Concepts of Programming Languages

- Syllabus
- Introduction to Concepts of Programming Languages

What do we teaching

- We do not teach individual programming languages here
- We talk about the common features, or concepts, of programming languages

Programming Languages

- Purpose of a language?
  - Someone A uses a language to represent something (e.g., a feeling) B as C so that another person D can obtain B from C.
- Purpose of a programming language?
  - Someone A uses a programming language to write a program C for **problem B** so that the **computer** can follow C to output a **solution** of B.
- Properties of programming languages
  - For computer to understand they, they have to be precise. In other words, we have to mathematically define the syntax of language.
  - Semantics of a languages: in principle, can be defined also mathematically (but we don’t go far in this course)

Sources of Concepts in Programming Languages

- From computer (instructions of the CPU of the computer)
- From mathematics (a language used to solve problems)
- From natural languages (also used to solve problems)

Groups of Programming Languages

- Imperative languages (language constructs closer to computer instructions, but also borrow ideas from mathematics)
  - von Neumann (Fortran, Pascal, Basic, C)
  - object-oriented (Smalltalk, Eiffel, C++?)
  - scripting languages (Perl, Python, JavaScript, PHP)
- Declarative language (language constructs closer to mathematics/human language)
  - Functional (Scheme, ML, pure Lisp, FP)
  - Logic, Constraint-based (Prolog, VisiCalc, RPG)

Why Study Concepts of Programming Languages

- User’s perspective
    - There are numerous programming languages there. We are supposed to use some of them as a programmer
    - By learning common concepts of programming languages, we can learn new language quickly because a language can usually taken as a combination of set of concepts.
    - These concepts can help us decide which language is used for which problems.
- Language designer’s perspective
    - Learn the concepts so that new languages can be created to address new challenges to programming languages
    - Learn principles and ideas how a program can finally be executed by a CPU which knows only 0s and 1s.

Concepts of programming languages

- How to define a language precisely – Syntax (good for both users and designers)
- Meaning of a language (semantic analysis)
- Names (their uses and meaning) – special in programming language: manage the complexity (of a program in this language)
- Type system

- Imperative languages
  - Control abstraction (subroutine, function or method)
  - Control flow (corresponding to CPU instructions)
  - Data Abstraction
- Declarative languages
  - Functions
  - Lists
  - Relations
- Concurrency: a phenomenon in computing. What it is, and how language can be designed to represent concurrency effectively.
- Implementation of a programming language

Implementation

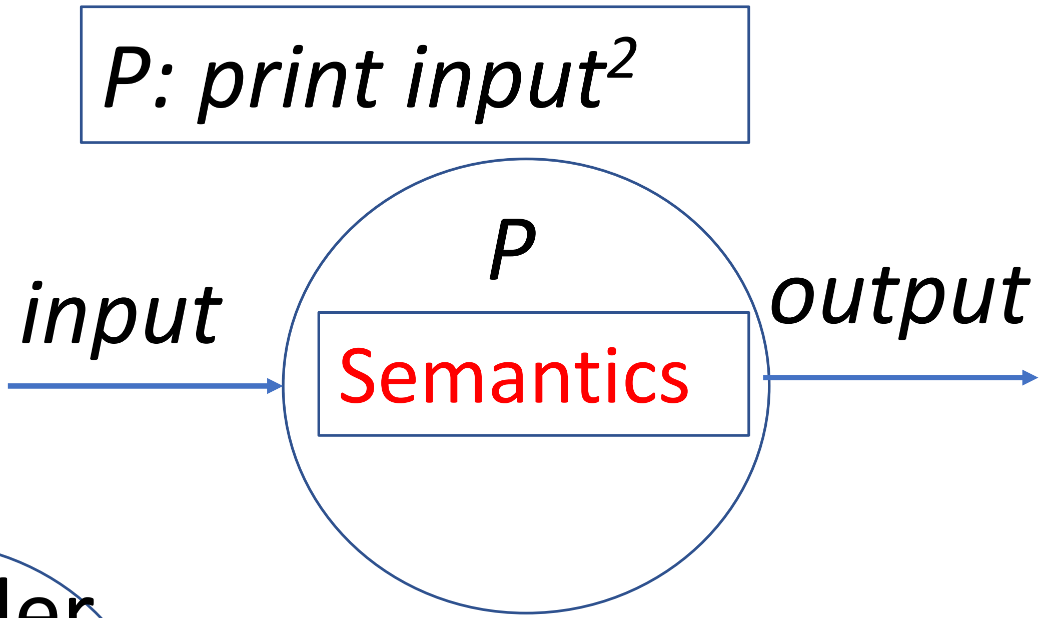
- Question: how to execute program *P* in a programming language by a computer? (A computer has its own language, i.e., the 0/1 language understandable/executable by CPU)
  - Idea 1: translate program *P* to a program P’ in another language (e.g., computer instructions) – **Compilation**

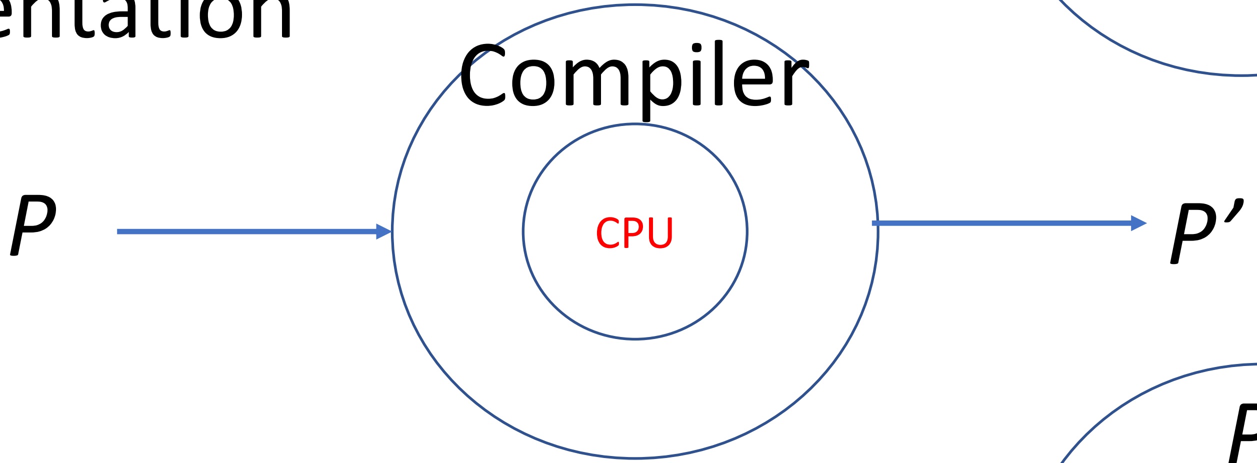
  - Idea 2: a program running on a computer will take *P* and output the result of the execution of *P* – **Interpretation**

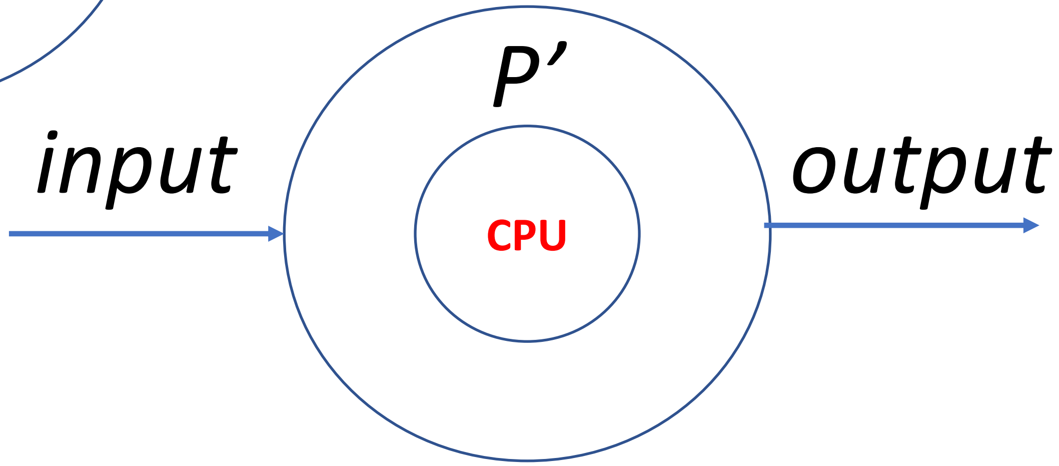
Compilation

Abstraction of a program

Implementation



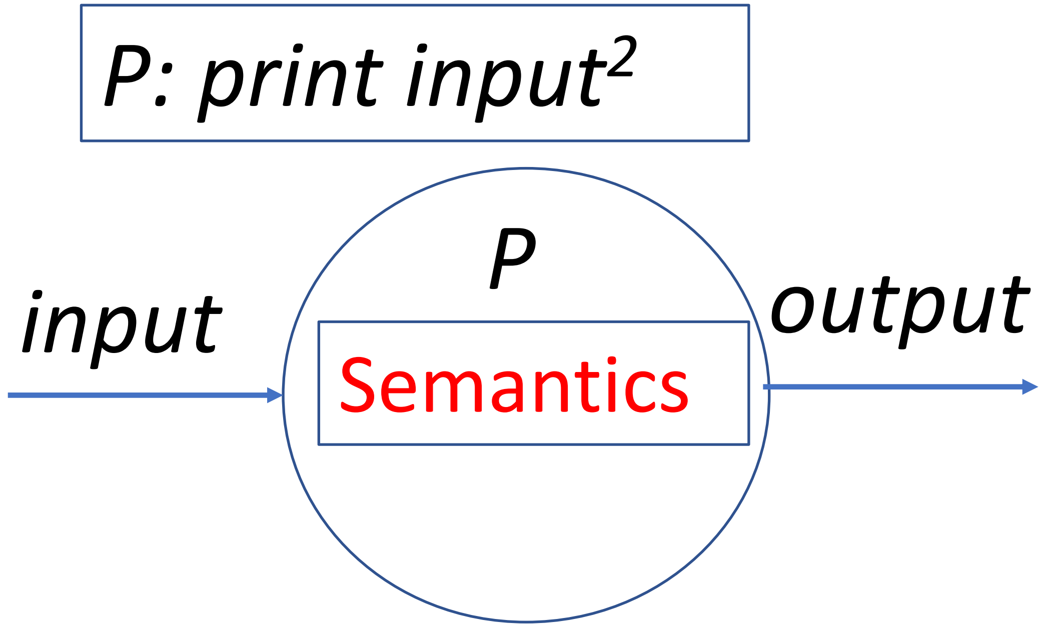


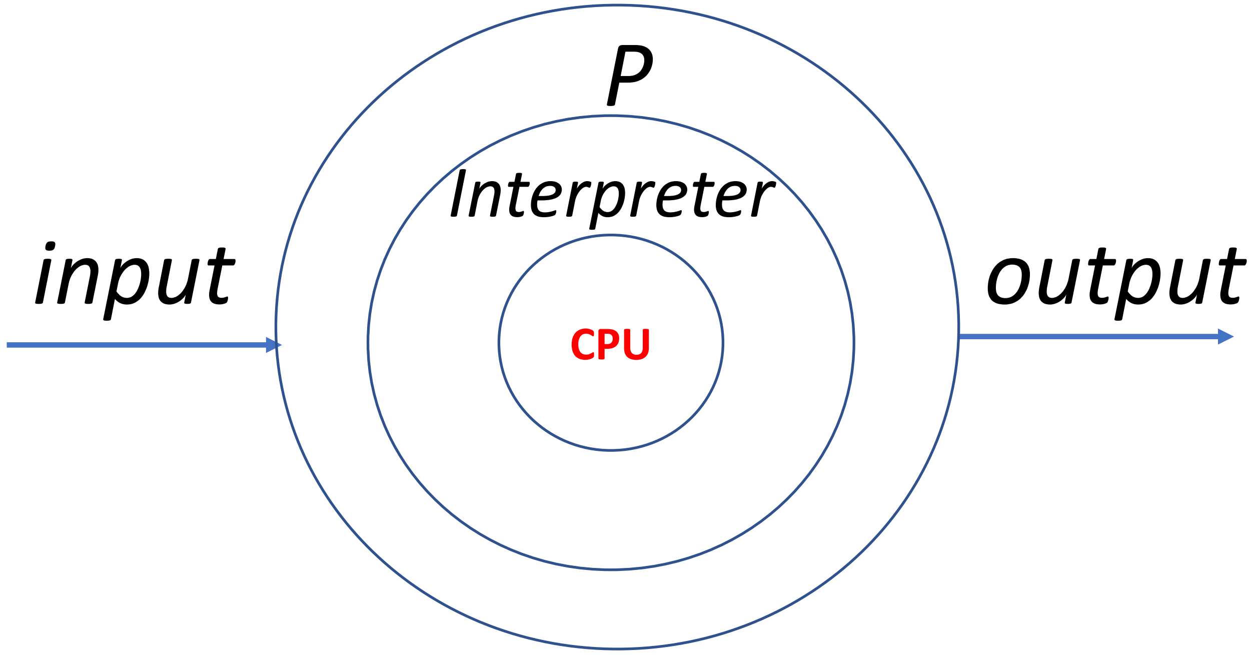


Interpretation

Abstraction of a program

Interpretation





Property comparison

- Compilation
  - More efficient
- Interpretation
  - Direct error message
- State of the art implementation – a mixture of compilation and interpretation
  - compilation followed by interpretation
  - Source language → virtual instructions (virtual machine) → intepretation