
# Formal Grammars

C6

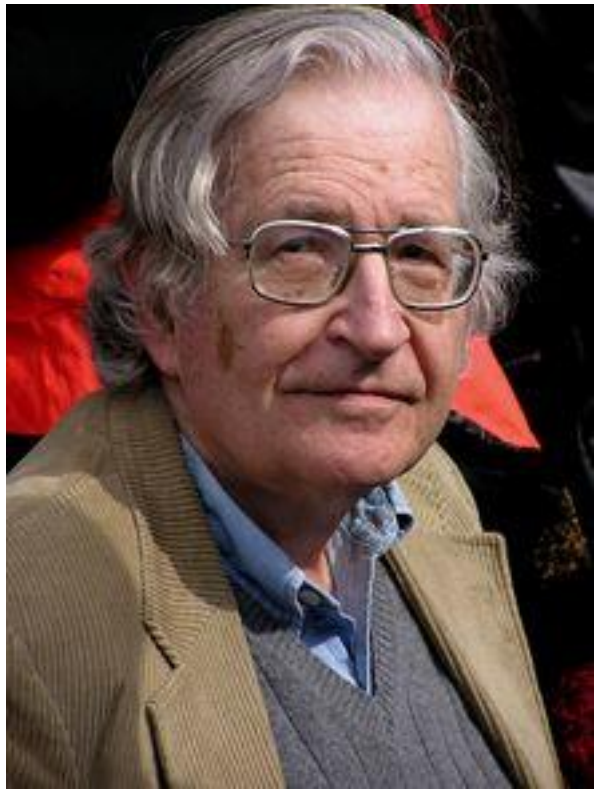
Grammars

EBNF

# Formal Grammars

Formal languages are distinguished from natural languages by their artificial construction (rather than natural emergence).

Noam Chomsky is often credited with opening the field of formal grammars while studying natural languages.



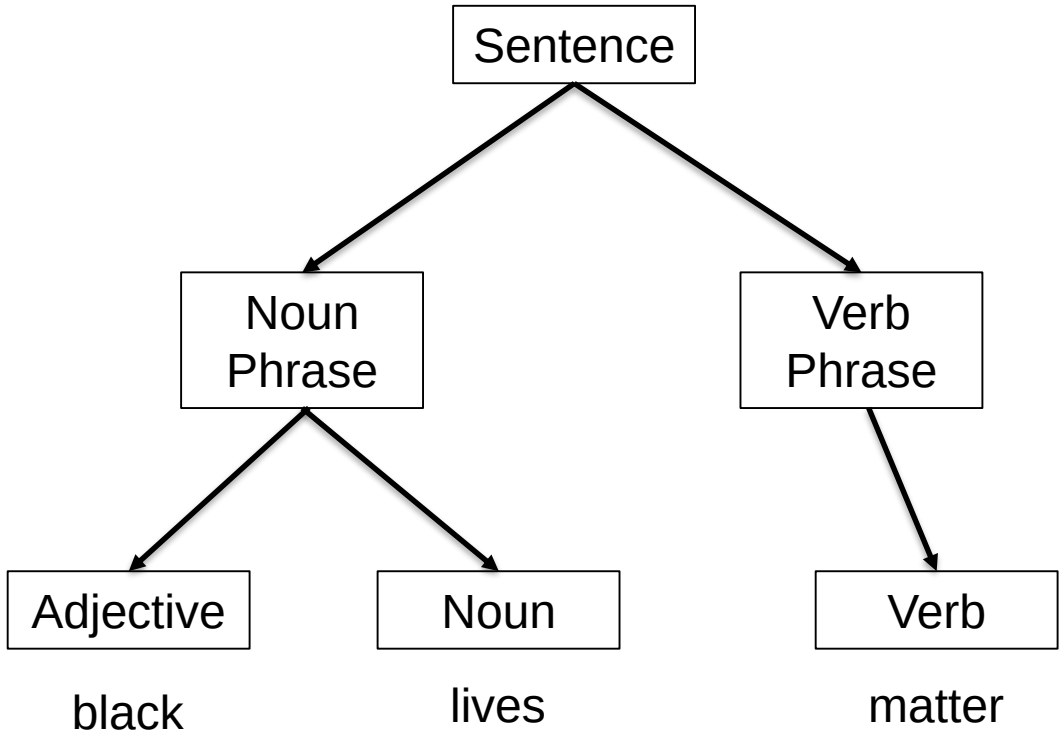
*Duncan Rawlinson (Creative Commons)*

Noam Chomsky

# Generative Grammars

Sentence = Noun Phrase, Verb Phrase, [Noun Phrase];
Noun = signs, directions, lives
Article = the
Verb = show, matter, look
Adjective = big, small, white, black
Noun Phrase = [Article], [Adjective], Noun | Noun Phrase;
Verb Phrase = Verb, [Noun Phrase];

The signs show the directions.
Small big directions matter the black white signs.




Syntactically correct productions (sentences) don't always convey meaning!
E.g. "I tested positively towards negative."

# Extended Backus-Naur Form

EBNF is a standard way of representing the syntax of a formal language (but *not* the semantics!)

- Terminal symbols
  - e.g. characters or strings
- Production rules
  - combinations of terminal symbols

*Robert McClure*

Niklaus Wirth

# Extended Backus-Naur Form

Very basic syntax of EBNF production rules:

- ‘=’ defines a production rule
- ‘|’ identifies alternates (e.g. ‘1’ | ‘2’ | ‘3’ )
- ‘{’, ‘}’ identify expressions that may occur zero or more times (e.g. ‘1’, { ‘0’ } )
- ‘[’, ‘]’ identify expressions that may occur zero or one time (e.g. ‘1’, [ ‘0’ ])
- ‘,’ identifies concatenation
- ‘-’ identifies exceptions
- ‘(’, ‘)’ identify groups
- ‘;’ terminates a production rule

# Example EBNF grammar

```
PROGRAM DEMO1
BEGIN
  A0:=3;
  B:=45;
  H:=-100023;
  C:=A;
  D123:=B34A;
  BABOON:=GIRAFFE;
  TEXT:="Hello world!";
END.
```

```
(* a simple program syntax in EBNF − Wikipedia *)
program = 'PROGRAM', white space, identifier, white space,
          'BEGIN', white space,
          { assignment, ";", white space },
          'END.' ;
identifier = alphabetic character, { alphabetic character | digit } ;
number = [ "-" ], digit, { digit } ;
string = '"' , { all characters − '"' }, '"' ;
assignment = identifier , ":=" , ( number | identifier | string ) ;
alphabetic character = "A" | "B" | "C" | "D" | "E" | "F" | "G"
                     | "H" | "I" | "J" | "K" | "L" | "M" | "N"
                     | "O" | "P" | "Q" | "R" | "S" | "T" | "U"
                     | "V" | "W" | "X" | "Y" | "Z" ;
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9" ;
white space = ? white space characters ? ;
all characters = ? all visible characters ? ;
```

# Simple EBNF Grammars

Grammar for arrangement of characters that are:

- Natural numbers?

```
natural = '0' | (nzdigit, { digit }) ;
nzdigit = '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' ;
digit = '0' | nzdigit ;
```

- Integers?

```
integer = '0' | (['-'], nzdigit, { digit }) ;
```

- Decimal numbers?

```
real = (['-'], natural, [('.' { digit }, nzdigit)]) - '-0' ;
```

- 24hr time, digital clock?

```
time = hour, ':', min ;
hour = ( ( '0' | '1' ) , digit ) | ( '2' , ( '0' | '1' | '2' | '3')) ;
min = ( '0' | '1' | '2' | '3' | '4' | '5' ), digit ;
```