Harvard CS 121 and CSCI E-207
Lecture 9: Context-Free Grammars

Harry Lewis

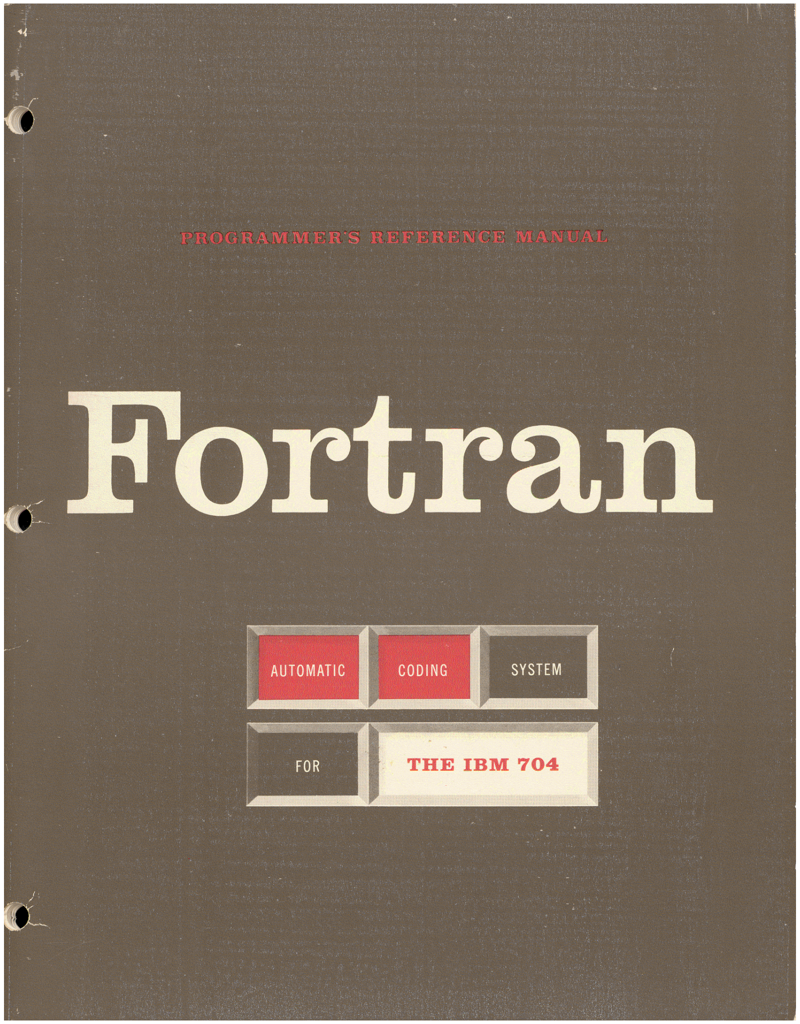
September 30, 20010

Reading: Sipser, §2.1 (except Chomsky Normal Form).

FORTRAN

John Backus

The Fortran Automatic Coding System for the IBM 704 EDPM (October 15, 1956)



A Fortran Lexical Definition

Functions

As in the above example, a FORTRAN expression may include the name of a *function* (e.g. the sine function SINF), provided that the routine for evaluating the function is either built into FORTRAN or is accessible to it as a pre-written subroutine in 704 language on the master FORTRAN tape.

GENERAL FORM	EXAMPLES
The name of the function is 4 to 7 alphabetic or numeric characters (not special characters), of which the last must be F and the first must be alphabetic. Also, the first must be X if and only if the value of the function is to be fixed point. The name of the function is followed by parentheses enclosing the arguments (which may be expressions), separated by commas.	SINF(A+B) SOMEF(X,Y) SQRTF(SINF(A)) XTANF(3.*X)

A Fortran Syntactic Definition

Formal Rules for Forming Expressions. By repeated use of the following rules, all permissible expressions may be derived.

1. Any fixed point (floating point) constant, variable, or subscripted variable is an expression of the same mode. Thus 3 and I are fixed point expressions, and ALPHA and A(I,J,K) are floating point expressions.
2. If SOMEF is some function of n variables, and if E, F, ... , H are a set of n expressions of the correct modes for SOMEF, then SOMEF (E, F, ... , H) is an expression of the same mode as SOMEF.
3. If E is an expression, and if its first character is not + or –, then +E and –E are expressions of the same mode as E. Thus –A is an expression, but +–A is not.
4. If E is an expression, then (E) is an expression of the same mode as E. Thus (A), ((A)), (((A))), etc. are expressions.
5. If E and F are expressions of the same mode, and if the first character of F is not + or –, then

   E + F
   E – F
   E * F
   E / F

   are expressions of the same mode. Thus A–+B and A/+B are not expressions. The characters +, –, *, and / denote addition, subtraction, multiplication, and division.

Peter Naur

Revised Report on the Algorithmic Language Algol 60 (1962)

1.1 Formalism for syntactic description.

The syntax will be described with the aid of metalinguistic formulae (1).

(1) Cf. J. W. Backus, The syntax and semantics of the proposed international algebraic language of the Zuerich ACM-GRAMM conference. ICIP Paris, June 1959.

Their interpretation is best explained by an example:

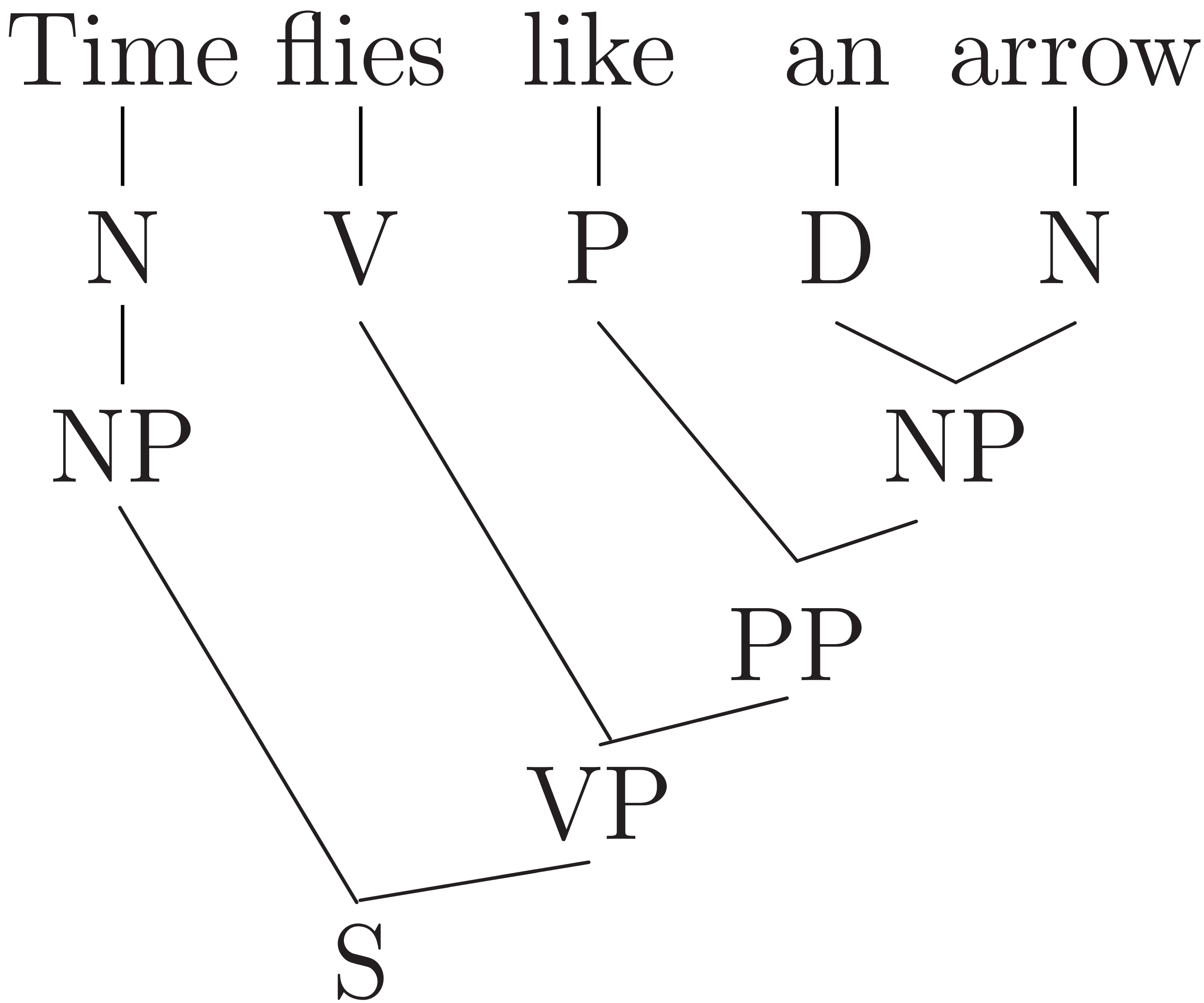
```
<ab> ::= ( | [ | <ab> ( | <ab> <d>
```

Sequences of characters enclosed in the bracket <> represent metalinguistic variables whose values are sequences of symbols. The marks `::=` and | (the latter with the meaning of **or**) are metalinguistic connectives. Any mark in a formula, which is not a variable or a connective, denotes itself (or the class of marks which are similar to it). Juxta position of marks and/or variables in a formula signifies juxtaposition of the sequences denoted. Thus the formula above gives a recursive rule for the formation of values of the variable <ab>. It indicates that `<ab>` may have the value `(` or `[` or that given some legitimate value of `<ab>`, another may be formed by following it with the character `(` or by following it with some value of the variable `<d>`. If the values of `<d>` are the decimal digits, some values of `<ab>` are:

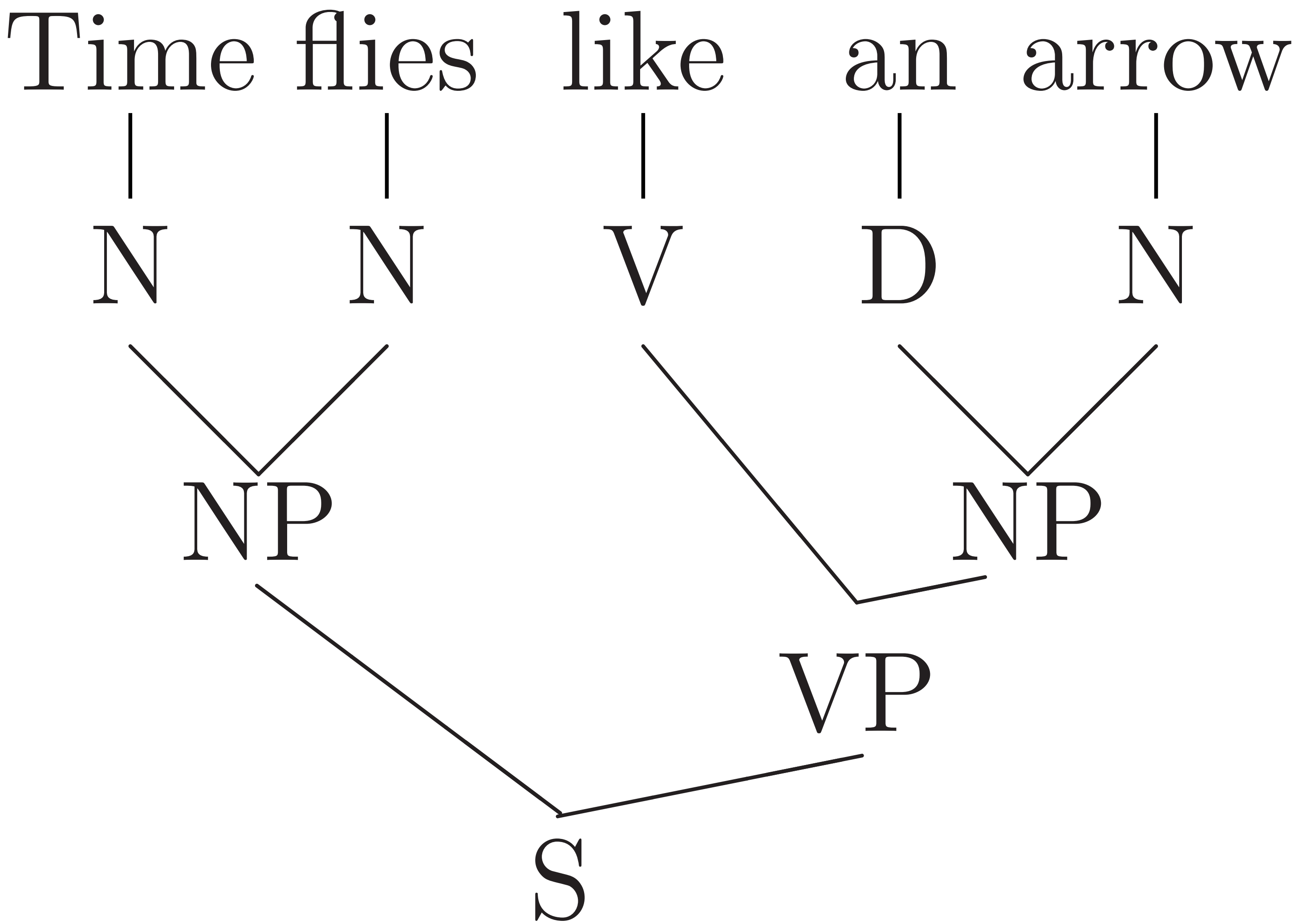
```
[(((1(37(
(12345(
(((
[86
```

Noam Chomsky

Parse Trees



Parse Trees



Context-Free Grammars

- Originated as abstract model for:
  - Structure of natural languages (Chomsky)
  - Syntactic specification of programming languages (Backus-Naur Form)
- A context-free grammar is a set of generative rules for strings

  e.g.

$$G \quad = \quad \begin{array}{l} S \rightarrow aSb \\ S \rightarrow \varepsilon \end{array}$$

- A derivation looks like:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$$

$$L(G) = \{\varepsilon, ab, aabb, \ldots\} = \{a^n b^n : n \geq 0\}$$

Equivalent Formalisms

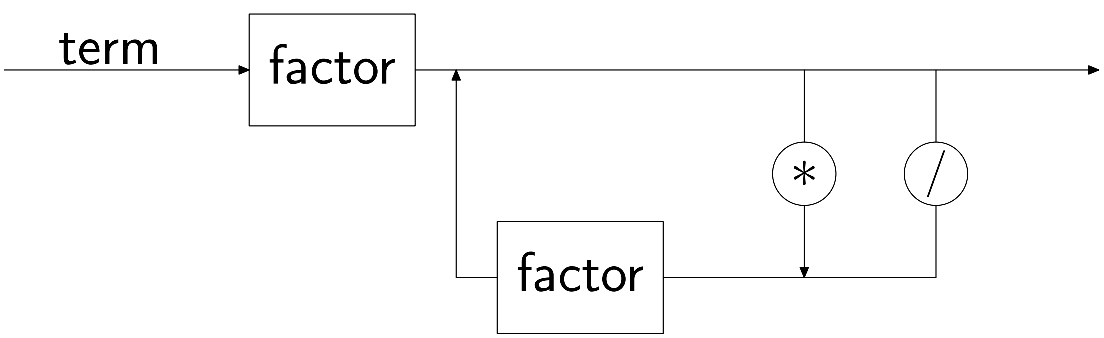
1. Backus-Naur Form (aka BNF, Backus Normal Form)

   due to John Backus and Peter Naur

$$
\begin{aligned}
< \text{term} > \; ::= \; < \text{factor} > \; & | \; < \text{factor} > * < \text{term} > \\
& | \; < \text{factor} > / < \text{term} >
\end{aligned}
$$

   "$|$" means "or" in the metalanguage $=$ same left-hand side

2. "Railroad Diagrams"



Formal Definitions for CFGs

- A CFG $G = (V, \Sigma, R, S)$

  $V =$ Finite set of variables (or nonterminals)

  $\Sigma =$ The alphabet, a finite set of terminals ($V \cap \Sigma = \emptyset$).

  $R =$ A finite set of rules, each of the form $A \to w$ for $A \in V$ and $w \in (V \cup \Sigma)^*$.

  $S =$ The start variable, a member of V

  e.g. $(\{S\}, \{a, b\}, \{S \to aSb, S \to \varepsilon\}, S)$

Formal Definitions for CFGs

- A CFG $G = (V, \Sigma, R, S)$

  $V =$ Finite set of <u>variables</u> (or <u>nonterminals</u>)

  $\Sigma =$ The alphabet, a finite set of <u>terminals</u> ($V \cap \Sigma = \emptyset$).

  $R =$ A finite set of <u>rules</u>, each of the form $A \Rightarrow w$ for $A \in V$ and $w \in (V \cup \Sigma)^*$.

  $S =$ The <u>start variable</u>, a member of V

  e.g. $(\{S\}, \{a, b\}, \{S \to aSb, S \to \varepsilon\}, S)$

- **Derivations**: For $\alpha, \beta \in (V \cup \Sigma)^*$ (strings of terminals and nonterminals),

  $\alpha \Rightarrow_G \beta$ if $\alpha = uAv, \beta = uwv$ for some $u, v \in (V \cup \Sigma)^*$ and rule $A \to w$.

  $\alpha \Rightarrow_G^* \beta$ ("α <u>yields</u> β") if there is a sequence $\alpha_0, \ldots, \alpha_k$ for $k \geq 0$ such that $\alpha_0 = \alpha$, $\alpha_k = \beta$, and $\alpha_{i-1} \Rightarrow_G \alpha_i$ for each $i = 1, \ldots, k$.

- $L(G) = \{w \in \Sigma^* : S \Rightarrow_G^* w\}$ (strings of terminals only!)

More examples of CFGs

- Arithmetic Expressions

G_1:

$$E \rightarrow x \mid y \mid E * E \mid E + E \mid (E)$$

G_2:

$$\begin{aligned} E &\rightarrow T \mid E + T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid x \mid y \end{aligned}$$

Q: Which is “preferable”? Why?

More examples of CFGs, cont.

- $L = \{x \in \{(,)\}^* :$ parentheses in x are properly ‘balanced’$\}$.

- $L = \{x \in \{a,b\}^* : x$ has the same # of a’s and b’s$\}$.

Parse Trees

Derivations in a CFG can be represented by parse trees.

Examples:

Each parse tree corresponds to many derivations, but has unique leftmost derivation.

Parsing

Parsing: Given $x \in L(G)$, produce a parse tree for x. (Used to 'interpret' x. Compilers parse, rather than merely recognize, so they can assign semantics to expressions in the source language.)

Ambiguity: A grammar is <u>ambiguous</u> if some string has two parse trees.
Example:

Context-free Grammars and Automata

What is the fourth term in the analogy:

Regular Languages : Finite Automata

as

Context-free Languages : ???

Regular Grammars

Hint: There is a special kind of CFGs, the **regular grammars**, that generate exactly the regular languages.

A CFG is **(right-)regular** if any occurrence of a nonterminal on the RHS of a rule is as the rightmost symbol.

Turning a DFA into an equivalent Regular Grammar

- Variables are states.
- Transition $\delta(P, \sigma) = R$ $\quad P \xrightarrow{\sigma} R$

  becomes $P \to \sigma R$
- If P is accepting, add rule $P \to \varepsilon$

  Example: $\{x : x$ has an even # of a's and an even # of b's$\}$

Other Direction: Omitted.