

Concepts of Programming Languages

- Syllabus
- Introduction to Concepts of Programming Languages

What do we teach

- 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.
 - Interestingly, programming languages are used from computers to solve problem, but understanding them well can enhance our problem-solving ability too.
- Properties of programming languages
 - For computers to understand a programming language, the language has to be precise. In other words,
 - The **syntax** of the language can be mathematically defined.
 - The **semantics** of the languages: in principle, can be defined also mathematically (but we don't go that far in this course)

Sources of Concepts in Programming Languages

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

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/natural language)
 - Functional (Scheme, ML, pure Lisp, FP)
 - Logic, Constraint-based (Prolog, VisiCalc, RPG)

Why Study Concepts of Programming Languages

- User (of the language, i.e., programmer) perspective
 - There are numerous programming languages there. We (users) are supposed to use some of them
 - 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 language concepts so that new languages can be created to address new challenges to programming languages
 - Learn principles and ideas how a program (in any language) 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
 - Data Abstraction
 - Control abstraction (subroutine, function or method)
 - Control flow (corresponding to CPU instructions)
- 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

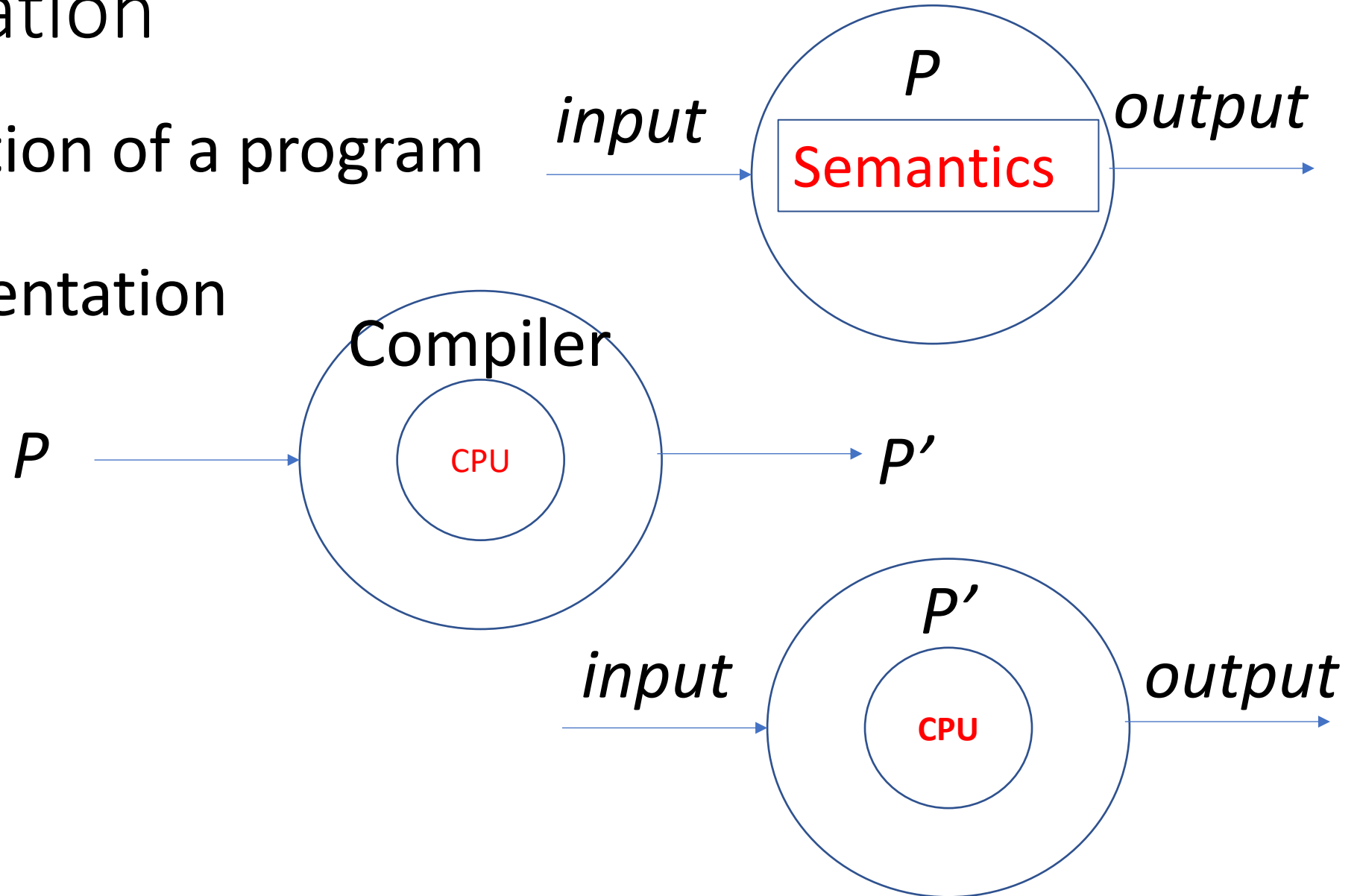
- Question: how to execute program P in a programming language by a computer? (A computer has its own language, i.e., the 0/1 language understandable/executable by CPU)
 - Idea 1: translate program P to a program P' in another language (e.g., a machine language which consists of computer instructions) – **Compilation**
 - Idea 2: a program I running on a computer will take P and output the result of the execution of P – **Interpretation** (the program I will interpret each statement of P)

Compilation

Abstraction of a program

Implementation

e.g., P : find input²

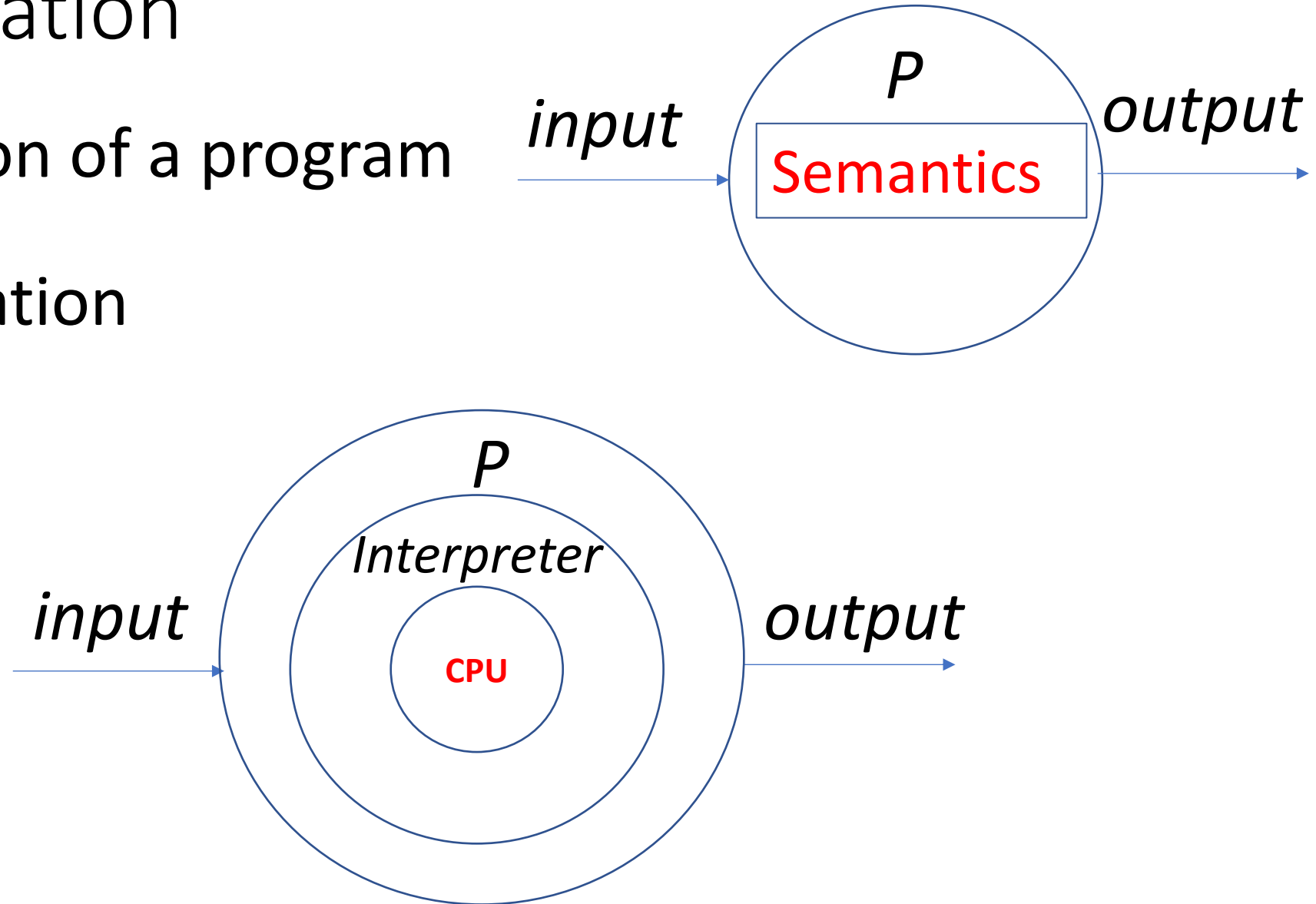


e.g., P : find input²

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) → interpretation