCE $P \overset{k}{\Rightarrow} X$ IS A PALINDROME THEN SO ARE $0X0$ AND $1X1$

QED

A Challenging CFL

The complement of $\{ww \mid w \in \{a,b\}^*\}$ is a CFL. It can be described as

$$\{xy \mid |x| = |y|, x \neq y\} \cup \{w \mid |w| \text{ is odd}\}$$

where all strings are over $\{a,b\}$

$$\{w \mid w \in \{a,b\}^+ \text{ and } |w| \text{ is odd}\}$$

is regular

$$S \to aT \mid bT$$
$$T \to aS \mid bS \mid \lambda$$

We will show CFLs are closed under $\cup$

Focus on

$$\{xy \mid |x| = |y|, x \neq y, x, y \in \{a,b\}^*\}$$

CHALLENGE CONTINUED

$\{ xy \mid |x| = |y| \; x \neq y \}$

NOTE THAT WHILE WW REQUIRES THAT THERE BE NO ERRORS TRANSCRIBING (COPYING) FIRST HALF TO SECOND HALF, ITS COMPLEMENT REQUIRES JUST ONE TRANSCRIPTION ERROR. ELEMENTS ARE OF FORM

$x_1 a x_2 y_1 b y_2$ OR $x_1 b x_2 y_1 a y_2$

WHERE $|x_1| = |y_1|$ AND $|x_2| = |y_2|$

NOTE THAT WE DO NOT CARE WHAT x_1, x_2, y_1 AND y_2 ARE. ONLY THEIR LENGTHS MATTER. WITH THAT INSIGHT IN MIND, WE CAN SEE THAT

$x_1 a y_1 x_2 b y_2$ OR $x_1 b y_1 x_2 a y_2$

WHERE $|x_1| = |y_1|$ AND $|x_2| = |y_2|$ ALSO DESCRIBES ELEMENTS IN LANGUAGE. THIS IS JUST

$$\left.\begin{array}{l} S \to AB \mid BA \\ A \to CAC \mid a \\ B \to CBC \mid b \\ C \to a \mid b \end{array}\right\} \quad (\{S, A, B, C\}, \{a, b\}, R, S)$$

WHICH IS A CFG.

Simple Closures

Let $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$

where $V_1 \cap V_2 = \emptyset$

Let $L_1 = \mathcal{L}(G_1)$ and $L_2 = \mathcal{L}(G_2)$

where G_1 and G_2 are CFGs and so L_1 and L_2 are CFLs

Union: $G_3 = (V_1 \cup V_2 \cup \{S\}, \Sigma, R_3, S)$

R_3: $S \to S_1 \mid S_2$

Add R_1 and R_2 to R_3 and get $L_1 \cup L_2$

Concatenation:

R_3: $S \to S_1 S_2$

Add R_1 and R_2 to R_3 and get $L_1 \circ L_2$

Star:

R_3: $S \to S S_1 \mid \lambda$

Add R_1 to R_3 and get L_1^*

Note: $S \to S_1 S \mid \lambda$ also works

NON-CLOSURES

ASSUME, FOR NOW (PUMPING LEMMA WILL VERIFY) THAT

$$L = \{a^n b^n c^n \mid n \geq 0\}$$

IS NOT A CFL

NOW, CONSIDER TWO CFLs

$$L_1 = \{a^n b^n c^m \mid n, m \geq 0\}$$
$$= a^n b^n c^*$$

AND $L_2 = \{a^m b^n c^n \mid n, m \geq 0\}$
$$= a^* b^n c^n$$

$$L_1 \cap L_2 = L = \{a^n b^n c^n \mid n \geq 0\}$$

BUT $L_1 = \mathcal{L}(G_1)$, $L_2 = \mathcal{L}(G_2)$ WHERE

$G_1 = (\{S,T\}, \{a,b,c\}, R_1, S)$ $G_2 = (\{S,T\}, \{a,b,c\}, R_2, S)$

R_1: $S \to Sc \mid T$
$T \to aTb \mid \lambda$

R_2: $S \to aS \mid T$
$T \to bTc \mid \lambda$

SO CFLs NOT CLOSED UNDER $\cap$

PREVIOUSLY WE SHOWED NOT UNDER COMPLEMENT, IF WE CAN SHOW

$$L = \{ww \mid w \in \{a,b\}^*\}$$

IS NOT A CFL (LATER)

Types of Derivations

(1) No constraint on which non-terminal to rewrite when choices exist

(2) Rightmost: always rewrite rightmost non-terminal

(3) Leftmost: always rewrite leftmost non-terminal

While (1), (2) and (3) make no difference on what can be derived at frontier (leaves) of tree, they do typically affect intermediate trees and are integrally tied to parsing strategy (top down vs bottom up)

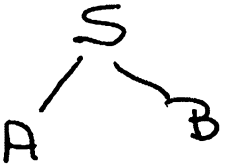
DERIVATIONS & TREES (CONTEXT FREE)

IF A GRAMMAR, G, IS A CFG THEN SYNTACTIC FORMS CAN BE ANY α, $\alpha \in (\Sigma \cup V)^*$

THUS, WHEN $S \overset{*}{\Rightarrow} \alpha$ AND α CONTAINS NON-TERMINALS, THERE ARE CHOICES

FIRST OF (1) WHICH NON-TERMINAL TO REWRITE
(2) WHICH RHS OF THAT NON-TERM TO USE

A DERIVATION (PARSE) TREE HAS A FORM THAT COULD INCLUDE CASES LIKE



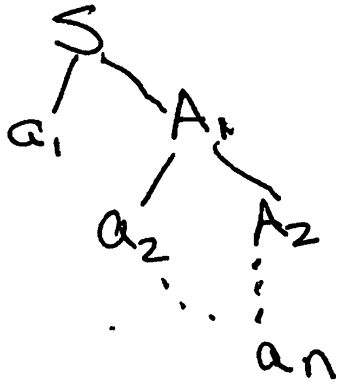
WE CAN NOW CHOOSE TO REWRITE A OR B

IMPORTANTLY, THOUGH, OUR CHOICE HAS NO EFFECT ON THE POSSIBLE OUTCOMES AS EACH SUBTREE EVOLVES INDEPENDENT OF ALL OTHER SUBTREES

DERIVATIONS & TREES (REGULAR)

IF A GRAMMAR, G, IS RIGHT LINEAR THEN EVERY NON-TERMINAL SYNTACTIC FORM IS OF THE CONSTRAINED FORM αB WHERE $\alpha \in \Sigma^*$ AND $B \in V$, THAT IS, IF $S \overset{*}{\Rightarrow} \beta$ THEN $\beta = \alpha B$ OR $\beta = \alpha$, WHERE $\alpha \in \Sigma^*$ AND $B \in V$.

IF WE DRAW A TREE ASSOCIATED WITH A DERIVATION FOR A REGULAR GRAMMAR, THE TREE LOOKS LIKE



AT EACH STAGE, THE ONLY CHOICE IS AMONG THE RIGHT-HAND SIDES (RHS) OF B-RULES, WHEN WE ARE AT

$S \overset{*}{\Rightarrow} \alpha B$

AMBIGUITY

$L = \{ a^i b^j c^k \mid k > i \text{ OR } k > j \}$

IS INHERENTLY AMBIGUOUS (WHY?)

BUT ARITHMETIC EXPRESSIONS

ARE NOT

$E \rightarrow E+T \mid E-T \mid T$

$T \rightarrow T*F \mid T/F \mid F$

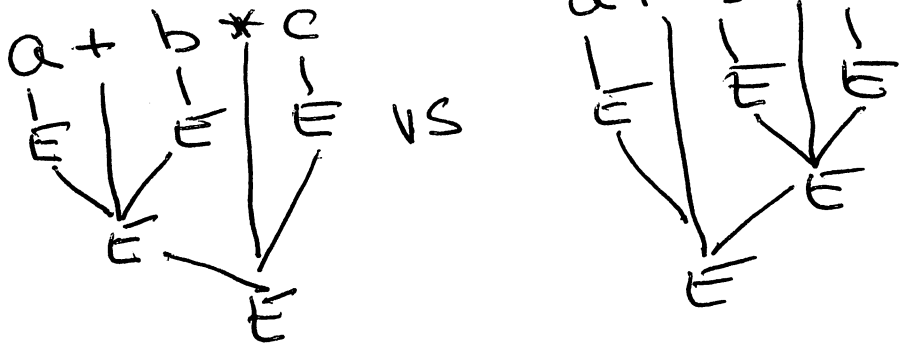
$F \rightarrow (E) \mid id \mid number$

A DFA CAN RECOGNIZE ID, NUMBER, OPS

POOR GRAMMAR

$E \rightarrow E+E \mid E-T \mid E*E \mid E/E \mid (E) \mid id \mid number$

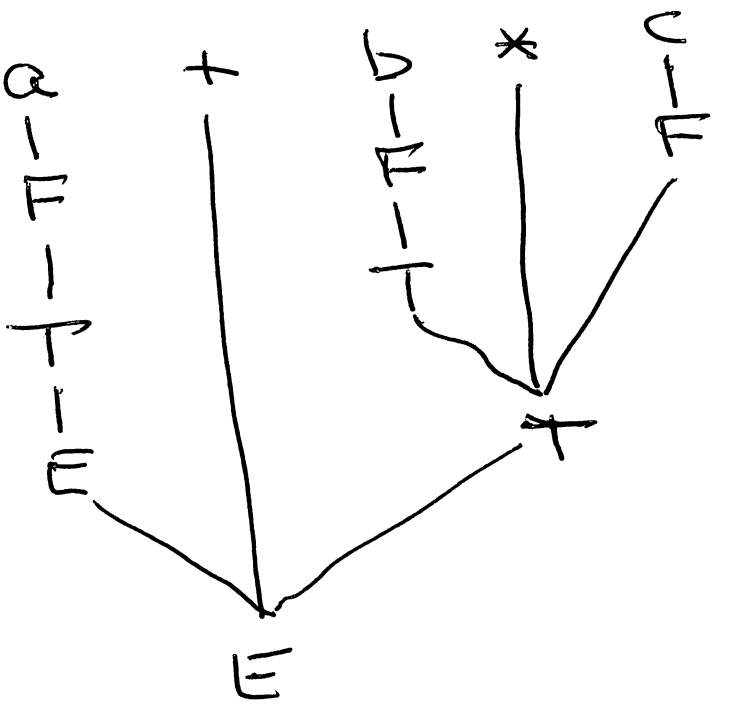
IS AMBIGUOUS BUT LANGUAGE IS NOT



LIKE 2ND AND EARLIER

GRAMMAR ENSURES THIS PARSE

UNAMBIGUOUS PARSE



IS ONLY POSSIBLE PARSE USING GRAMMAR WE PREFERRED

LEFTMOST

$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T$

$\Rightarrow a+T \Rightarrow a+T*F \Rightarrow a+F*F$

$\Rightarrow a+b*F \Rightarrow a+b*c$

RIGHTMOST

$E \Rightarrow E+T \Rightarrow E+T*F$

$\Rightarrow E+T*c \Rightarrow E+F*c \Rightarrow E+b*c$

$\Rightarrow T+b*c \Rightarrow F+b*c \Rightarrow a+b*c$

AMBIGUITY

A CFG IS AMBIGUOUS IF IT HAS

(i) TWO DISTINCT L.M. DERIVATIONS OF SOME STRING;

OR (ii) TWO DISTINCT R.M. DERIVATIONS OF SOME STRING

OR (iii) TWO DISTINCT PARSE TREE OF SOME STRING

A LANGUAGE IS INHERENTLY AMBIGUOUS IF ALL GRAMARS FOR THAT LANGUAGE ARE AMBIGUOUS

TO BE SHOWN: THERE IS NO ALGORITHM THAT CAN ANSWER THE QUESTONS, FOR ANY ARBITRARY CFG, G, "IS G AMBIGUOUS?"

AMBIGUOUS GRAMMAR

A GRAMMAR IS AMBIGUOUS IFF THERE IS SOMESTRING W IN LANGU. SUCH THAT

a) W IS THE YIELD OF TWO OR MORE DISTINCT PARSE TREES

OR b) W IS THE YIELD OF TWO OR MORE DISTINCT LEFTMOST DERIVATIONS

OR c) W IS THE YIELD OF TWO OR MORE DISTINCT RIGHTMOST DERIVATIONS

NOTE: TOP DOWN PARSER

ASSOCIATED WITH LEFTMOST DERIV.
DOES NOT LIKE LEFT RECURSION

BOTTOM UP PARSER

ASSOCIATED WITH REVERSING RIGHTMOST DERIV.
DOES NOT LIKE RIGHT RECURSION

AMBIGUITY EXAMPLES

$E \to E+E \mid E*E \mid (E) \mid id$

IS AMBIGUOUS BUT EXPRESSION LANGUAGE IS NOT INHERENTLY AMBIGUOUS

$L = \{a^i b^j c^k \mid k > i \text{ OR } k > j\}$

IS INHERENTLY AMBIGUOUS

$S \to S_1 \mid S_2$

$S_1 \to aS_1c \mid S_1c \mid T_1c$

$T_1 \to bT_1 \mid \lambda$

$S_2 \to aS_2 \mid T_2$

$T_2 \to bT_2c \mid T_2c \mid c$

CASES WHERE $k > i$ AND $k > j$ CAN ARISE FROM S_1 AND S_2

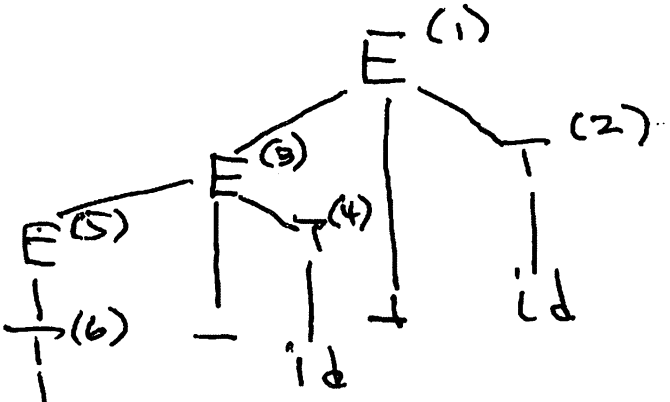
NOTE: $\{a^i b^j c^k \mid k > i \text{ AND } k > j\}$ IS NOT A CFL

RIGHTMOST AND BOTTOM UP

RIGHTMOST TREES EVOLVE AS FOLLOWS:

$E \Rightarrow E+T \mid E-T \mid T$

$T \Rightarrow id \mid (E)$



$E \Rightarrow E+T \Rightarrow E+id \Rightarrow E-T+id \Rightarrow E-id+id \Rightarrow T-id+id$
$\Rightarrow id-id+id$

BOTTOM UP IS IMPLEMENTED AS "SHIFT"/"REDUCE". THIS IS MADE DIFFICULT BY RIGHT RECURSION, SO LOVES OUR CURRENT GRAMMAR.

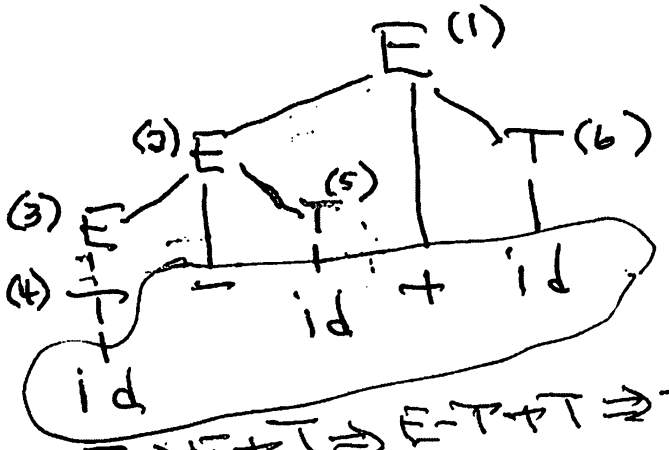
NOTE: THE LAST REWRITE IS OF LEFTMOST NON-TERMINAL AND SO FIRST REDUCE IS ON LEFT IN BOTTOM-UP

LEFTMOST AND TOP DOWN

LEFTMOST TREES EVOLVE AS FOLLOWS:

$E \rightarrow E+T \mid E-T \mid T$

$T \rightarrow id \mid (E)$



$E \Rightarrow E+T \Rightarrow E-T+T \Rightarrow T-T+T \Rightarrow id-T+T \Rightarrow id-id+T \Rightarrow id-id+id$

PROBLEM HERE IS THE PROCESS OF BUILDING TREE, BASED ON INPUT, IS "PREDICTIVE" (TOP DOWN) AND IS MADE VERY DIFFICULT WHEN GRAMMAR IS LEFT-RECURSIVE.

TOP DOWN IS OFTEN IMPLEMENTED BY "RECURSIVE DESCENT"

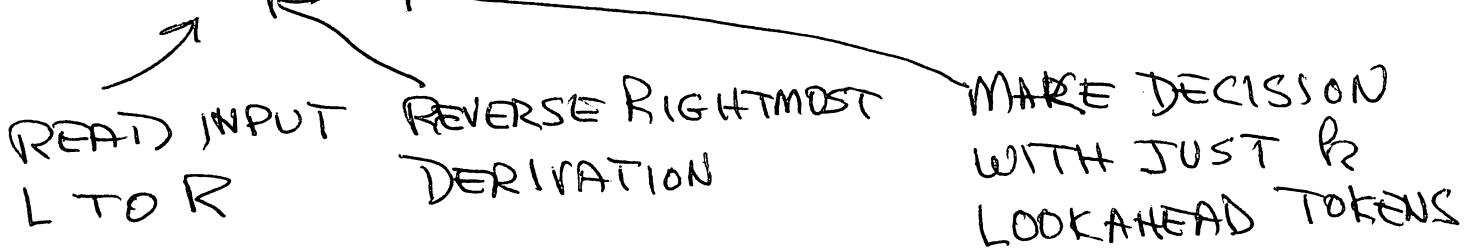
THEORY DRIVES PRACTICE

SINCE AMBIGUITY OF A GRAMMAR IS UNSOLVABLE NOT JUST UNSOLVED

CS COMPILER WRITERS COULD NOT USE ARBITRARY CFGs.

ENTER DONALD KNUTH

LR(k) PARSERS AND LR(k) GRAMMARS



LR(0): NOT GOOD ENOUGH

LR(1): GOOD ENOUGH

LR(1) LANGUAGES ARE ALL UNAMBIGUOUS CFLs

LR(1) = LR(2) = ... WHEN CONSIDERING LANGUAGES

LR(1) $\neq$ LR(2) $\neq$ ... FOR GRAMMARS

LR vs LL

LL(k) PARSERS AND LL(k) GRAMMARS

↑ USE LEFTMOST DERIVATION

$LL(1) \subsetneq LL(2) \subsetneq \cdots$

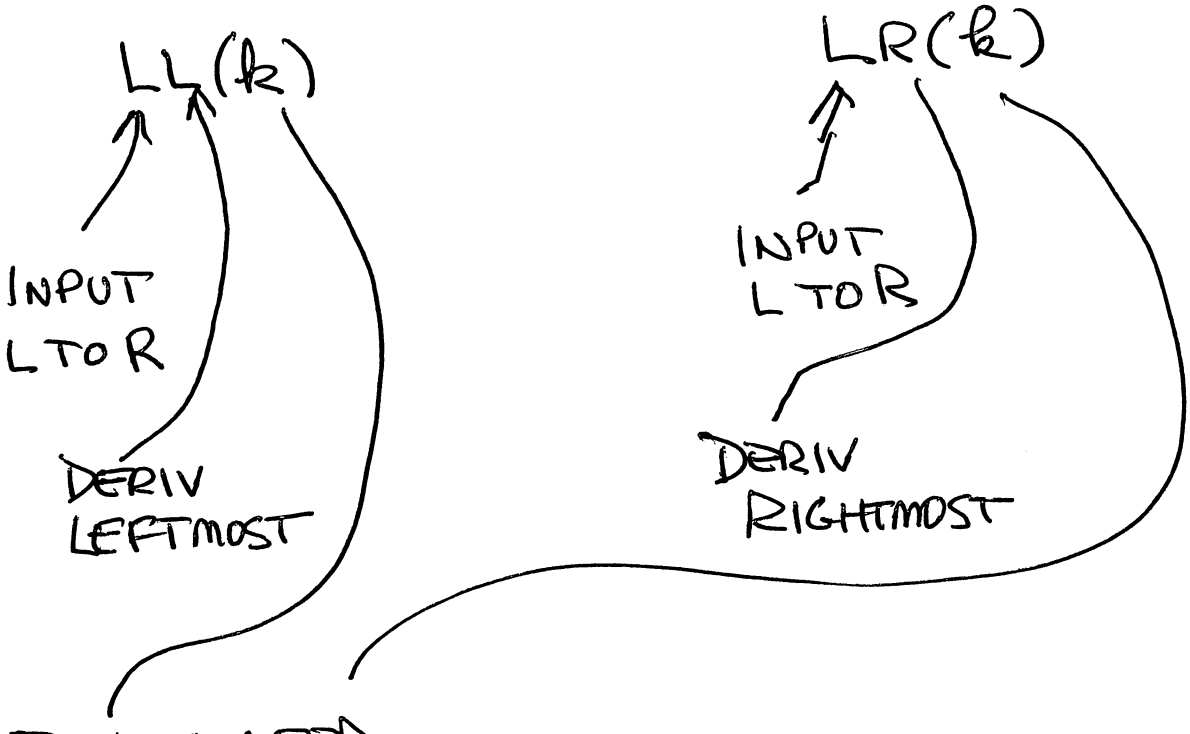
FOR BOTH GRAMMARS AND LANGUAGES

$\lim_{k \to \infty} LL(k)$ IS ALL UNAMBIGUOUS LANGUAGES

BUT

LL(1) IS USUALLY GOOD ENOUGH FOR COMPUTER LANGUAGES

PARSERS AND GRAMMAR GLASSES (CFG SUBCLASSES)



LR(1) LANGUAGES = DET. CFLs

LR(1) GRAMMARS $\subsetneq$ DET CFGs

LR(k+1) LANGUAGES = LR(k) LANGS, $k>0$

LR(k+1) GRAMMARS $\supsetneq$ LR(k+1) GRAMMARS, $k>0$

LL(k+1) LANGS $\supsetneq$ LL(k) LANG.S

LL(k+1) GRAMS $\supsetneq$ LL(k) GRAMS

$\lim_{k\to\infty}$ LL(k) LANGS. = DET CFLs

HOWEVER, LL(1) IS, IN PRACTISE, ALL WE NEED FOR PROGR. LANGS.

TOP DOWN VS BOTTOM UP

TOP DOWN PARSERS

RECURSIVE DESCENT
AKA PREDICTIVE
BASED ON L.M. DERIVATION
CHALLENGE IS WHICH RHS

BOTTOM UP PARSERS

SHIFT/REDUCE

CHALLEGE IS
SHIFT OR REDUCE
IF REDUCE, WHICH ONE

CONFLICTS IN BOTH STRATEGIES
BUT LL(k) CAN MAKE RIGHT DECISION
WITH k LOOKAHEAD
AND LR(k) CAN MAKE RIGHT DECISION
WITH k LOOKAHEAD
BOTH USE STACK (PDA LATER)

Chomsky Normal Form

All rules of form

$A \to BC$ $\quad B, C \in V$

$A \to a$ $\quad a \in \Sigma$

Or, if λ in language can have

$S \to \lambda$

Provide S is never used in RHS of any rule

Every CFG can be converted to an equivalent CNF version

Why?

- Pumping Lemma
- Polynomial Parser (CKY)
  - Great example of Dynamic Programming

Notes: 173-186