https://pixabay.com/photos/tongyeong-nature-landscape-bench-2384216/

# Defining a Language

- There are 3 ways to define a language
  - Set Definitions
  - Decision Programs
  - Grammars

# Grammars

- A set of rules for defining which strings are grammatical and which are not.
- For grammatical strings it prescribes the structure of the string.
- Grammars do not give us the meaning of a sentence

# Uses of Grammars in Computer Science

- To process human language by computer
- To process computer languages (compilers)
- A general way of specifying an input format and as an aid to processing the input data
- Examining theoretical questions (grammar equivalence and grammar complexity)

# Examples from English

*The doctor hates the dalek.*

*The dalek hates the doctor.*

- These are grammatical, but have different meanings
- The words *the doctor* and *the dalek* form groups of words (noun phrases) – sub-structures inside the overall structure

# Examples from English

- The same words jumbled up are not grammatical:

  *Hates the doctor dalek the*

- Some strings become grammatical as part of a larger string:

  *The dalek the doctor hates*

  *The dalek the doctor hates hates the tin dog*

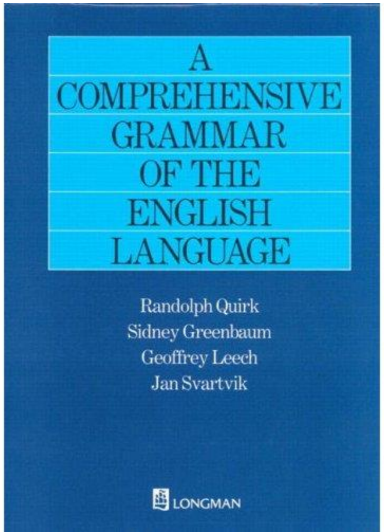
# “Word” Order

- The order of “words” is very important
- It is how we distinguish the *subject* from the *object* of a verb in English
- This is also true of programming languages
  - if ( a > b ) x = y ;
  - if ( ) > a b y x = ;
  - x = y ; has a different meaning to y = x ;

aside:
in other (natural) languages there may be inflections (changes in the word, usually at the end) to indicate the function, but English largely relies on word order

# Specifying an English Grammar

- “A Comprehensive Grammar of the English Language” by Quirk, Greenbaum, Leech and Svartvik (Longman 1985)
  - 1779 pages long
  - Incomplete
  - Describes the grammar in English
- Too complicated and ambiguous for a computer



# Specifying a Formal English Grammar

- We need a more precise and unambiguous form in which to specify the grammar
- This would allow us to automatically reason and process with it

# A formal grammar …

- defines which strings are grammatical and which are not
- for the grammatical strings it prescribes the structure of the string

# An Example Toy English Grammar

***R1*** - a SENTENCE could be a NOUN-PHRASE followed by a VERB-PHRASE

***R2*** - a NOUN-PHRASE could be an ARTICLE followed by a NOUN

***R3*** - (alternatively) a NOUN-PHRASE could be an ARTICLE followed by an ADJECTIVE followed by a NOUN

***R4*** - a VERB-PHRASE could be a VERB followed by a NOUN-PHRASE

***R5*** - (alternatively) a VERB-PHRASE could be just a VERB

*where, for example,*

***R6*** - an ARTICLE could be *a* or *the*

***R7*** - a NOUN could be *man*, *doctor*, *dalek*, *boy* or *dog*

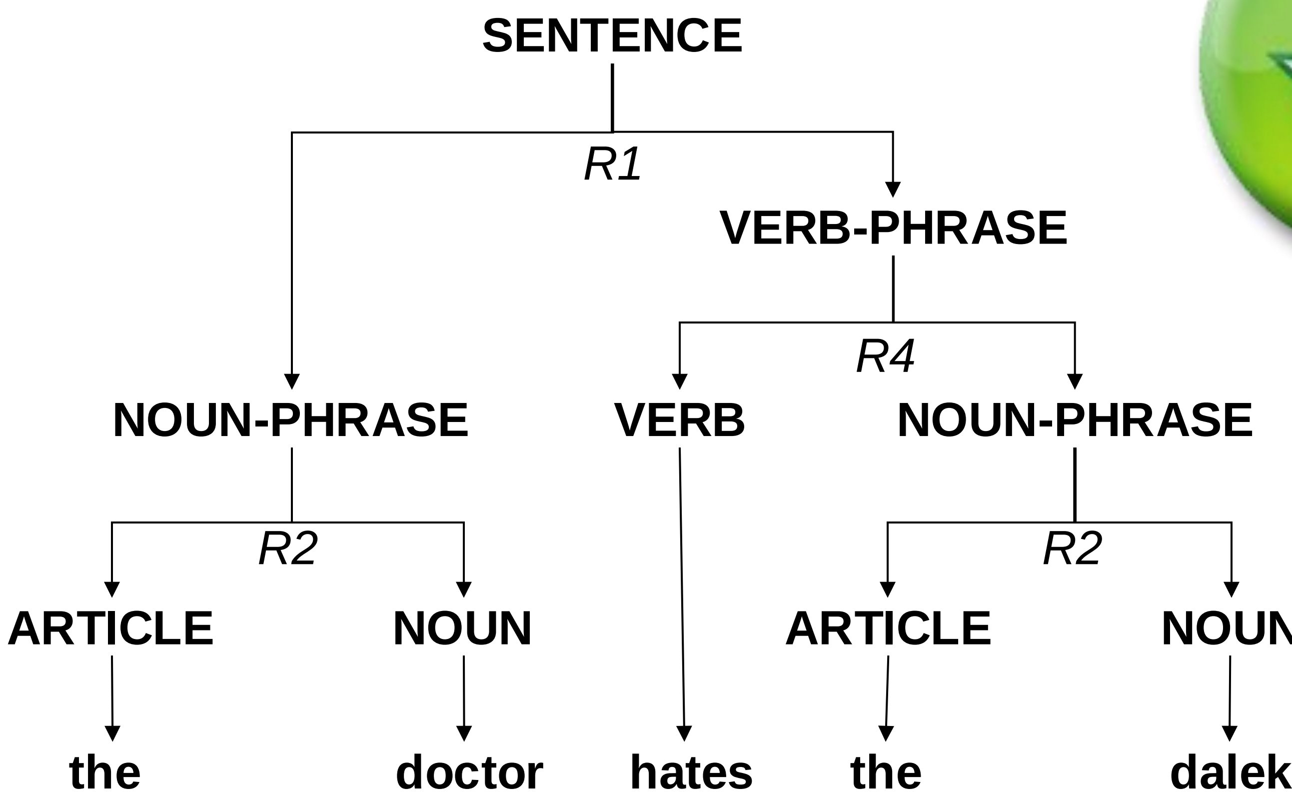
***R8*** - an ADJECTIVE could be *big*, *small* or *red*

***R9*** - a VERB could be *hates, likes* or *bites*

# Using the Toy English Grammar

| | |
|---|---|
| | SENTENCE |
| use R1 | NOUN-PHRASE VERB-PHRASE |
| use R2 | ARTICLE NOUN VERB-PHRASE |
| use R6 | the NOUN VERB-PHRASE |
| use R7 | the doctor VERB-PHRASE |
| use R4 | the doctor VERB NOUN-PHRASE |
| use R9 | the doctor hates NOUN-PHRASE |
| use R2 | the doctor hates ARTICLE NOUN |
| use R6 | the doctor hates the NOUN |
| use R7 | the doctor hates the dalek |

# Represented as a Tree



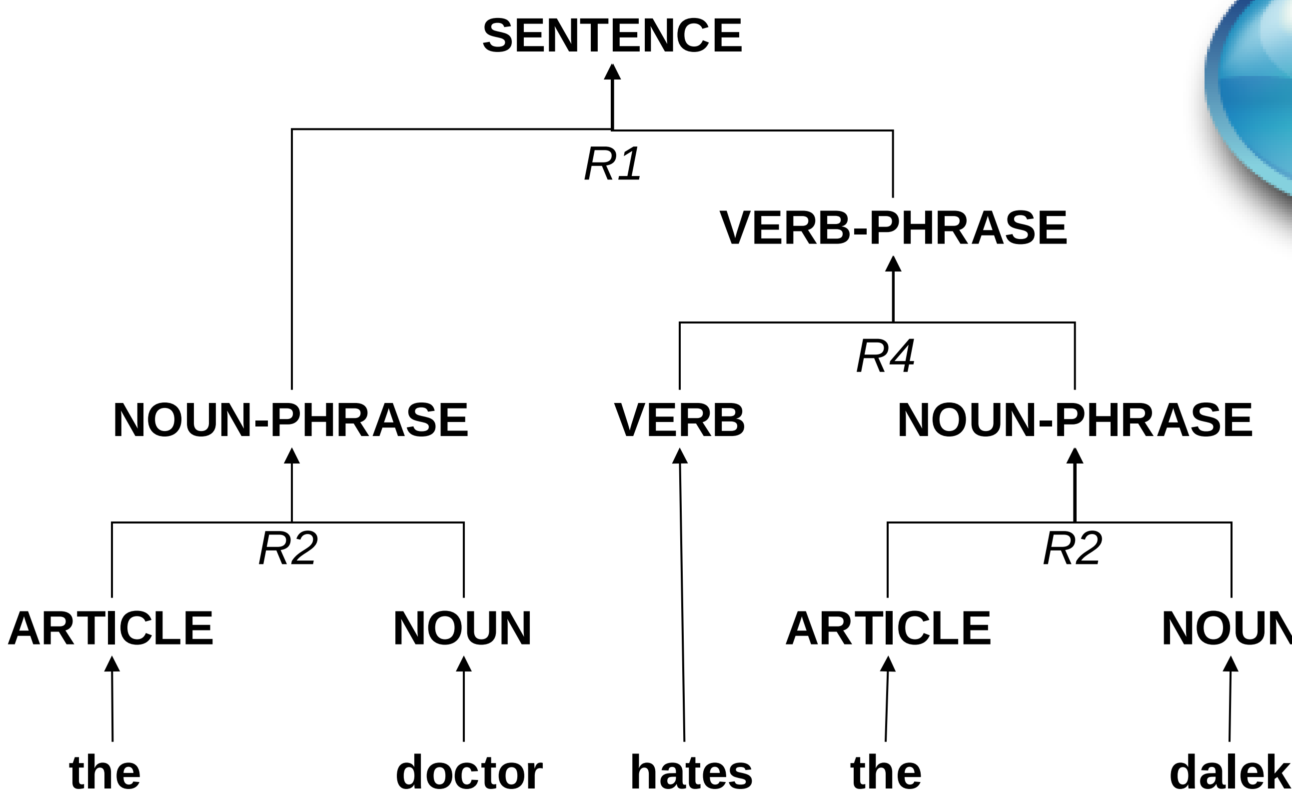

# Derivation

- Start with the first symbol
- Searches for a sequence in the current string that matches the left hand side of a rule.
- Replaces the sequence with the right hand side of the rule.
- Continues until it gets stuck or has a valid string (grammatically correct)

# Derivation

- Trying all possible derivations from the first symbol will give us all the grammatical strings of the language.
- A string that cannot be derived from the first symbol is ungrammatical (according to the particular grammar)
- top-down, depth-first search

# Represented again as a Tree




# Parsing

- This is more ‘interesting’ than derivation
- Given a string, what productions would we need to derive it from the first symbol?
- This is the basic principle of compilers
- Bottom-up

# Derivation and Parse Trees

- There are many different orders in which the rules of a grammar can be applied
- Do not always give the same tree diagram
- We will see examples where different derivations have different trees and the problems this might cause

# Using the Toy English Grammar

- According to the grammar the string *the doctor hates the dalek* can be derived from SENTENCE.
- The grammar can also derive *the dalek hates the doctor*, or *a big man likes the red dog*
- The grammar cannot derive *hates the boy man*, or *the man likes the cat*. These are not sentences of the language.

# Toy English Grammar Limitations

- No transitivity
  - some verbs like *hates* require an object noun phrase, and some like *sleeps* do not
- No agreement
  - some singular and plural subject NOUN-PHRASEs go with different forms of the verb
- The grammar can be improved by adding more rules to take these into account.

# Improvements (1)

***R7*** - a NOUN could be *man, men, boy, boys, dog, dogs*

***R9*** - a VERB could be *hate, hates, like, likes, bite, bites*

- this would allow
  - the man hates the dog
  - the men hate the dog
- but also
  - the man hate the dog
  - the men hates the dog

# Modifying for agreement (2)

- we keep rules R1 to R5 and drop R6 to R9, which we replace with:

***R6*** - you can rewrite NOUN followed by VERB as SINGULAR-NOUN followed by S-FORM-OF-VERB

***R7*** - (alternatively) you can rewrite NOUN followed by VERB as PLURAL-NOUN followed by SIMPLE-FORM-OF-VERB

***R8*** - an ARTICLE could be *a* or *the*

***R9*** - a NOUN could be a SINGULAR-NOUN

***R10*** - (alternatively) a NOUN could be a PLURAL-NOUN

***R11*** - a SINGULAR-NOUN could be *man, boy* or *dog*

***R12*** - a PLURAL-NOUN could be *men, boys* or *dogs*

***R13*** - an ADJECTIVE could be *big*, *small* or *red*

***R14*** - an S-FORM-OF-VERB could be *hates, likes* or *bites*

***R15*** - a SIMPLE-FORM-OF-VERB could be *hate, like* or *bite*

# Modifying (3)

- SENTENCE
- NOUN-PHRASE VERB-PHRASE (use R1)
- ARTICLE NOUN VERB-PHRASE (use R2)
- the NOUN VERB-PHRASE (use R8)
- the NOUN VERB NOUN-PHRASE use R4)
- the SINGULAR-NOUN S-FORM-OF-VERB NOUN-PHRASE (use R6)
- the man S-FORM-OF-VERB NOUN-PHRASE (use R11)
- the man hates NOUN-PHRASE (use R14)
- the man hates ARTICLE ADJECTIVE NOUN (use R3)
- the man hates a ADJECTIVE NOUN (use R8)
- the man hates a small NOUN (use R13)
- the man hates a small SINGULAR-NOUN (use R9)
- the man hates a small dog (use R11)

# Modifying (4)

- so we can derive
  - the man hates a small dog
- similarly we can derive
  - the men hate a small dog
- but not
  - the man hate a small dog
  - the men hates a small dog

# A Toy Grammar for Programming

***R1*** - an IF-STATEMENT could be the word **if** followed by a CONDITION followed by a STATEMENT

***R2*** - (alternatively) an IF-STATEMENT could be the word **if** followed by a CONDITION followed by a STATEMENT followed by the word **else** followed by a STATEMENT

***R3*** - a CONDITION could be a **(** followed by an identifier followed by a CONDITION-OPERATOR followed by an identifier followed by a **)**

***R4*** - (alternatively) a CONDITION could be ...

***R5*** - a CONDITION-OPERATOR could be == != < <= ...

***R6*** - a STATEMENT could be an ASSIGNMENT-STATEMENT or a FOR-STATEMENT or an IF-STATEMENT ...

# A Grammar for Programming

IF-STATEMENT

use R1 if CONDITION STATEMENT

use R3 if (identifier CONDITION-OPERATOR identifier) STATEMENT

use R5 if (identifier < identifier) STATEMENT

use R6 if (identifier < identifier) ASSIGNMENT-STATEMENT

… etc.

# Recursion in Programming

- The grammar is recursive
  - An IF-STATEMENT is defined partly in terms of IF-STATEMENTs
- There must be at least one non-recursive definition for an IF-STATEMENT so that we do not loop indefinitely.
- We also have recursion in natural languages

# Recursion in natural language

- a NL grammar could contain **recursive** rules:
  - a NOUN-PHRASE could be an ARTICLE followed by a NOUN followed by a PREPOSITION-PHRASE
  - a PREPOSITION-PHRASE could be a PREPOSITION followed by a NOUN-PHRASE
  - a PREPOSITION could be *of, with* or *in*
- of course, there must be at least one non-recursive definition for a NOUN-PHRASE, so that we do not recurse for ever

# Different Grammar Rules

- Most rules look like this: 
  - a NOUN-PHRASE could be an ARTICLE followed by a NOUN
- But you can also have rules like this: 
  - you can rewrite NOUN followed by VERB as SINGULAR-NOUN followed by the S-FORM-OF-VERB
- There are different grammars depending on how much we restrict the form of the rules.

# Phase Structure Grammars

Made up of 4 parts

Pay attention: definition coming up!

# Phrase Structure Grammar (1)

1. a set of basic objects of the language
   - characters, atomic symbols
   - words (natural languages)
   - reserved words like ‘if’, identifiers and numbers (programming languages)
   - These are called the **terminals** of the grammar, the **vocabulary** or the **alphabet**.

# Phrase Structure Grammar (2)

2. A set of things like NOUN-PHRASE or CONDITION
   – We use these in our definition of the structure of a valid sentence, but they do not appear in the actual string.
   – These are **non-terminals**
   – Always written as capitals

# Phrase Structure Grammar (3)

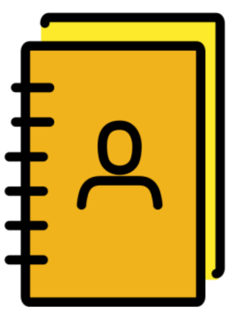
3. Contains a particular non-terminal with which we always start the derivation of a valid sentence
   - e.g. SENTENCE or IF-STATEMENT
   - This is the **start symbol**. It may also be called the **distinguished** or **sentence** symbol.

# Phrase Structure Grammar (4)

4. A set of **productions** or **rules** of the grammar. They may have the form:
    - STRING1 can be written as STRING2
    - STRING1::= STRING2
    - STRING1$\rightarrow$ STRING2
  - STRING1 and STRING2 are sequences of one or more terminals and/or non-terminals
  - STRING2 could be an empty string

# Contacts, Address Database, or Telephone Directory Grammar

- **terminals**: a, b, ... z, A, B, ... Z, 0, 1 ... 9
- **non-terminals**: ENTRY, PERSON-NAME, ADDRESS, TELEPHONE-NUMBER, SURNAME, FORENAME-LIST, FORENAME, PROPER-FORENAME, INITIAL, NUMBER, STREET-NAME, AREA-NUMBER, LOCAL-NUMBER, ...
- **start symbol**: ENTRY



# Contacts, Address Database, or Telephone Directory Grammar

- **productions**:

ENTRY $\rightarrow$ PERSON-NAME ADDRESS TELEPHONE-NUMBER

PERSON-NAME $\rightarrow$ SURNAME FORENAME-LIST

FORENAME-LIST $\rightarrow$ FORENAME

FORENAME-LIST $\rightarrow$ FORENAME FORENAME-LIST

FORENAME $\rightarrow$ PROPER-FORENAME

FORENAME $\rightarrow$ INITIAL

ADDRESS $\rightarrow$ NUMBER STREET-NAME

TELEPHONE-NUMBER $\rightarrow$ AREA-NUMBER LOCAL-NUMBER

etc.

# Sentential Form

- Consider part of the derivation from earlier:
  - the man VERB NOUN-PHRASE
- This intermediate stage between the start symbol and the final sentence is called a **sentential form**
- It is “any string that can be derived in zero or more steps from the start symbol”
  - Can contain terminal and non-terminal symbols
- So a sentence is a sentential form, but a sentential form is not necessarily a sentence

# Backus-Naur Form (BNF)

- This is another form of notation for specifying a grammar
- The production is written as ::=
- Terminals are written as before
- Non-terminals are written in brackets – <...>
- Alternatives for the same non-terminal are separated by | (meaning ‘or’)

# BNF history lesson

History won't be in the exam!!

- **John Backus** created the notation in order to express the grammar of ALGOL.
  - At the first World Computer Congress, which took place in Paris in 1959, Backus presented "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM Conference", a formal description of the IAL which was later called ALGOL 58. The formal language he presented was based on Emil Post's production system. Generative grammars were an active subject of mathematical study, e.g. by Noam Chomsky, who was applying them to the grammar of natural language.

- **Peter Naur** later simplified Backus's notation to minimize the character set used, and, at the suggestion of **Donald Knuth**, his name was added in recognition of his contribution.

- [https://en.wikipedia.org/wiki/Backus-Naur_form](https://en.wikipedia.org/wiki/Backus-Naur_form)

# Backus-Naur Form Example 1/2

- <unsigned integer> ::= 0 | 1 | 2 | … | 8 | 9 | 1<digit sequence> | 2<digit sequence> | … | 8<digit sequence> | 9<digit sequence>
- <digit sequence> ::= 0 | 1 | 2 | … | 8 | 9 | 0<digit sequence> | 1<digit sequence> | 2<digit sequence> | … | 8<digit sequence> | 9<digit sequence>

# Backus-Naur Form Example 2/2

- The start symbol is <unsigned integer>
- This can be 0, 7, 2015 or any other number
- <unsigned integer> cannot start with 0 unless it is 0
  - Cannot be 015
- <digit sequence> is any sequence of one or more digits from 0 to 9

# Converting Rules To BNF

- Rules such as “an IF-STATEMENT could be the word **if** followed by a CONDITION followed by a STATEMENT” need to be converted into something a computer can understand
  - <IF-STATEMENT> ::= if <CONDITION><STATEMENT>
- This is how languages such as Pascal are specified.

# Another Conversion Example

- This rule…
  - you can rewrite NOUN followed by VERB as SINGULAR-NOUN followed by the S-FORM-OF-VERB
- Becomes…
  - <NOUN><VERB> ::= <SINGULAR-NOUN><S-FORM-OF-VERB>

# Summary and comments

Acknowledgements: Thanks to Roger Garside and Lynne Blair for
earlier versions of the slides and workbooks for weeks 11-15

# Formal Grammars

- The examples give precise recipes for defining how a string can be a grammatical sentence in the language as specified by the grammar
- A grammar specifies the **syntax** of a language
  - What the basic elements are and how they are put together into a structure

# Formal Grammars

- They do not specify the **semantics** of a sentence.
  - The meaning corresponding to a syntactically valid sentence
- These two sentences are grammatically different but have the same meaning
  - *The boy hits the dog*
  - *The dog is hit by the boy*

# Summary (week 11)

- Introduction to the elements of formal languages
- Definitions of basic elements of languages
  - Strings, sentences, alphabets, operations
- Different ways of defining languages
  - Set definitions, decision programs, grammars
- Introduction to phrase structure grammars

https://pixabay.com/photos/coffee-cake-cup-food-dessert-cafe-1197760/