# SOM2IF15 – Compilation

## Introduction

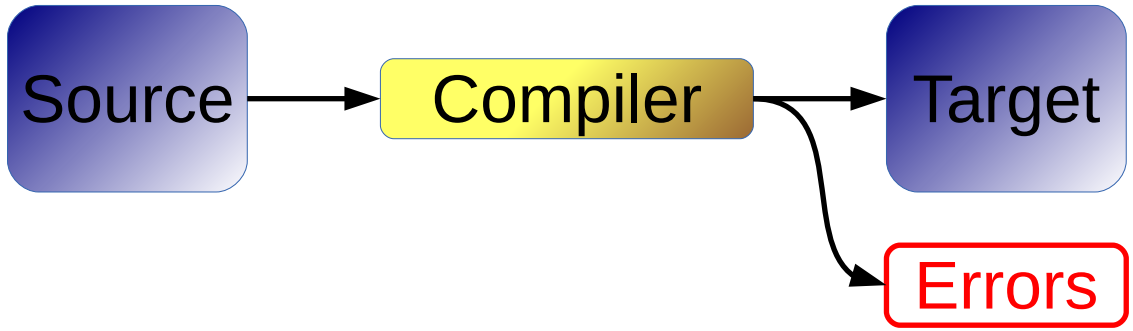
Frédéric Loulergue


2022

# What is a Compiler?

## A compiler is

a program that transforms a text written in a source language into a text written in a target language.




## Examples

- `javac`, `gcc`, `ocamlc`, `ocamlopt`, ...
- `latex`, `dot`
- Code generation from UML models
- Program extraction from formal proofs (Coq, Isabelle)

## SOM2IF15

We are interested in **programming** languages

# Let's compile a few programs

## Demo

- **gcc**
- **javac**

## Compilers: a vast area

- numerous source languages
- numerous target languages
- automata theory
- formal semantics
- algorithms (data structures, graphs, ...)
- hardware architecture
- software architecture
- ...

## Compilers: a vast area

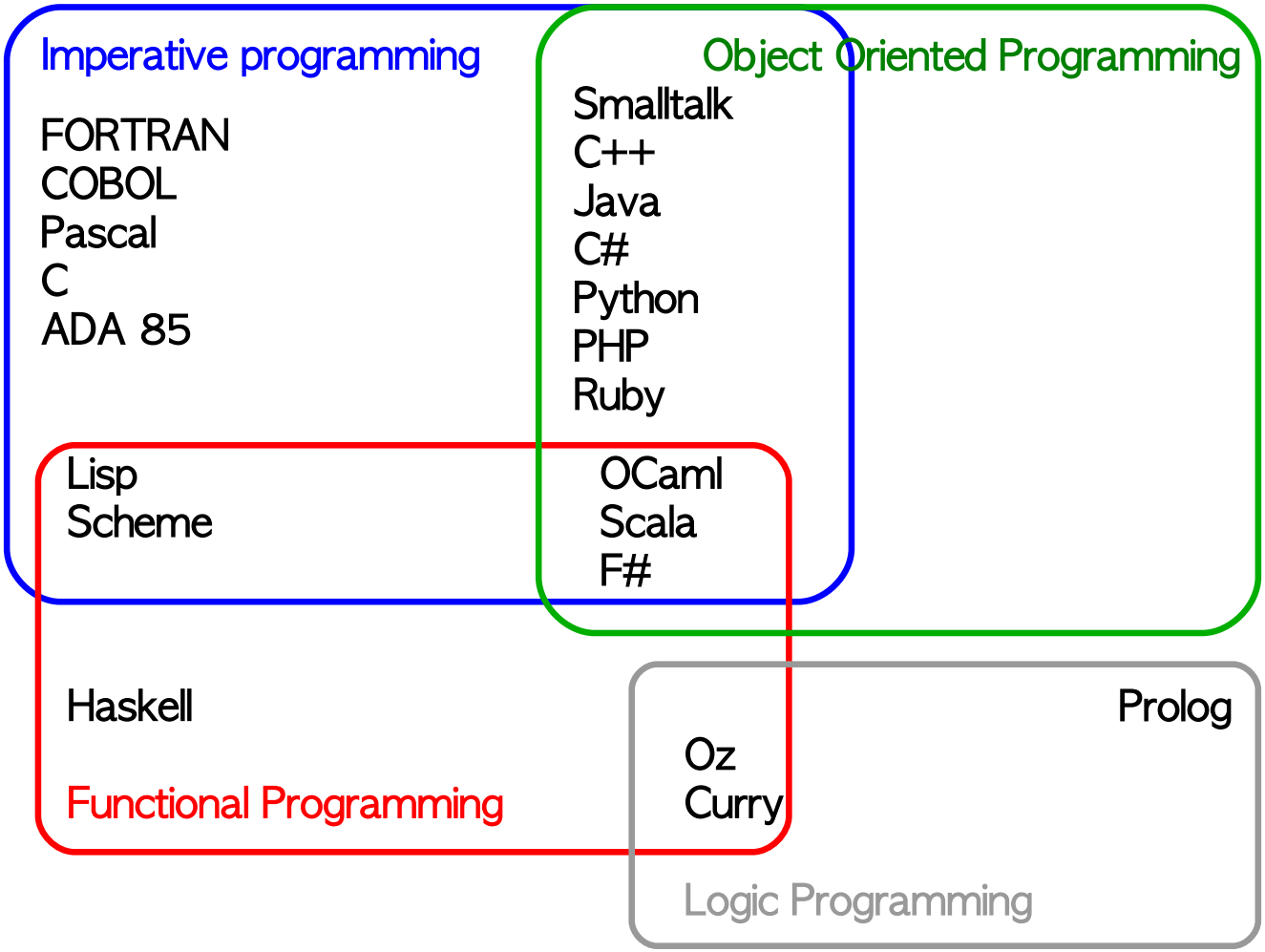
- numerous source languages
- numerous target languages
- automata theory
- formal semantics
- algorithms (data structures, graphs, ...)
- hardware architecture
- software architecture
- ...

## This Class

- Programming a compiler from A to Z
- Questions:
  - Source language?
  - Target language?

# Outline

# Programming Languages and Paradigms

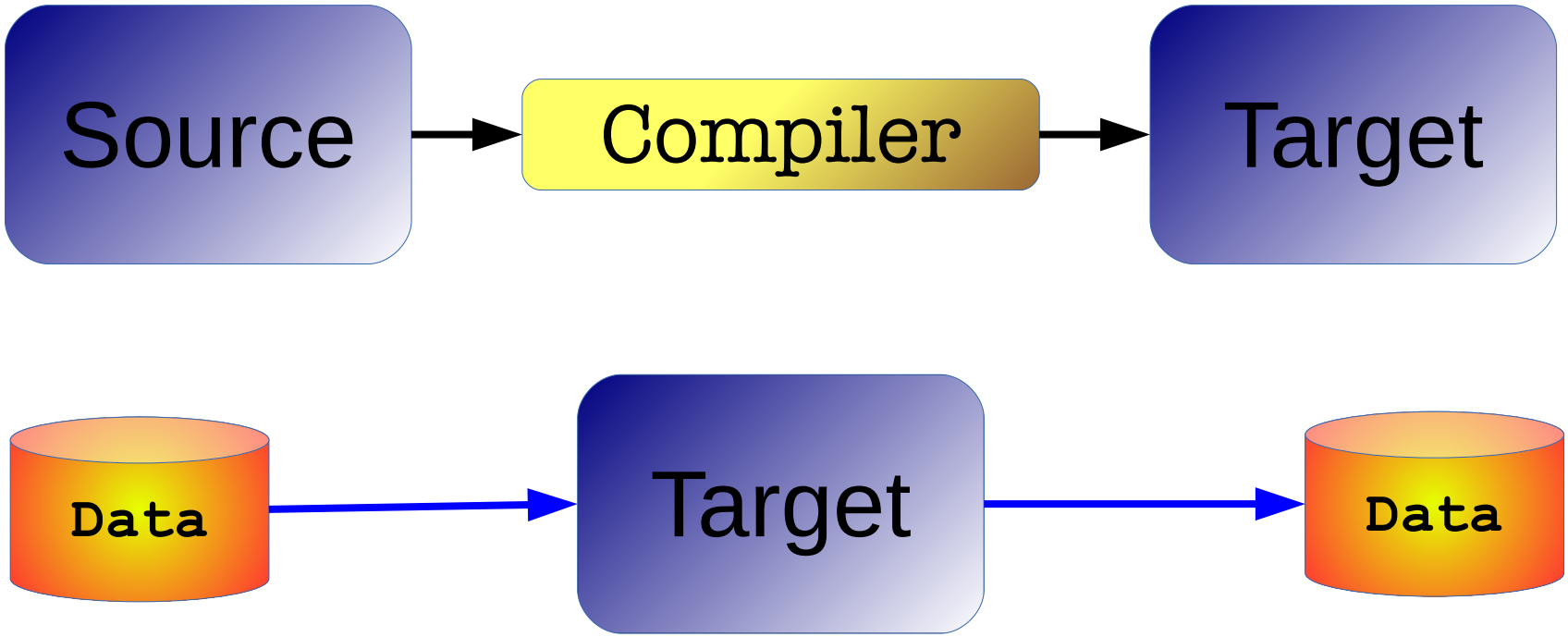



## LUO: An Imperative Programming Language

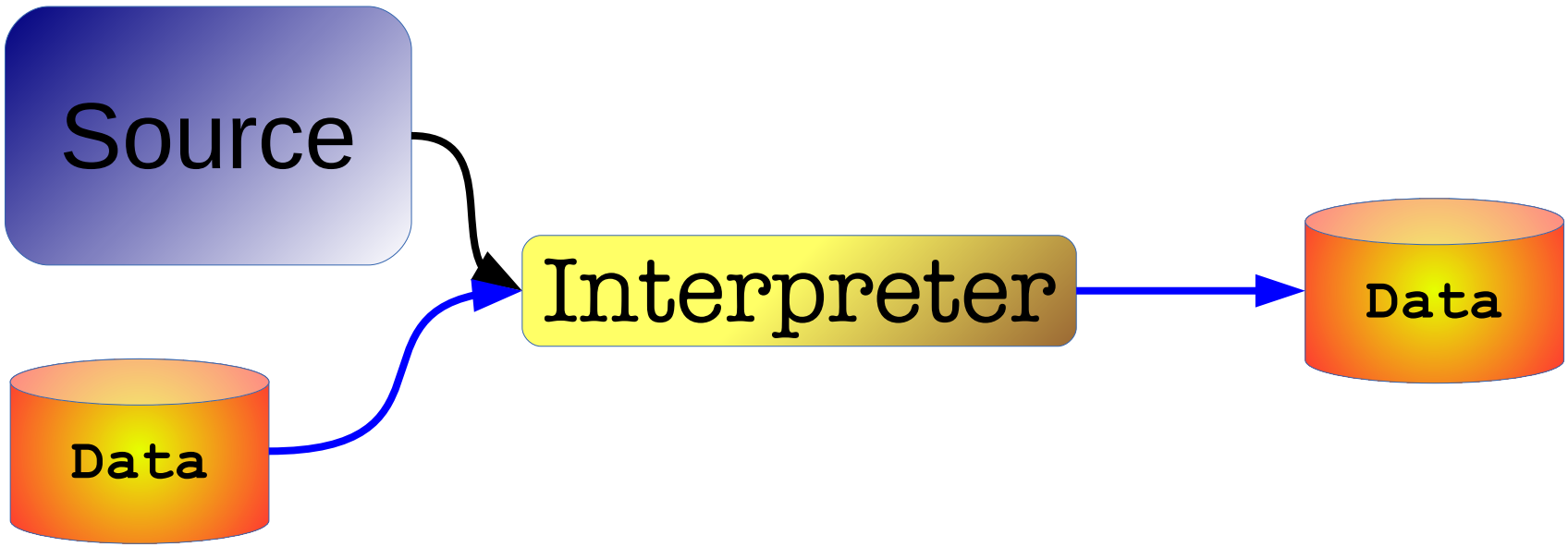
- Features to be defined together

# Outline

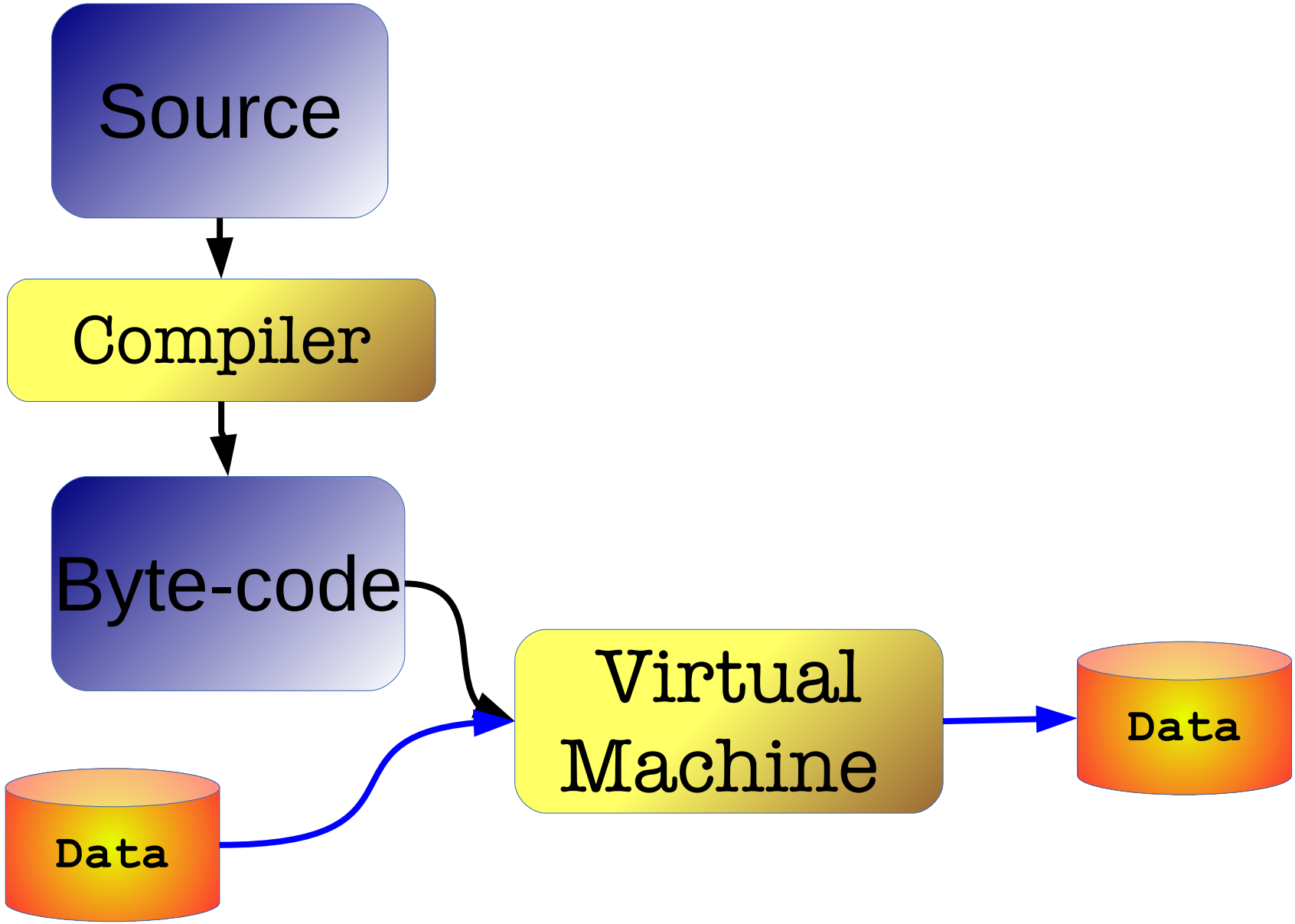
# Compilation and Execution



# Interpretation



# Byte-code Compilation and Virtual Machine



# Outline

# Last Phases of Compilers

## Object Files

- compiler produces assembly code
- transformed into machine code object by the assembler

Several object files are put together by the **linker** that establishes links with the system libraries to produce an executable file

## *ABI - Application Binary Interface*

For a given *operating system* and a given *architecture* the ABI specifications include:

- data placement
- call conventions
- object file formats
- functions of the system libraries

# Machine Code

## CISC Processors – *Complex Instruction Set Computer*

- many complex instructions (variable sizes)
- few registers
- direct programming of the machine

## RISC Processors – *Reduced Instruction Set Computer*

- small and regular instruction set
- many registers
- simplified and unified decoding
- not meant for direct programming

# Byte-Code Compilation

## Examples of Byte-codes and Virtual Machines

- P-code
- Byte-code-octet OCaml
- Byte-octet Java and the *Java Virtual Machine*
- Byte-octet Dalvik and the DVM (Android)
- *Common Language Runtine*

Available specification, assemblers

# Native-code/Byte-code Compilation, Interpretation?

## Comparison

Interpreter:

- ✓ easy to implement, to test, better interaction
- ✗ memory usage, execution speed

Compiler:

- ✓ memory usage, execution speed, error detection, optimization
- ✗ more complex, less flexible
- native-code: complex but efficient, less portable
- byte-code: less efficient, more portable, virtual machine to design and implement

# This Class

- Compiler to (almost) native code
- Interpreter
- RISC processor: MIPS
- The SPIM simulator:
  - Direct execution of assembly code
  - Without the need of a operating system
  - Very small system library for basic input/output