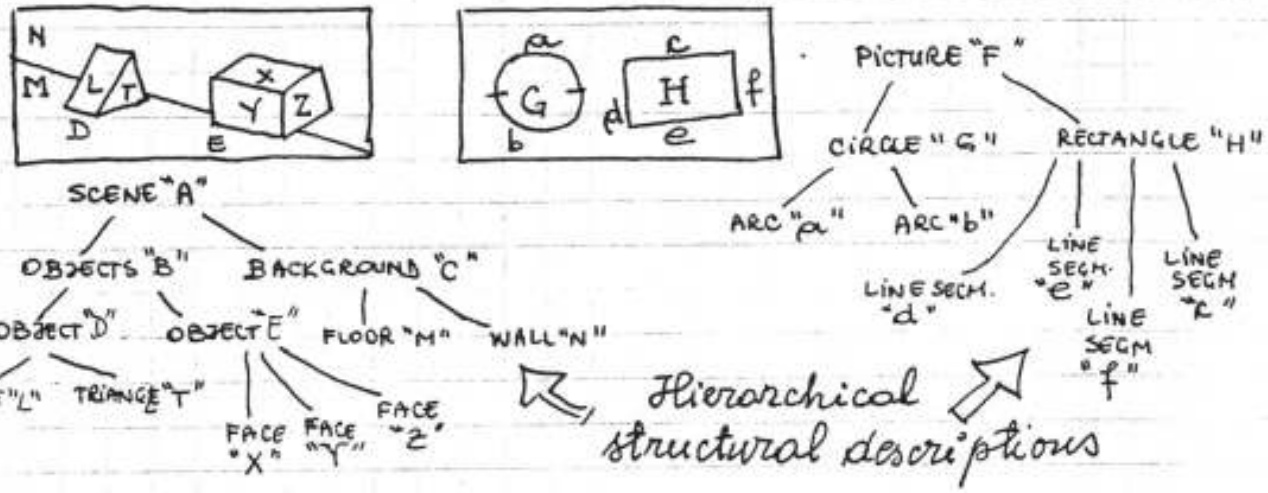
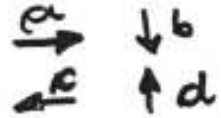
SYNTACTIC PATTERN RECOGNITION

PROVIDES A CAPABILITY FOR DESCRIBING A LARGE SET OF COMPLEX PATTERNS BY USING A SMALL SET OF SIMPLE PATTERN PRIMITIVES AND OF GRAMMATICAL RULES. THE "LANGUAGE" THAT PROVIDES THE STRUCTURAL DESCRIPTION OF PATTERNS IN TERMS OF A SET OF PATTERN PRIMITIVES AND THEIR COMPOSITION OPERATORS IS CALLED "PATTERN DESCRIPTION LANGUAGE"

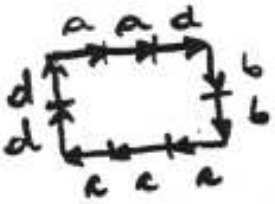


- ONE OF THE MOST ATTRACTIVE ASPECTS OF THE SYNTACTIC PATTERN RECOGNITION IS THE RECURSIVE NATURE OF THE GRAMMAR RULES WHICH ALLOWS TO EXPRESS IN A VERY COMPACT WAY SOME BASIC STRUCTURAL CHARACTERISTICS OF AN INFINITE SET OF "SENTENCES" (DESCRIBING A PATTERN).

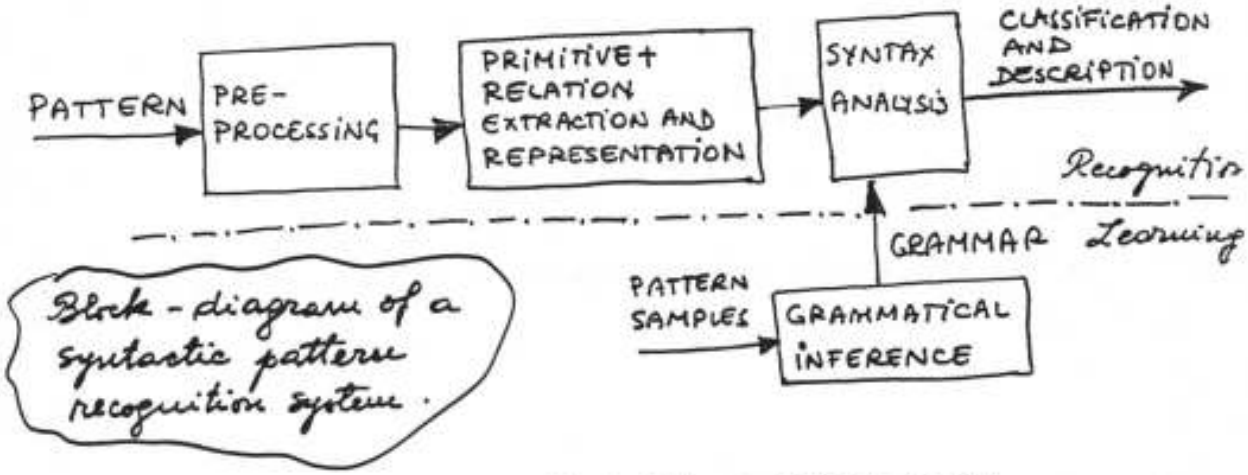
- THE VARIOUS RELATIONS OR COMPOSITION OPERATIONS DEFINED AMONG SUBPATTERNS CAN USUALLY BE DESCRIBED IN TERMS OF LOGICAL AND/OR MATHEMATICAL OPERATIONS.



e.g. USING "CONCATENATION" AS THE ONLY COMPOSITION OPERATION, THEN THE RECTANGLE FROM THE NEARBY FIG. IS REPRESENTED BY THE STRING



a a a b b c c c d d



Block-diagram of a syntactic pattern recognition system.

PRIMITIVE + RELATION EXTRACTION AND REPRESENTATION

EACH PATTERN IS DECOMPOSED IN SUBPATTERN UNTIL IT IS REPRESENTED BY A SET OF PRIMITIVES WITH SPECIFIED RELATIONS.

NO GENERAL SOLUTION FOR PRIMITIVE SELECTION; THERE ARE, HOWEVER, TWO REQUIREMENTS:

i.) THE PRIMITIVES SHOULD PROVIDE A COMPACT AND APPROPRIATE DESCRIPTION OF THE PATTERN IN TERMS OF THE SPECIFIED STRUCTURAL RELATIONS.

ii.) THE PRIMITIVES SHOULD BE EASILY OBTAINED BY NON-SYNTACTIC METHODS.

THE CONSTRUCTION OF THE GRAMMAR

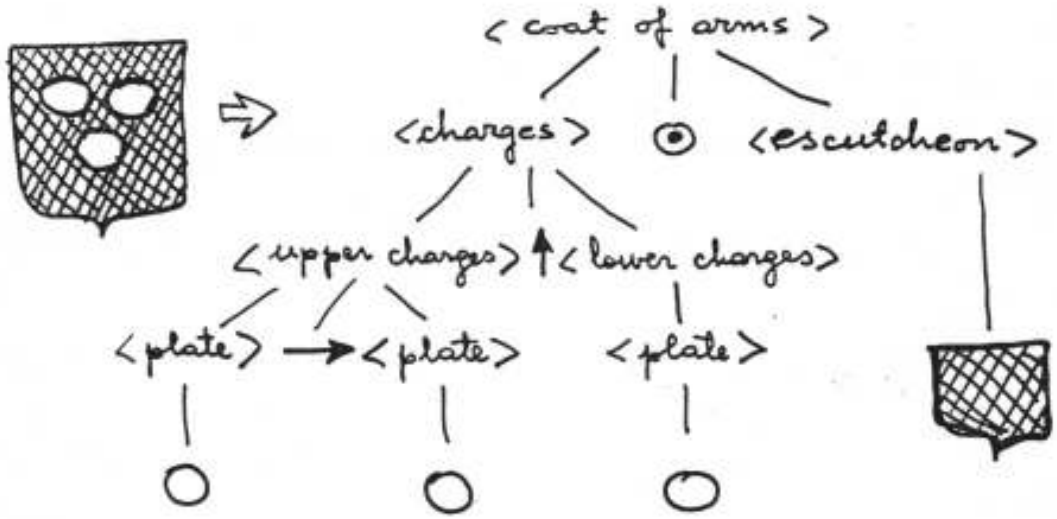
IT IS AFFECTED BY THE SELECTED PRIMITIVES AND BY THE TRADE-OFF BETWEEN THE DESCRIPTIVE POWER OF THE GRAMMAR AND EFFICIENCY OF THE SYNTACTIC ANALYSIS.

GRAMMARS
- ONE-DIMENSIONAL
- MULTI-DIMENSIONAL

SYNTAX ANALYSIS (PARSING)

THE "PARSER" (SYNTAX ANALYSER) DECIDES WHETHER OR NOT THE REPRESENTATION (THE "SENTENCE") IS SYNTACTICALLY CORRECT. i.e. WHETHER IT BELONGS TO THE CLASS OF PATTERNS DEFINED BY A GIVEN GRAMMAR.

EXAMPLE OF A TREE (PARSE) DESCRIPTION OF THE STRUCTURE OF AN IMAGE:



THE COMPOSITION RULE FOR THIS IMAGE ARE:

<coat of arms> ::= <charges> ⊙ <escutcheon>

<charges> ::= <upper charges> ↑ <lower charges>

<upper charges> ::= <plate> → <plate>

PRIMITIVES

<plate> ::= ○

<escutcheon> ::= 

(azure) (gules) (sable)

RELATIONS BETWEEN PRIMITIVES AND BETWEEN SUBSTRUCTURES.

⊙ "is contained in"

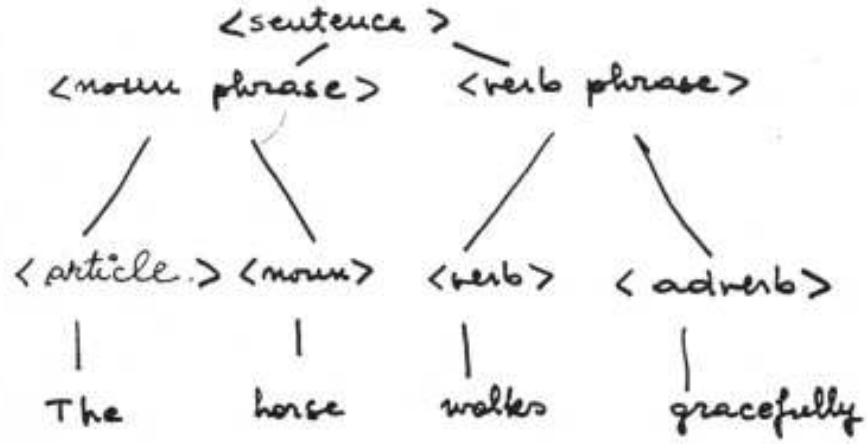
↑ "is above"

→ "is to the left of"

ELEMENTS OF FORMAL LANGUAGES

PHRASE-STRUCTURE GRAMMARS.

PARSING OF A SIMPLE ENGLISH SENTENCE:



▶ Language representation < Grammars ≡ Automata

▶ A PHRASE-STRUCTURE GRAMMAR IS AN ORDERED 4-TUPLE:

$$G = (V_N, V_T, P, S)$$

WHERE:

V_N is A FINITE, NON-EMPTY SET OF NONTERMINALS (VARIABLES);

V_T is A FINITE, NON-EMPTY SET OF TERMINALS

SUCH THAT $V_N \cap V_T = \phi$

P is A FINITE, NON-EMPTY SET OF PRODUCTION RULES (OR REWRITING RULES)

$S \in V_N$ IS THE START SYMBOL.

• CONVENTIONS CONCERNING NOTATIONS:

NONTERMINALS: $(A, B, C, D, \cdots)$; STRINGS OF NONTERMINALS AND TERMINALS: $(\alpha, \beta, \gamma, \delta, \ldots)$

TERMINALS: $\{(a, b, c, d, \ldots)$; $(0, 1, 2, 3, \ldots)$; STRINGS OF TERMINALS: $(u, v, w, x, y, \ldots)$

CONSTRUCTING A GRAMMAR "G" FOR A GIVEN LANGUAGE "L" IS REQUIRED THAT $L(G)$ CONTAINS ALL STRINGS OF "L", $L(G)$ CONTAINS NO STRING NOT IN "L"

EXAMPLE : $G = (V_N, V_T, P, S)$

$V_N = \{S, A, B, C\}$; $V_T = \{a, b, c\}$

$$P = \begin{cases} S \to aSBc \ ; (1) & bB \to bb \ ; (4) \\ S \to abC \ ; (2) & C \to c \ ; (5) \\ CB \to BC \ ; (3) & \end{cases}$$

IT CAN BE PROVED THAT : $L(G) = \{a^n b^n c^n / n \geq 1\}$

FOR THE "SENTENCE" $a^2b^2c^2$ A DERIVATION IS:

$$S \xrightarrow[(1)]{} aSBC \xrightarrow[2]{} aabCBC \xrightarrow[(3)]{} aabBCC \xrightarrow[(4)]{} aabbCC \xrightarrow[(5)]{(5)} aabbcc \ (= a^2b^2c^2)$$

CHOMSKY HIERARCHY OF GRAMMARS

TYPE 0 : UNRESTRICTED OR RECURSIVLY ENUMERABLE "RE"
NO RESTRICTIONS IN THE PRODUCTION RULES : $\alpha \to \beta$ | α CONTAINS AT LEAST ONE NONTERMINAL

TYPE 1 : CONTEXT SENSITIVE "CS" : $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$

TYPE 2 : CONTEXT FREE "CF" : $A \to \beta$

TYPE 3 : REGULAR "R" : $A \to aB$ | $A, B \in V_N$
or $A \to a$ | $a \in V_T$

EXAMPLE OF A REGULAR LANGUAGE: $G_1 = (\{S, A_1, A_2, A_3, B_1, B_2, B_3\}, \{a, b\}, P, S)$

$$P : \begin{cases} S \to aA_1 \ | \ A_1 \to aA_2 \ | \ A_2 \to aB_3 \ | \ B_2 \to bB_1 \\ S \to aB_1 \ | \ A_1 \to aB_2 \ | \ B_3 \to bB_2 \ | \ B_1 \to b \end{cases}$$

$L(G_1) = \{ab, a^2b^2, a^3b^3\}$ WHICH IS A FINITE LANGUAGE

EXAMPLE OF A CONTEXT FREE LANGUAGE: $G_2 = (\{S\}, \{a, b\}, P, S)$

$P : S \to aSb \ ; \ S \to ab \ ; \ L(G_2) = \{a^n b^n / n \geq 1\}$ WHICH IS AN INFINITE LANGUAGE

PDL (developed by Shaw) HAS A CONTEXT-FREE GRAMMAR $G(V_N, V_T, P, S)$, WHERE

$V_N = \{S, SL\}$, $V_T = \{b\} \cup \{+, \times, -, *, \sim, /, (,)\} \cup \{\ell\}$,

"b" MAY BE ANY PRIMITIVE (INCLUDING THE "NULL POINT PRIMITIVE" λ WHICH HAS IDENTICAL TAIL AND HEAD)

"ℓ" IS A LABEL DESIGNATOR WHICH IS USED TO ALLOW CROSS REFERENCE TO THE EXPRESSIONS "S" WITHIN A DESCRIPTION;

"P": $S \to b$, $S \to (S \phi_b S)$, $S \to (\sim S)$, $S \to SL$,
$S \to (/SL)$, $SL \to S^{\ell}$, $SL \to (SL\, \phi_b\, SL)$,
$SL \to (\sim SL)$, $SL \to (/SL)$, $\phi_b \to +$,
$\phi_b \to \times$, $\phi_b \to -$, $\phi_b \to *$

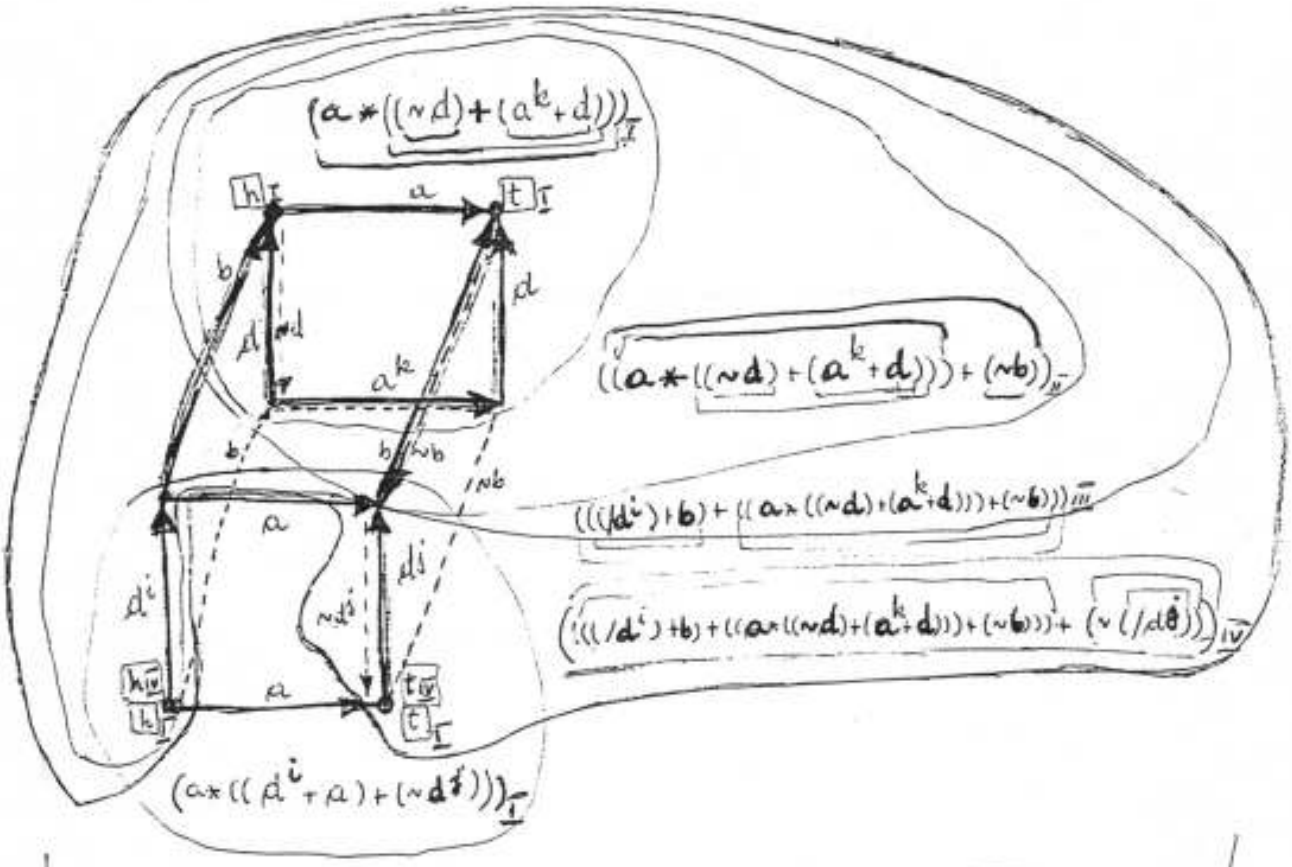
THE / OPERATOR IS USED TO ENABLE THE TAIL AND HEAD OF AN EXPRESSION TO BE ARBITRARILY LOCATED.

EXAMPLE A 3-D CUBE:

PRIMITIVES: $\underset{a}{\rightarrow}$ $\underset{b}{\nearrow}$ $\underset{d}{\uparrow}$

PDL EXPRESSION DESCRIBING THE 3-D CUBE PATTERN IS:

$$(((a * ((d^i + a) + (\sim d^i))) * ((((/d^i) + b) +$$
$$+ ((a * ((\sim d) + (a^k + d))) + (\sim b))) + (\sim (/d^i))))$$
$$* ((b + (/a^k)) + (\sim b)))$$



$$\Big(\big((a*((d^i+a)+(\sim d^i)))*(((((/d^i)+b)+((a*((\sim d)+(a^k+d)))+(\sim b)))+(\sim(/d^i)))\big)$$

$$*\ \big((b+(/a^k))+(\sim b)\big)\Big)$$

THIS IS THE "PDL" DESCRIPTION OF THE 3-D CUBE PATTERN.

EXAMPLE "PDL" STRUCTURAL DESCRIPTION OF THE CHARACTER "A", AND "HOUSE" PATTERN

$G = (V_N, V_T, P, S)$

$V_N = \{S, A, \text{HOUSE}, \text{TRIANGLE}\}$

$V_T = \{ \xrightarrow{a}, {}^{b}\nearrow, \searrow^{c}, \downarrow^{e}, (,), +, \times, -, *, \sim \}$

P: $S \to A$, $S \to \text{HOUSE}$

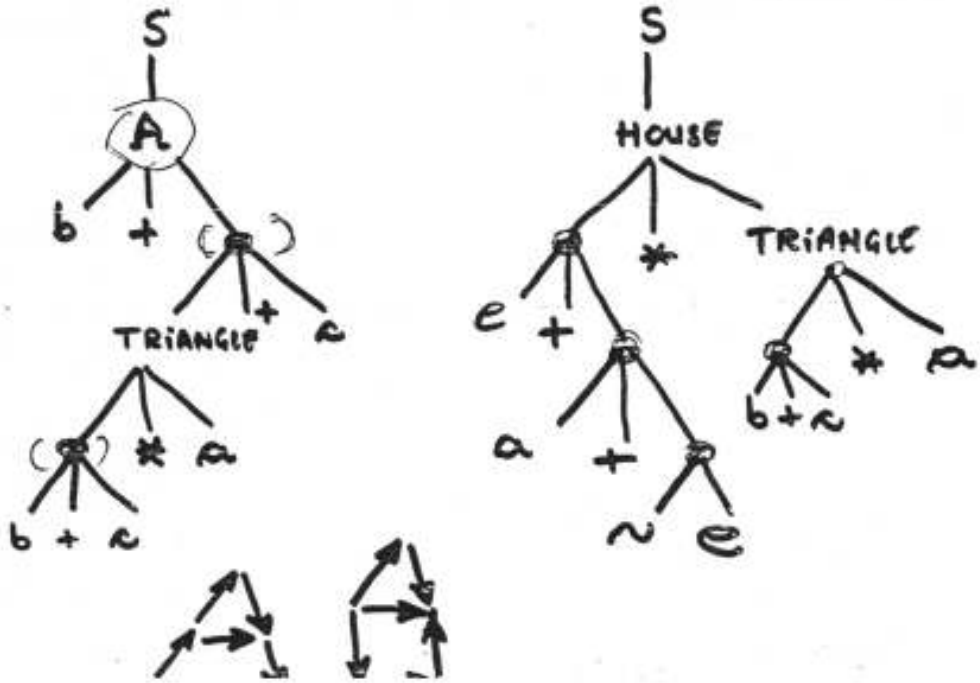
$A \to (b + (\text{TRIANGLE} + c))$

$\text{HOUSE} \to ((e + (a + (\sim e))) * \text{TRIANGLE})$

$\text{TRIANGLE} \to ((b + c) * a)$

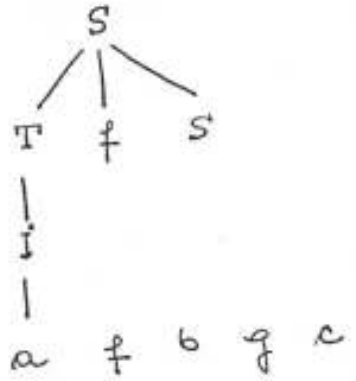
$L(G) = \{(b + (\overbrace{((b + c) * a)}^{\text{TRIANGLE}} + c)),$

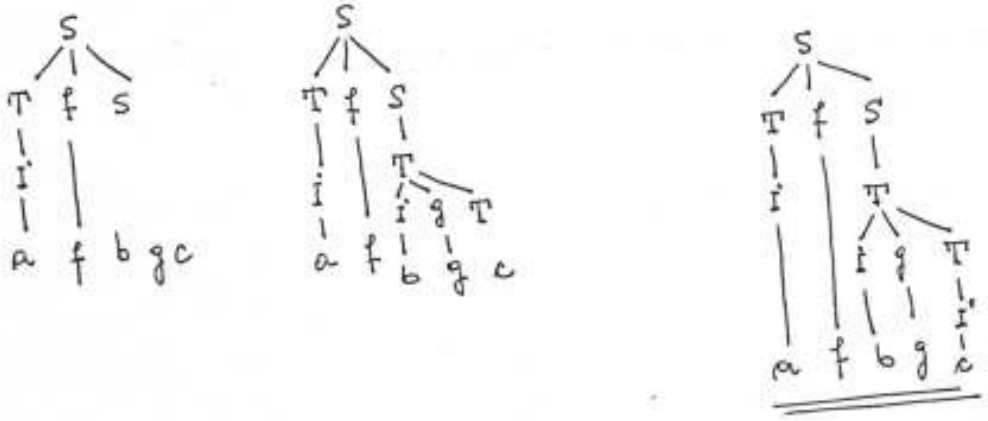
$((e + (a + (\sim e))) * ((b + c) * a))\}$



1-9

$x = afbgc$



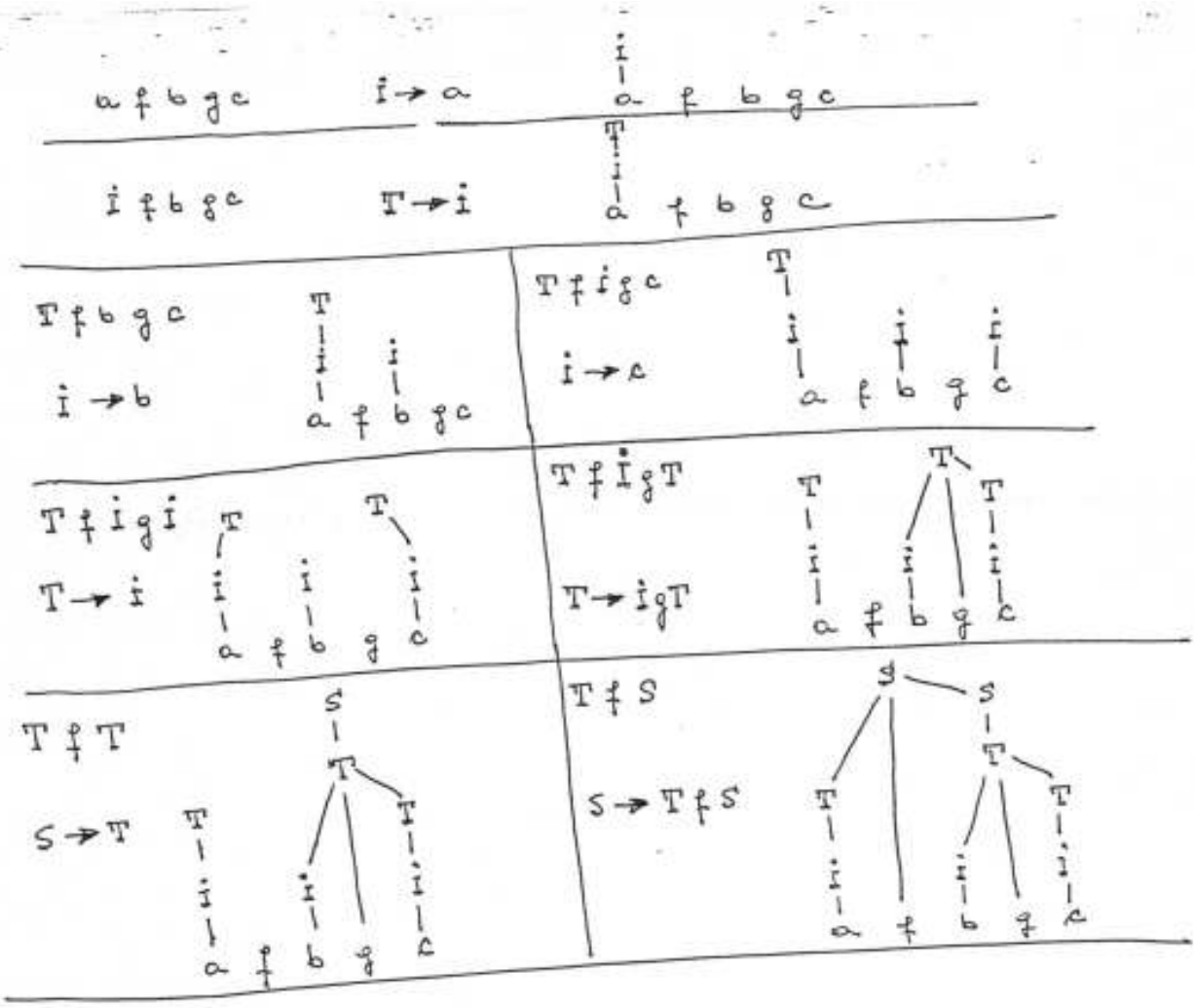


SINCE THE EXAMPLE HAS CHOOSED THE CORRECT PRODUCTIONS, THE TERMINALS MATCH THE SYMBOLS OF x. IN GENERAL THE MATCH MAY NOT BE MADE, AND THE PARSE FAILS.

▶ BOTTOM-UP PARSING

IT STARTS WITH THE STRING x AND APPLIES THE PRODUCTION BACKWARD, TRYING TO CONTACT TO THE "SENTENCE" SYMBOL S. i.e. THE STRING IS SEARCHED FOR SUBSTRINGS WHICH ARE RIGHT PARTS OF PRODUCTIONS. THESE ARE THEN REPLACED BY THE CORRESPONDING LEFT SIDE.

EXAMPLE - P2 CONSIDER THE SAME GRAMMAR AS IN "EXAMPLE-P1".



- "SHANN" HAS USED "TOP-DOWN" METHOD.
- A PURE "TOP-DOWN" METHOD HAS THE ADVANTAGE THAT THE SYNTAX DIRECTLY EXPRESSES THE ALGORITHM FOR ANALYSIS.
  - ANY INEFFICIENCIES DUE TO THE BACKTRACKING CAUSED BY FALSE TRIALS WOULD BE INSIGNIFICANT FOR THE RECOGN. TIME. ONCE A PATTERN PRIMITIVE IS RECOGNIZED, IT IS STORED; THUS, IF A GOAL FAILS, ITS PRIMITIVES MAY BE USED LATER IN THE ANALYSIS WITHOUT RECOGNITION.
- THE PURE "BOTTOM-UP" METHOD IS NOT PARTICULARLY EFFICIENT BECAUSE A LARGE NUMBER OF FALSE TRIALS MAY BE MADE.

IT IS IN PRINCIPLE UNIMPORTANT HOW WE FILL THE INTERIOR OF THE TRIANGLE. IT MAY BE DONE "TOP-DOWN" OR BOTTOM-UP.

BOTH STRATEGIES ARE "LEFT-RIGHT" FROM THE POINT OF VIEW OF THE ORDER OF PROCESSING THE SYMBOLS IN THE SENTENCE.

TOP-DOWN PARSING

IT STARTS WITH THE "SENTENCE" (START) SYMBOL S AND MAKES SUCCESSIVE SUBSTITUTIONS FOR NONTERMINALS TO TRY TO FIT THE SENTENCE.

IT IS "GOAL-ORIENTED". THE GOAL IS TO MAKE A "PREDICTION" THAT THE STRING TO BE ANALYZED IS ACTUALLY A SENTENCE WITH RESPECT TO THE GIVEN GRAMMAR.

- THE FIRST STEP IS TO SEE WHETHER THE STRING CAN BE REDUCED TO THE RIGHT PART $X_1 X_2 \ldots X_m$ OF SOME PRODUCTION:

$$S \rightarrow X_1 X_2 \cdots X_m$$

IF X_1 IS A TERMINAL, THEN THE STRING MUST BEGIN WITH THIS TERMINAL. IF X_1 IS A NONTERMINAL, A SUBGOAL IS ESTABLISHED AND TRIED: SEE WHETHER SOME HEAD OF A STRING MAY BE REDUCED TO A TERMINAL. IF THIS PROVES TO BE POSSIBLE, X_2 IS TESTED IN THE SAME MANNER, AND SO ON.

IF NO MATCH CAN BE FOUND FOR SOME X_i, THEN APPLICATION OF AN ALTERNATIVE PRODUCTION $S \rightarrow X_1' X_2' \ldots X_m'$ IS ATTEMPTED.

EXAMPLE-P1 $\quad G = (V_N, V_T, P, S)$

$V_N = \{S, T, i\}$, $V_T = \{a, b, c, f, g\}$.

$$P: \quad S \rightarrow T, \quad S \rightarrow T f S, \quad T \rightarrow i g T, \quad T \rightarrow i,$$
$$i \rightarrow a, \quad i \rightarrow b, \quad i \rightarrow c.$$

SYNTAX ANALYSIS AS A RECOGNITION PROCEDURE

- AFTER A GRAMMAR IS CONSTRUCTED TO GENERATE A LANGUAGE TO DESCRIBE THE PATTERN UNDER STUDY, THE NEXT STEP IS TO DESIGN A RECOGNIZER THAT WILL RECOGNIZE THE PATTERNS (REPRESENTED BY STRINGS) GENERATED BY THE GRAMMAR.

- IF THE GRAMMAR IS FINITE STATE, A DETERMINISTIC FINITE STATE AUTOMATON CAN BE CONSTRUCTED TO RECOGNIZE THE STRINGS GENERATED BY GRAMMAR.

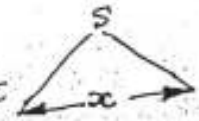
- IF THE GRAMMAR IS CONTEXT FREE, A NONDETERMINISTIC AUTOMATON IS USUALLY REQUIRED. THE RECOGNITION ALGORITHM/PROCEDURE IS CALLED "SYNTAX ANALYSIS".

  THE OUTPUT FROM THE ANALYZER USUALLY INCLUDES NOT ONLY THE DECISION OF ACCEPTING THE STRING GENERATED BY GRAMMAR, BUT ALSO THE DERIVATION TREE OF THE STRING:

"PARSING" (SYNTAX ANALYZING) { GIVEN A SENTENCE x AND A CONTEXT-FREE GRAMMAR, CONSTRUCT A DERIVATION OF x AND FIND A CORRESPONDING DERIVATION TREE

ALTERNATIVELY

GIVEN A PARTICULAR SENTENCE x AND A GRAMMAR G, CONSTRUCT A TRIANGLE: AND ATTEMPT TO FILL THE INTERIOR OF THE TRIANGLE



WITH A SELF-CONSISTENT TREE OF DERIVATIONS, NAMELY, THE PARSE.