# Introduction to Programming Languages

Syntax, Semantics and brief history of programming languages

Ayoub Ouarrak

11/06/2020

## Languages

## Languages and Alphabet

The concept of language is **very general**: A language $L$ is a set of strings over an alphabet $\Sigma$.

**Definition**

An alphabet $\Sigma$ is a finite set of symbols.

We call **symbol** a primitive, abstract not better defined entity. Examples: letter and digits, ideograms, gestures, emojis, ...

## Example Languages

Following examples of languages:

- {if, then, else, import, switch, for, ⋯ }
- {select, where, count, from, in, join, ⋯ }
- $\{|, *, +, -, ?, [\,, ]\,, \cdots\}$
- {𝅝, 𝅗𝅥, 𝅘𝅥𝅯, 𝄴, 𝄫, ♯, 𝄴$^{3}_{2}$, ⋯ }
- All complete sentences occuring in the tragedy of Macbeth
- Italian gestures

[0]"*" is called Kleene star, from Stephen Kleene who helped formalise regular expressions

## Formal and Natural Languages

- **Formal** means precisely defined, both in term of **Syntax** and **Semantics**.
- Contrast this with natural languages used in Nature: Syntax is relatively free and the Semantics depend subjective interpretation.
- **Language** should be interpreted in its broadest sense: there are formal visual languages and natural body languages; sounds can be used to define formal languages as well as to compose music.

Now, how do we specify a formal language?

## Grammars

## Syntax: ”How is defined”

Syntax of Formal languages can be defined in terms of Grammars (like natural languages). These are called *generative grammars*, formalised by Noam Chomsky in the 50’ and he classified them into types now known as the **Chomsky hierarchy**.

Different types of grammars generate languages with different expressivity power.

## Grammars: Chomsky Hierarchy

| Grammar | Language | Example |
|---|---|---|
| Type-0 | Unrestricted | Subset of English |
| Type-1 | Context-Sensitive | C |
| Type-2 | Context-Free | Basic SQL |
| Type-3 | Regular | Regex |

Languages described by Type-0 grammars are the most expressive, compared to Type-3 which are the less expressive.

[1]Regular $\subset$ *ContextFree* $\subset$ *ContextSensitive* $\subset$ *Unrestricted*
[2]The parsing phase of a compiler uses Context-Free grammars to build the Parse Tree

## Grammars: Expressivity power

*Can we write a regular expression that match the minimum number in a finite array of integers (represented as string)?*

[3]Finding the minimum requires the ability to have "knowledge of the past" (store infos and compare) and this is possible only from Type-2 grammars

## Example Grammar: Boolean expressions

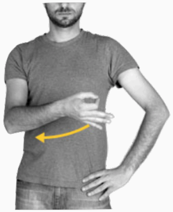
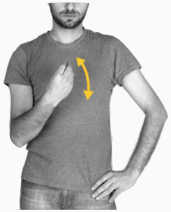
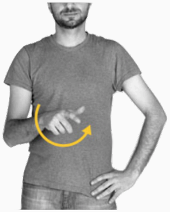
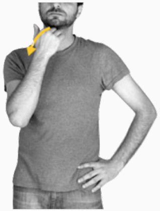
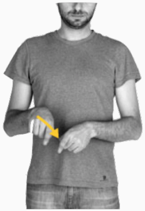
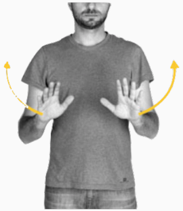
The follwing an example of grammar that describe the basic boolean expressions.

$$\begin{aligned} E &\Rightarrow \mathbf{0} \\ &\Rightarrow \mathbf{1} \\ &\Rightarrow (\text{not } E) \\ &\Rightarrow (E \text{ or } E) \\ &\Rightarrow (E \text{ and } E) \end{aligned} \tag{1}$$

Grammars of more complex languages:

- Grammar of Java 8
- Grammar of Javascript
- A dialect of SQL

## Example Syntax: Italian gestures

## Semantics

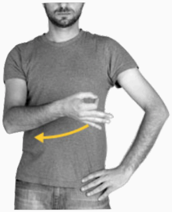
## Semantics: ”What does it means”

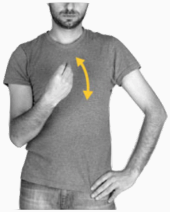
So we have defined how we can describe the syntax of a formal language through a grammar, but a grammar say nothing about the meaning.

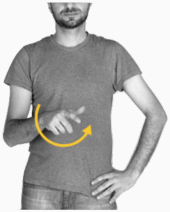
**Definition**

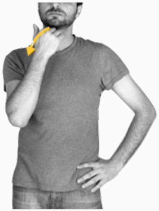
**Semantics** reveals the meaning of syntactically valid strings in a language. For natural languages, this means correlating sentences and phrases with objects, thoughts and feelings based on our experiencse. For programming languages semantics describe a behaviour that a computer follows when executing a program in the language.

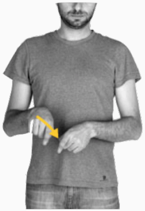
## Example informal Semantics: Italian gestures

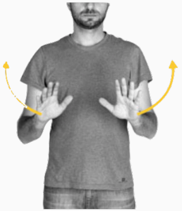

Perfect!


What in God's name are you saying?


Nothing.


I don't care.


Those two get along.


It wasn't me or I don't know.

## Translational Semantics

**Definition**

The semantics of a programming language $L$ can be preserved when the language is translated into another form, called target language $L_t$.

[4] In other words, this definition tell us that a program written in **C**, when compiled into **Assembly**, the meaning of the program will remain the same.
Note: the compiler can make changes on the structure (loop -> unroll, switch -> if else)

## Type system

Semantics give us the meaning of the operations of a Language.
For example in Java, the semantic of **+** between Strings is not addition, but concatenation (In Javascript God only knows)

The type system instead tell us what's the expexted behaviour when semantics rules are applied on a type.
Example of operation where the result depends on the type system of the language

$$1 + "1"$$

## Static and Dynamic typing

- **Static typing:** all expressions have their types before the program is executed, typically at compile-time.
- **Dynamic typing:** determines the type-safety of operations at run time; in other words, types are associated with run-time values rather than textual expressions

## Strong and Weak typing

- **Weak typing:** allows a value of one type to be treated as another, e.g. treating a string as a number.
- **Strong typing:** restrictions are applied on conversions between types. An attempt to perform an operation on the wrong type of value raises an error. Strongly typed languages are often termed **type-safe or safe**.

## Programming languages

## A universal programming language?

*Why there are so many programming languages?*

*One universal language is not enough?*

[5] There are more than 1000 programming languages

## A universal programming language?

For the same reason why there are multiple natural languages; multiple programming languages has been created to introduce new ideas and new ways of thinking.

With ultimate goal to describe toughts and instruct machines with a language that is the most natural possible.

## Abstraction of reality

**Definition**

To increase expressivity, programming languages introduce new **abstractions** of physical concepts.

## Timeline of Programming Languages: late 50s

- **Assembly** (low level stuff)
    - Naming abstraction
- **Fortran**: (Formula Translation)
    - the concept of Enviroment
    - basic DO loops
- **Algol** (Algorithmic Language)
    - Block structure
    - Recursive procedures

## Timeline of Programming Languages: 60s

- **Cobol** (common business-oriented language)
  - English like language
  - Files
  - Comments
- **Lisp** (LISt Processor)
  - Garbage collection
  - AI
  - Father and Mother of Functional languages
- **Simula** (Simulation)
  - Classes
  - Inheritance
  - Object-oriented Programming

## Timeline of Programming Languages: 70s

- **Pascal**
  - Enforcement of programming discipline
- **Prolog** (Programmation en logique)
  - AI
  - Logic Paradigm
- **C**
  - Is everywhere, and influenced everything
- **Ada** (Ada lovelace)
  - Generic types
  - For his safety standard is used by US Department of Defense

[7] Ada lovelace daughter of Lord Byron, she's considered the first programmer

## Timeline of Programming Languages: 80s

- **Smalltalk**
  - Fully OO
  - Father of Object-C (Apple favourite language)
- **C++** (C with classes)
  - Operator Overloading

[7]This is also the period where the number of female programmers start to reduce drastically. Possibly related with the introduction of personal computers and first videogames that were advertised for a mostly Male audience

## Timeline of Programming Languages: 90s

- **Perl**
    - Scripting language
- **Python** (Monty Python)
    - Scripting language
    - Simple with introduction of indentation as structure
- **Java**
    - Based on C++

## Programming Languages Type systems

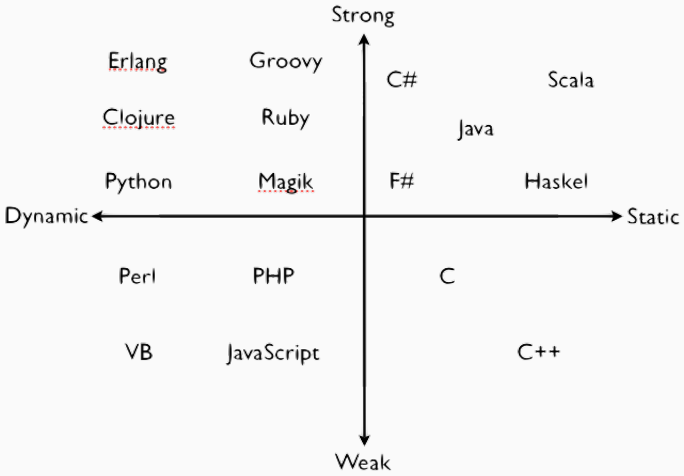


**Figure 1:**

## Esoteric programming languages

Not all programming languages introduce usefull features or more expressivity, some languages are created just as a ”joke” or to test the boundaries of computer programming language design.

## Esoteric programming languages: BrainFuck

*Created to duck your brain.*

**Example**

```
,[.[-],]
```

[8]The base for many esoteric programming languages

## Esoteric programming languages: Chiken

*The only valid word is, yes you guessed right: Chiken*

[9] The number of chickens corresponds to an opcode

## Esoteric programming languages: Folders

*A language where the code is written with folders. Perhaps the most Windows of languages.*

[10] The structure of the folders in the filesystem define the instructions

## Esoteric programming languages: Whitespace

*Only valid tokens are white-characters (space, tabs and newlines).*

**Example**

**Questions?**

## References

- Syntax and Semantics of Programming Languages
- Noam Chomsky - The Structure of Language
- Chomsky Hierarchy - Computerphile
- Lectures of Prof. R. Bagnara (University of Parma)
- Lexicon of Italian gestures