1 - introduction

programming linguistics: the study of programming languages

programming languages		natural languages
syntax	→	form
semantics	→	meaning
• can be analyzed, designed, and implemented on computers		• can only be analyzed

programming languages' fundemental reqirements

universal = every program must have a solution that can be programmed in the language (church- turing hypothesis) like loops, recursion.

natural = for solving problems within its intended application area (like numerics, strings, files)

implementable = on a computer. (mathematical notation and natural languages are not implementable, so they cannot be classified as programming languages)

efficient = works with acceptable CPU and memory

concepts of programming languages → underlying design

- data(values) and types
- variables and storage
- bindings and scope
- precedural abstraction
- data abstraction
- generic abstraction
- type systems
- control flow
- concurrency

paradigms of programming languages → different selection of key concepts for different styles

* imperative = use of variables, commands, and procedures Fortran, C, C++, Java

object oriented = use of objects, classes, and inheritance C++, Java

concurrent = use of concurrent processes, and various control abstractions Ada

* functional = use of functions ML and Haskell

* logic = use of relations Prolog (was the ancestall, but still the most popular)

scripting = by the presence of very high-level features Python

* mainly there are three types. imperative, functional, logic. object-oriented langu are all imperative. (objects are just like big variables)

programming language → $\boxed{\text{syntax}}$ + $\boxed{\text{semantics}}$ + $\boxed{\text{pragmatics}}$

form: how expressions, commands, declarations must be arranged

how
written by the programmer
read by other programmers
parsed by the computer

meaning: how it works

how
composed by the programmer
understood by other programmers
interpreted by the computer

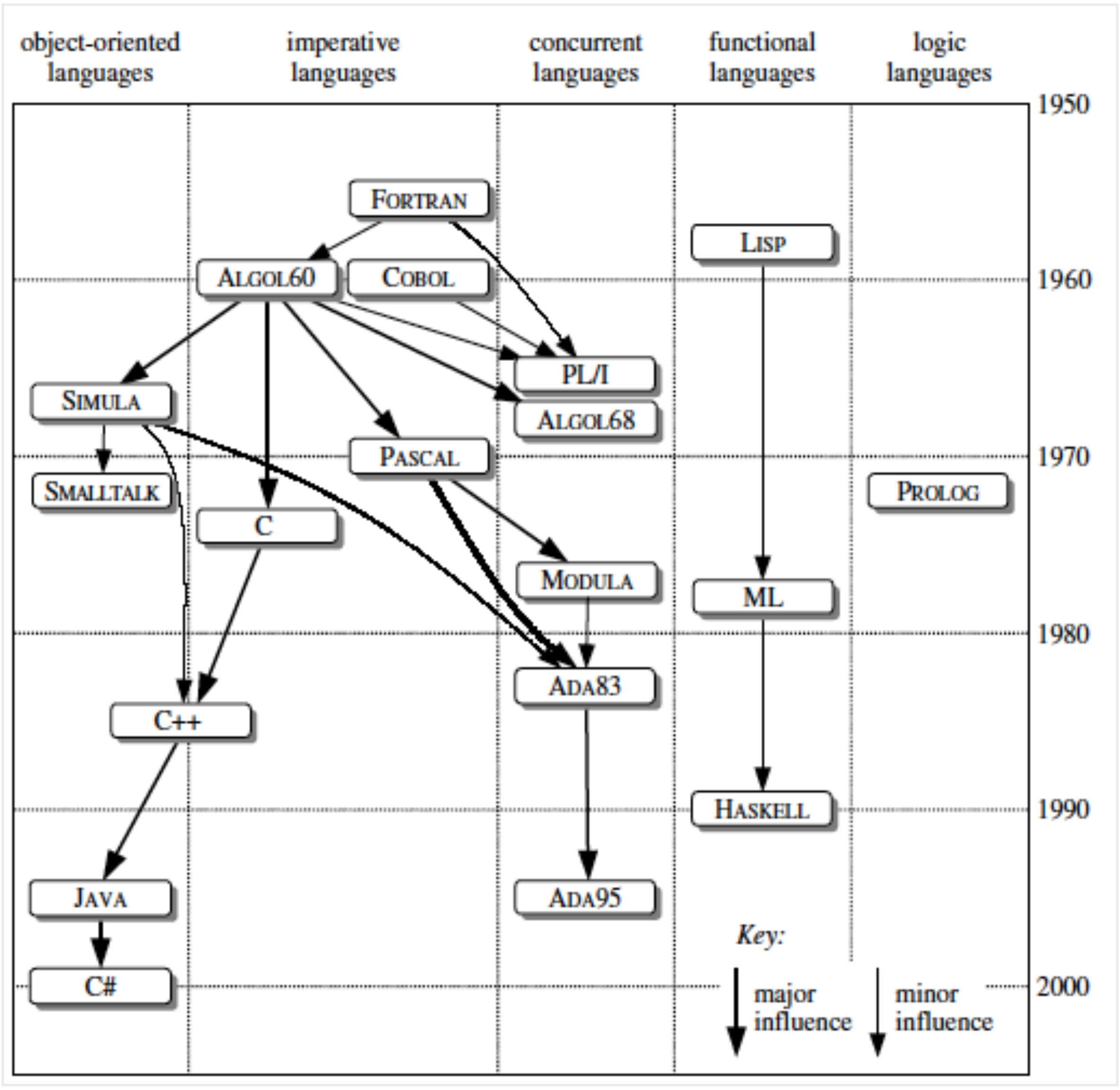
in which way the language is intended to be used in practice

how
programmers are expected to design and implement programs in practice

high level languages:

- independent of the machines on which programs are executed
- implemented by compiling programs into machine language
  by interpreting them directly
  by some combination of compilation and interpretation
- FORTRAN was the earliest major high-level language

language processors= any system for processing (executing, or preparing for execution) programs. include compilers
interpreters
auxiliary tools (like source-code editors and debuggers) like eclipse, visual studio



C → originally designed to be the system prog. lang. of the Unix operating system
↳ is suitable for writing both high/low level, but its low-level features are easily misused, causing code to be unportable and unmaintainable.

C++ → designed by adding object oriented concepts to C
↳ its design is clumsy, because it has all C's shortcomings and some more of its o

JAVA → designed by simplifying C++, by removing all its shortcomings.

C# → similar to java. more efficient implementation for ordinary application programming