# CONTEXT-FREE GRAMMARS

**GRAMMAR** NOTATION FOR RECURSIVE DEFINITION OF LANGUAGE (NATURAL OR PROGRAMMING)

---

**EXAMPLE** (PALINDROMES)

RECALL PALINDROMES ARE STRINGS WHICH READ THE SAME FORWARDS AND BACKWARDS

THUS $L_{PAL} = \{ w \in \{0,1\}^* \mid w^R = w \}$

GRAMMAR

$$\left.\begin{array}{l} P \rightarrow \epsilon \\ P \rightarrow 0 \\ P \rightarrow 1 \end{array}\right\}$$ BASIS $\epsilon, 0, 1$ ARE PALINDROMES

$$\left.\begin{array}{l} P \rightarrow 0P0 \\ P \rightarrow 1P1 \end{array}\right\}$$ INDUCTION IF P IS A PALINDROME, THEN SO IS 0P0, 1P1.

---

**IMPLICIT**

- ALL PALINDROMES CAN BE OBTAINED BY APPLYING RULES RECURSIVELY
- ONLY PALINDROMES CAN BE SO OBTAINED.

---

**EXERCISE** PROVE ALL/ONLY BY INDUCTION ON NUMBER OF RULES APPLIED.

---

**SAMPLE APPLICATIONS**

0110 $\quad P \rightarrow 0P0 \rightarrow 01P10 \rightarrow 0110$

00100 $\quad P \rightarrow 0P0 \rightarrow 00P00 \rightarrow 00100$

# EXAMPLE (REGISTRATION LIST IN XML)

SUPPOSE WE WANTED TO GENERATE LISTS OF REGISTERED STUDENTS IN THE FORMAT:

```
<roll>
    <course> CS103 </course>
        <students>
            <name> Sally </name>
            <name> Fred </name>
        </students>
    <course> CS154 </course>
        <students>
            <name> Peter </name>
            <name> Eve </name>
        </students>
</roll>
```

---

## GRAMMAR

ROLL → <roll> CLASSES </roll>

CLASSES → CLASS CLASSES
CLASSES → CLASS

CLASS → CLASS-NAME STUDENT-LIST

CLASS-NAME → <course> TEXT </course>

STUDENT-LIST → <students> NAMES </students>

NAMES → NAME NAMES
NAMES → NAME

NAME → <name> TEXT </name>

TEXT → CHAR TEXT
TEXT → CHAR

CHAR → ... {ALL ASCII CHARACTERS

## EXAMPLE (NATURAL LANGUAGE)

SENTENCE $\longrightarrow$ NOUN-PHRASE VERB-PHRASE

NOUN-PHRASE $\longrightarrow$ ADJECTIVE NOUN-PHRASE

NOUN-PHRASE $\longrightarrow$ NOUN

NOUN $\longrightarrow$ boy

NOUN $\longrightarrow$ house

ADJECTIVE $\longrightarrow$ little

ADJECTIVE $\longrightarrow$ ...

---

## EXAMPLE ($L_{EQ}$)

$$L_{EQ} = \{ w \in \{0,1\}^* \mid w \text{ HAS EQUAL \# OF 0'S AND 1'S} \}$$

GRAMMAR

$$\begin{cases} S \longrightarrow \epsilon \\ S \longrightarrow 0A \\ S \longrightarrow 1B \end{cases} \qquad \begin{cases} A \longrightarrow 1S \\ A \longrightarrow 0AA \end{cases} \qquad \begin{cases} B \longrightarrow 0S \\ B \longrightarrow 1BB \end{cases}$$

IDEA "A" GENERATES STRINGS WITH ONE MORE 1 THAN 0, WHILE "B" GENERATES ALL STRINGS WITH ONE MORE 0 THAN 1.

VERIFY "S" GENERATES EXACTLY $L_{EQ}$ (INDUCTION).

APPLICATION

$$S \longrightarrow 0A \longrightarrow 00AA \longrightarrow 001SA \longrightarrow 0015S$$
$$\longrightarrow 0011S \longrightarrow 00111B \longrightarrow 001110S$$
$$\longrightarrow 001110$$

**FORMALLY** GRAMMAR $G = (V, T, P, S)$

**TERMINALS $T$** BASICALLY $T = \Sigma$

**VARIABLES $V$** NON-TERMINAL SYMBOLS THAT REPRESENT SETS OF STRINGS BEING DEFINED RECURSIVELY

**START SYMBOL $S$** $S \in V$ IS THE SPECIAL VARIABLE THAT GENERATES THE DESIRED LANGUAGE.

**PRODUCTION RULES $P$** RECURSIVE DEFINITIONS.

**NOTE** $T$, $V$, AND $P$ ARE ALWAYS FINITE SETS

---

**$L_{EQ}$ EXAMPLE** THE GRAMMAR $G_{EQ} = (V, T, P, S)$

$$T = \{0, 1\}$$

$$V = \{S, A, B\}$$

$$P = \{ \quad S \rightarrow \epsilon \mid 0A \mid 1B,$$
$$A \rightarrow 1S \mid 0AA,$$
$$B \rightarrow 0S \mid 1BB \}.$$

---

**NOTATION** FOR SET OF RULES

$$A \rightarrow \alpha_1,\ A \rightarrow \alpha_2,\ \ldots,\ A \rightarrow \alpha_k$$

SHORT-CUT

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_k.$$

**CONTEXT-FREE GRAMMAR (CFG)** ARE ONLY ALLOWED TO HAVE PRODUCTION RULES FOR SUBSTITUTION OF THE FORM:

$$A \longrightarrow \alpha_1 \alpha_2 \cdots \alpha_k$$

WHERE L.H.S. $A \in V$

R.H.S. $\alpha_i \in V \cup T$, FOR ALL $i$

---

**NON-CONTEX-FREE?** RULES MIGHT SPECIFY CONTEXT IN WHICH THE RULE'S SUBSTITUTION CAN BE PERFORMED

E.G. $0A1 \longrightarrow 0\alpha_1\alpha_2 \cdots \alpha_k 1$

THUS RULES CANNOT BE APPLIED IN OTHER CONTEXTS

---

## DERIVATIONS

**SINGLE STEP** FOR CFG $G$, $\forall\ \alpha, \beta, \gamma \in (V \cup T)^*$, $\forall A \in$

WE SAY $\alpha A \beta \Longrightarrow \alpha \gamma \beta$ (DIRECTLY DERIVES

IF THE RULE $A \rightarrow \gamma$ IS IN $G$.

MEANS CAN DERIVE $\alpha \gamma \beta$ FROM $\alpha A \beta$

---

**DERIVATION SEQUENCE**

SUPPOSE $\alpha_1 \Longrightarrow \alpha_2 \Longrightarrow \alpha_3 \Longrightarrow \cdots \Longrightarrow \alpha_n$

WE SAY $\alpha_1 \overset{*}{\Longrightarrow} \alpha_n$

MEANS $\alpha_1$ DERIVES $\alpha_n$ IN 0 OR MORE DERIVATION STEPS.

**OBSERVE** $\overset{*}{\Rightarrow}$ IS REFLEXIVE / TRANSITIVE CLOSURE OF $\Rightarrow$

- **REFLEXIVE** $\alpha \overset{*}{\Rightarrow} \alpha$
- **TRANSITIVE** $\alpha \overset{*}{\Rightarrow} \beta,\ \beta \overset{*}{\Rightarrow} \gamma$ IMPLIES $\alpha \overset{*}{\Rightarrow} \gamma$

---

**EXAMPLE LEQ**

$$G \begin{cases} S \to \epsilon \mid 0A \mid 1B \\ A \to 1S \mid 0AA \\ B \to 0S \mid 1BB \end{cases}$$

a) $001SA \Rightarrow 001S1S$ (USING $A \to 1S$)

b) $001S1S \Rightarrow 0011S$ (USING $S \to \epsilon$)

c) $001SA \overset{*}{\Rightarrow} 0011S$ (USING a, b)

d) $S \overset{*}{\Rightarrow} 001110$

---

**LANGUAGE** OF CFG $G = (V, T, P, S)$ IS DEFINED AS

$$L(G) = \{ w \in T^* \mid S \overset{*}{\Rightarrow} w \}$$

**CONTEXT-FREE LANGUAGE (CFL)** ANY LANGUAGE WHICH HAS A CFG G

---

## TERMINOLOGY

**SENTENCE** ANY $w \in T^*$ SUCH THAT $S \overset{*}{\Rightarrow} w$ (I.E. $w \in L$)

**SENTENTIAL FORM** ANY $\alpha \in (V \cup T)^*$ SUCH THAT $S \overset{*}{\Rightarrow} \alpha$

---

**TERMINALS** $a, b, c, \ldots$

**VARIABLES** $A, B, C, \ldots$

**TERMINAL STRINGS** $\ldots u, v, w, x, y, z$

$V \cup T$ $\ldots X, Y, Z$

SENTENTIAL FORMS $\alpha, \beta, \gamma \ldots$

## EXAMPLE (ARITHMETIC EXPRESSIONS)

- $G = (V, T, P, S)$
- $V = \{E, I\}$
- $S = E$
- $T = \{x, y, z, +, *, (, )\}$
- $P$

$$\begin{cases} E \rightarrow I \mid (E) \mid E+E \mid E*E \\ I \rightarrow x \mid y \mid z \end{cases}$$

---

**APPLY**

$$E \Rightarrow E+E \Rightarrow E+E*E \Rightarrow I+E*E$$
$$\Rightarrow x+E*E \Rightarrow x+I*E \Rightarrow x+y*E$$
$$\Rightarrow x+y*I \Rightarrow x+y*z$$

---

## NON-DETERMINISM

CONSIDER $E \overset{*}{\Rightarrow} E+E*E$

NOW POSSIBLE DERIVATION STEPS?

- EXPAND ANY ONE OF THREE E'S
- USE ANY ONE OF FOUR RULES FOR E

NON-DETERMINISM DECIDE

- WHICH VARIABLE?
- WHICH RULE?

---

OBSERVE DECIDING WHICH RULE TO APPLY IS IMPORTANT, NON-DETERMINISM WHICH GIVES CFG ITS POWER & FLEXIBILITY

BUT DECIDING WHICH VARIABLE ONLY SERVES TO CONFUSE US (OR, THE PARSER).

**LEFTMOST DERIVATION** ALWAYS SUBSTITUTES THE LEFTMOS VARIABLE BY A PRODUCTION RULE IN SENTENTIAL FORMS ARISING IN COURSE OF A DERIVATION

**NOTATION** $\underset{lm}{\Longrightarrow}$

**EXAMPLE** $E \underset{lm}{\Longrightarrow} E+E \underset{lm}{\Longrightarrow} I+E \underset{lm}{\Longrightarrow} x+E$

$\underset{lm}{\Longrightarrow} x+E*E \underset{lm}{\Longrightarrow} x+I*E \underset{lm}{\Longrightarrow} \cdots$

---

**SIMILARLY** CAN DEFINE RIGHTMOST DERIVATIONS

$E \underset{rm}{\Longrightarrow} E+E \underset{rm}{\Longrightarrow} E+E*E \underset{rm}{\Longrightarrow} E+E*I \cdots$

---

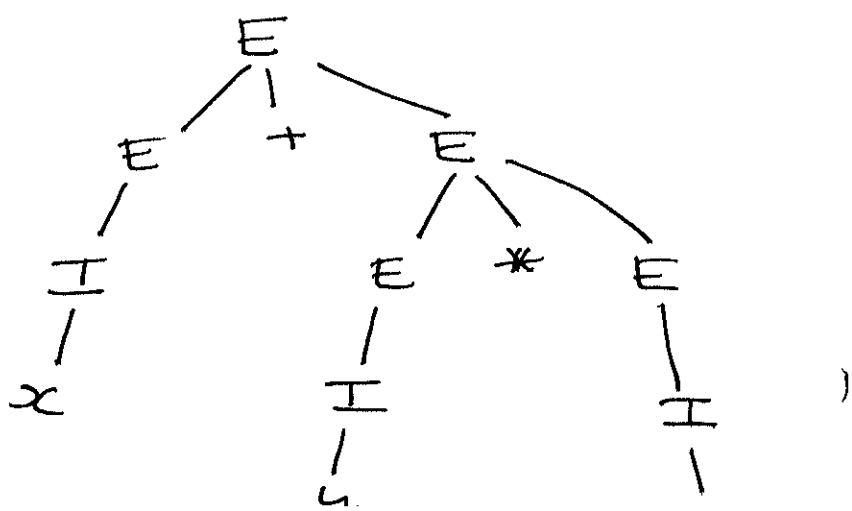
**NOW** WE CAN TALK ABOUT "CANONICAL" DERIVATION SEQUENCES $\overset{*}{\underset{lm}{\Longrightarrow}}$ OR $\overset{*}{\underset{rm}{\Longrightarrow}}$

---

## DERIVATION TREES

**ALSO CALLED** PARSE TREES IN COMPILERS

**IDEA** REPRESENT DERIVATIONS PICTORIALLY



**ABOVE TREE** REPRESENTS THE DERIVATION

$$E \overset{*}{\Longrightarrow} x+y*z.$$

---

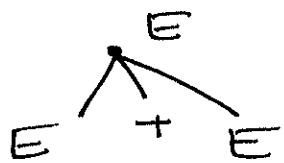
**IN GENERAL** TO REPRESENT $A \overset{*}{\Longrightarrow} \alpha$

**ROOT** LABELED A

**LEAVES** IN LEFT-TO-RIGHT ORDER GIVES $\alpha$

**INTERNAL NODES** ARE LABELED WITH VARIABLE AND THEIR CHILDREN SPECIFY THE PRODUCTION RULE APPLIED TO THEM.

**EXAMPLE**




COMES FROM $E \rightarrow E+E$

---

**A-TREE** TREE ROOTED AT VARIABLE A

**BUT** SIMPLY CALL PARSE TREE IF ROOT IS S

---

**YIELD | FRONTIER** OF A TREE IS THE SEQUENCE OF LEAF LABELS IN LEFT-TO-RIGHT ORDER.

---

**THEOREM** $\alpha$ IS THE YIELD OF AN A-TREE

$$\Longleftrightarrow \quad A \overset{*}{\Longrightarrow} \alpha$$

**PROOF** — INDUCTION ON TREE HEIGHT

— SEE TEXTBOOK

<u>OBSERVE</u> PRECEDING PARSE TREE DOES NOT SPECIFY A UNIQUE WAY TO DERIVE $\alpha$ FROM A.

<u>IN FACT</u> IT REMOVES NON-DET OF WHICH RULES TO APPLY, BUT ORDER IS LEFT UNSPECIFIED

---

<u>LEFTMOST DERIVATION</u> IS OBTAINED BY TRAVERSING THE TREE IN DEPTH-FIRST ORDER, ALWAYS GOING INTO LEFT SUBTREES BEFORE RIGHT ONE

$$E \underset{lm}{\Rightarrow} E+E \underset{lm}{\Rightarrow} I+E \underset{lm}{\Rightarrow} x+E \underset{lm}{\Rightarrow} x+E*E$$

<u>SIMILARLY</u> RIGHTMOST DERIVATIONS COME FROM DEPTH-FIRST TRAVERSAL GOING INTO RIGHT SUBTREES FIRS

$$E \underset{rm}{\Rightarrow} E+E \underset{rm}{\Rightarrow} E+E*E \underset{rm}{\Rightarrow} E+E*I \underset{rm}{\Rightarrow} \cdots$$

---

<u>THEOREM</u> FOLLOWING ARE ALL EQUIVALENT STATEMENTS FOR CFG $G=(V,T,P,S)$ AND STRING $w \in T^*$

(a) $w \in L(G)$ (<u>OR</u> $S \overset{*}{\Rightarrow} w$)

(b) $S \overset{*}{\underset{lm}{\Rightarrow}} w$

(c) $S \overset{*}{\underset{rm}{\Rightarrow}} w$

(d) THERE EXISTS AN S-TREE WITH YIELD $w$

---

<u>PROOF</u> OBVIOUS

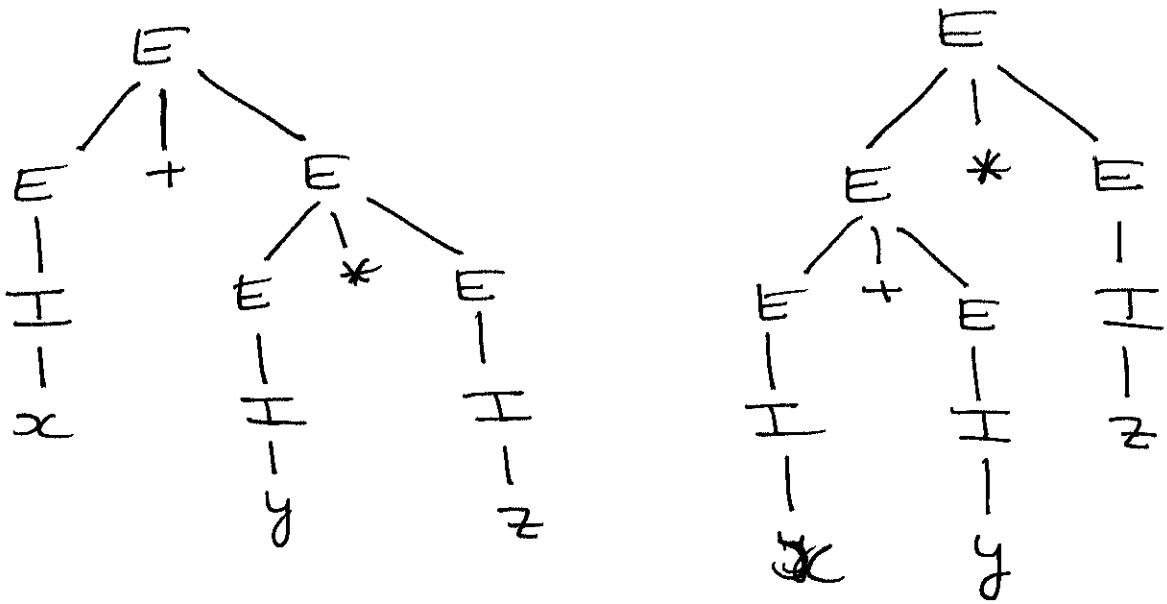
REMARK HOLDS EVEN IF $w$ IS REPLACED BY A SENT FOR

OF COURSE WE COULD ALWAYS USE $\ell m$ DERIVATION TO SPECIFY A CANONICAL WAY TO DERIVE ANY $w \in L(G)$ OR CONVERT PARSE TREE TO UNIQUE DERIVATION — SIMPLIFYING TASK OF PARSERS

---

BUT WHAT IF SOME $w \in L(G)$ HAS TWO DISTINCT PARSE TREES & HENCE TWO DISTINCT $\ell m$ DERIVATIONS?

EXAMPLE $x + y * z$




L.M. DERIVATIONS

TREE 1 $E \underset{\ell m}{\Longrightarrow} E+E \underset{\ell m}{\Longrightarrow} I+E \underset{\ell m}{\Longrightarrow} x+E \underset{\ell m}{\Longrightarrow} x+E*E$

TREE 2 $E \underset{\ell m}{\Longrightarrow} E*E \underset{\ell m}{\Longrightarrow} E+E*E \underset{\ell m}{\Longrightarrow} \ldots$

**NOTE** THIS NOT JUST A SYNTACTIC PROBLEM, AS WE GET TWO DIFFERENT SEMANTIC INTERPRETATIONS

TREE 1 $x + (y * z)$

TREE 2 $(x+y) * z$.

---

**DEFN** A CFG $G$ IS AMBIGUOUS IF FOR SOME $w \in L(G)$ THERE EXIST TWO DISTINCT PARSE TREES.

**IN COMPILERS** PARSE TREE DETERMINE INTERPRETATION AND WE CANNOT ALLOW AMBIGUITY

**OF COURSE** WE CAN FIX PROBLEM BY FORCING USE OF PARENTHESIS, BUT WE SHOULD REALLY RE-DESIGN THE GRAMMAR TO BE UNAMBIGUOUS BY ENCODING PRECEDENCE OF OPERATORS (SEE TEXTBOOK FOR RE-DESIGNED GRAMMAR)

---

**WHILE** ABOVE GRAMMAR CAN BE RE-DESIGNED TO REMOVE AMBIGUITY, THIS IS NOT ALWAYS POSSIBLE

**DEFN** A CFL IS SAID TO BE INHERENTLY AMBIGUOUS IF ALL ITS GRAMMARS ARE AMBIGUOUS

---

**EXAMPLE** $L = \{a^n b^n c^m d^m \mid n, m \geq 1\} \cup \{a^n b^m c^m d^n \mid n, m \geq 1\}$

**CONSIDER** STRINGS OF THE FORM $a^k b^k c^k d^k$

— CANNOT TELL WHETHER THIS COMES FROM FIRST OR SECOND TYPE OF STRINGS IN $L$ AND ANY CFG MUST ALLOW BOTH OF THESE POSSIBILITIES.