# Compiler Construction: Practical Introduction

# Projects

**Eugene Zouev**

Fall Semester 2025

Innopolis University

# Project List 1

- **Four "toy" languages**

**D** Dynamic *(like JavaScript)*
**O** Object-oriented *(like Oberon-2)*
**F** Functional *(~CoreLisp; like Lisp & Scheme)*
**I** Imperative *(like Pascal/Oberon)*

- **Implementation details**

- Compiler or interpreter
- Implementation languages: C/C++, C#, Java, Rust(?)
- Tools: yacc/bison, ~~ANTLR~~, hand-written
- Target codes & platforms:
  - MSIL
  - Java bytecode
  - LLVM bitcode
  - WASM
  - Own VM ☺

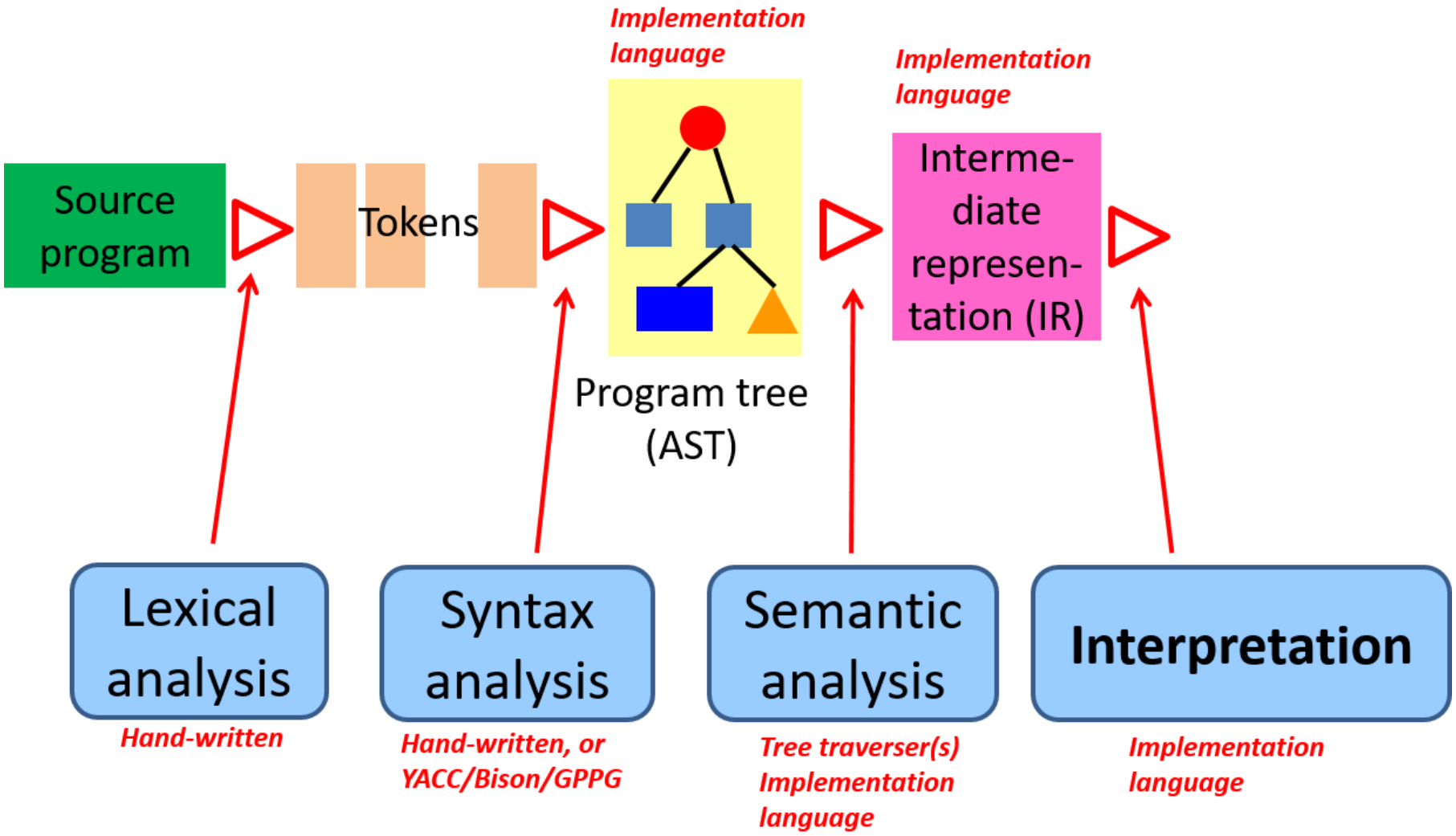
# Project D: Dynamic

**Note: lambdas and captures should be supported!!**

**Short language characteristics**

- Object types are not specified and can change while program execution
- The language is **interpreted**
- Major notion: variable & literal (constant)
- Program structure: a sequence of declarations and statements
- Builtin types: built-in: integer, real, boolean, string
  User-defined types: array, tuple, function
- Implicit type conversions
- Statements: assignment, if/while, return, input/output

# Project D: Dynamic

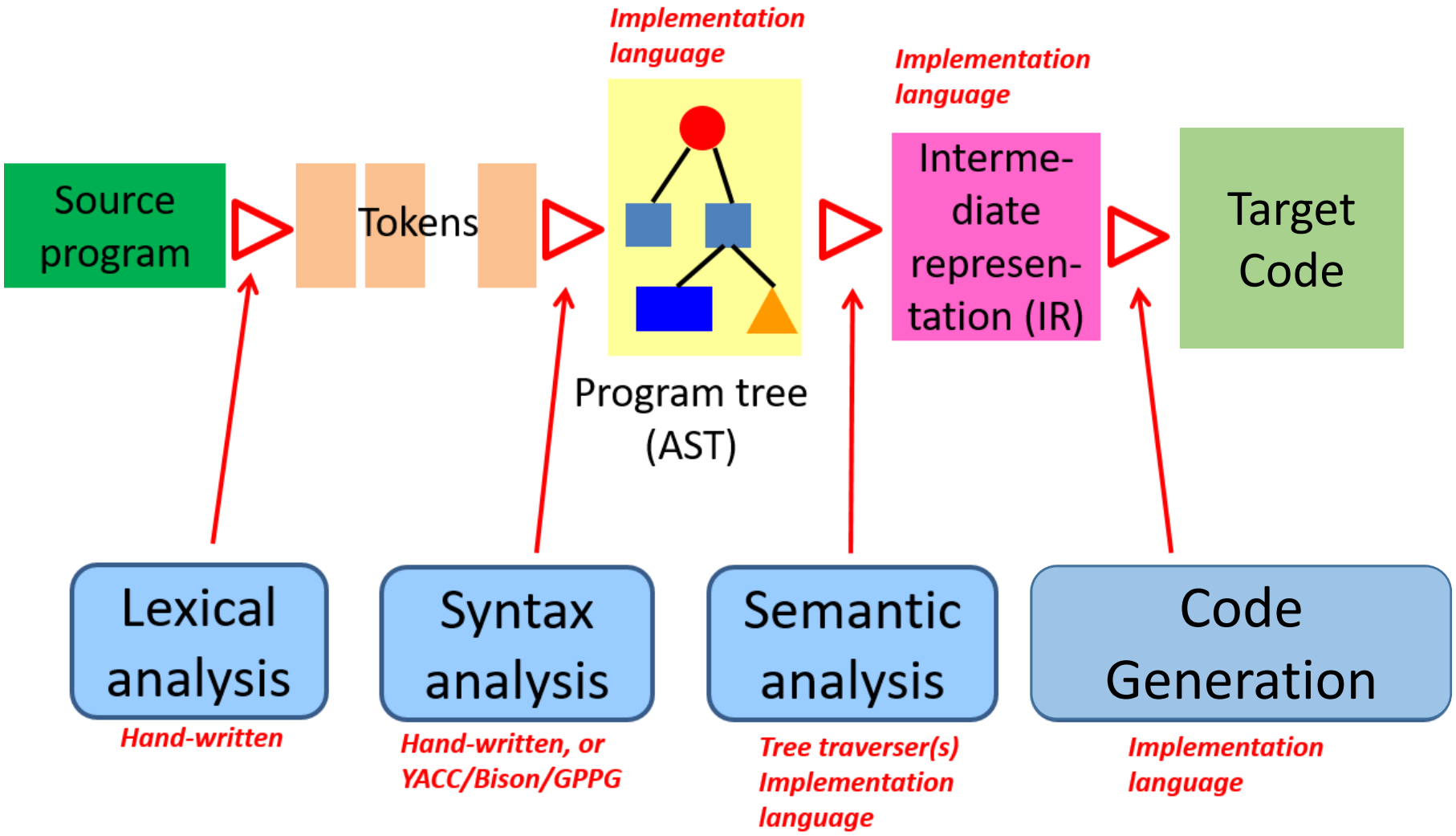


# Project O: Object-Oriented

**Short language characteristics**

- The language is statically typed. Object types are fixed by object declarations and cannot change while program execution.
- Object types are classes – either predefined of user-defined.
- Classes contain data declarations and method declarations.
- The single inheritance and virtual functions are supported.
- No expressions! – only function calls.
- The language is **compiled**. The target code is either an assembly language, of LLVM bitcode, or JVM bytecode, or .NET CIL.
- Program structure: a sequence of classes.
- Statements: a standard set (assignment, if/while, return, input/output)

# Project O: Object-Oriented



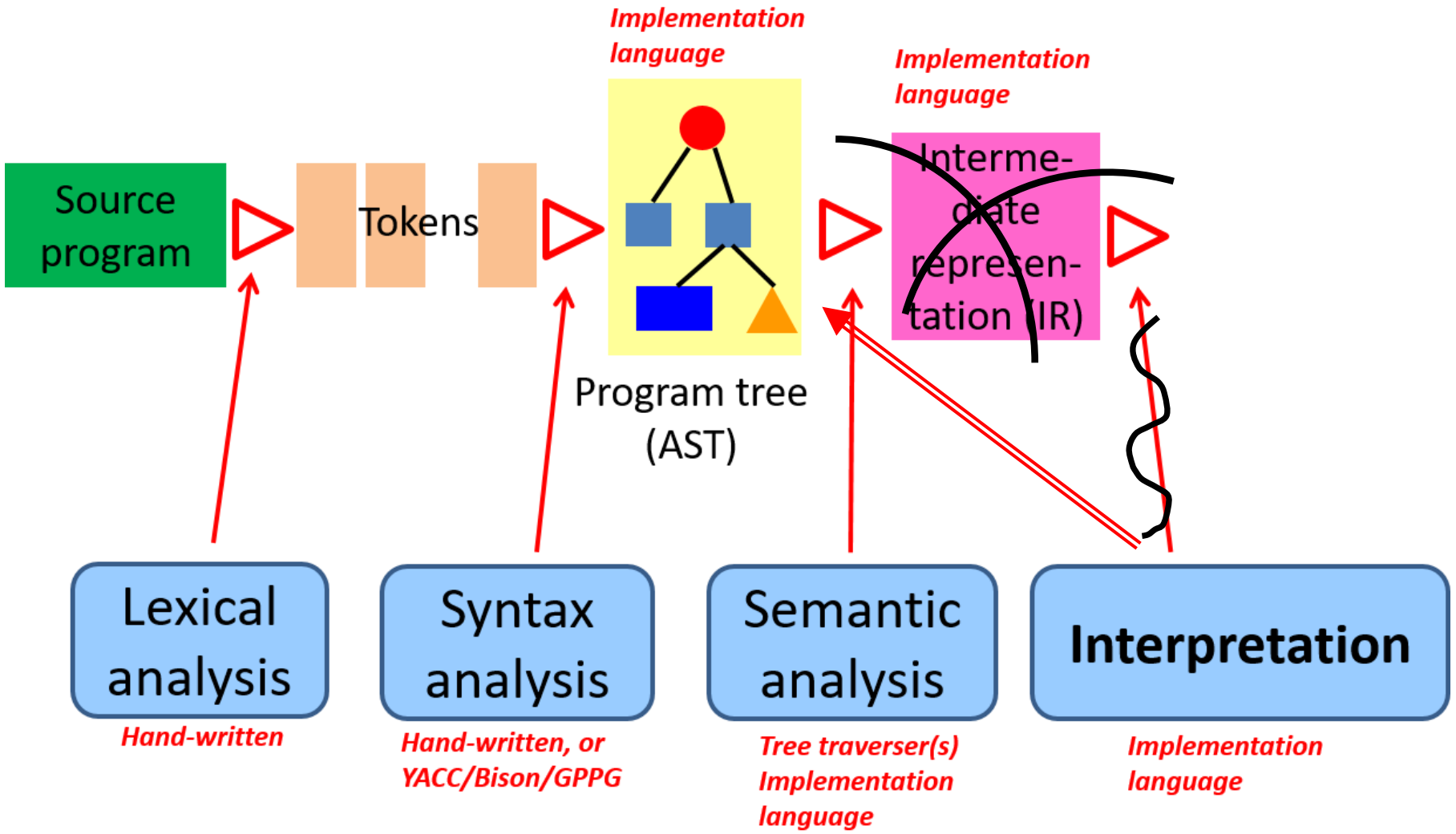
# Project F: Functional

**Note: lambdas and captures should be supported!!**

**Short language characteristics**

- There are three atomic predefined types: integers, reals & booleans.
- There are two basic structures: atoms and lists.
  Atom has the name and the value. The atom value can be either of a predefined type, or, or list. In some contexts atom represents itself.
  A list is an ordered sequence of atoms, literals and lists.
- Basic list access rules include taking the head of the list, the "tail" of the list, and constructing a list from its head and the "tail"..
- The language is dynamically typed and **interpreted**.
- Program structure: a sequence of lists and/or atoms
- Special forms (lists): functions, lambdas (unnamed functions), control structures.
- A minimal set of predefined functions: arithmetic, lists operations, etc.

# Project F: Functional

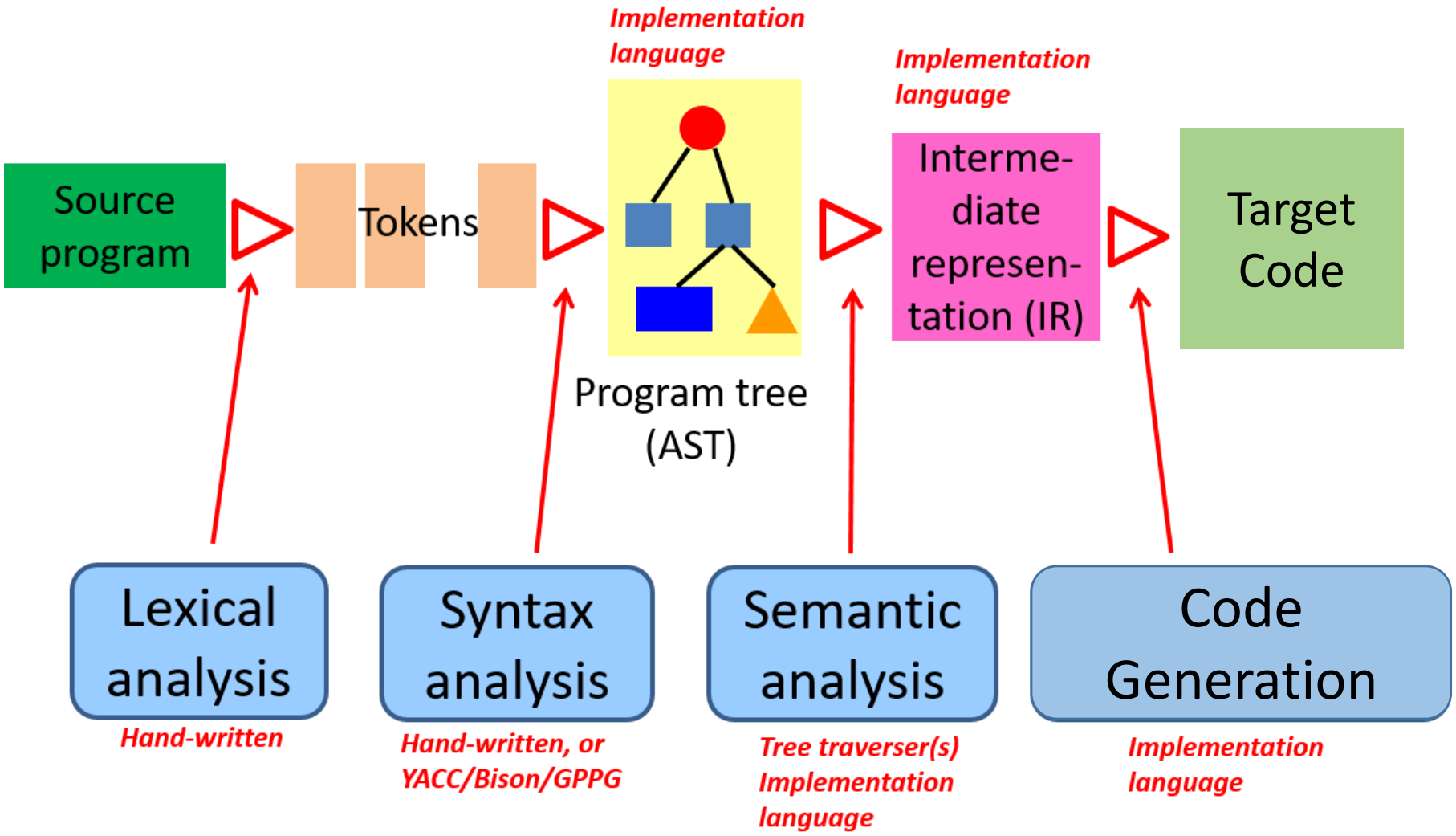


# Project I: Imperative

**Short language characteristics**

- The language is statically typed. Object types are fixed by object declarations and cannot change while program execution.
- There are three predefined data types: integer, real and boolean. There are two predefined data structures: structs and arrays.
- Full expression syntax with usual set of operators.
- The language is **compiled**. The target code is either an assembly language, of LLVM bitcode, or JVM bytecode, or .NET CIL.
- Program structure: a sequence of data and routine declarations.
- Statements: a standard set (assignment, if/while, return, input/output)

# Project I: Imperative



# What You Should Do at the Beginning

- Carefully read all language descriptions.
- Choose the project: until next Tuesday!
- Create a team: 2 persons, not more.
- Fill the Form with all details: until next Tuesday!
- Find inconsistences and missed parts in the description of the chosen project. Ask lecturer or TA.
- Your start: Write a small test suite (~15 test cases) for the language chosen.
- Notice the whole working plan (Lecture slide 14).