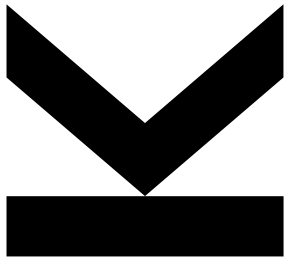
LLVM IR IN GRAALVM: MULTI-LEVEL, POLYGLOT DEBUGGING WITH SULONG



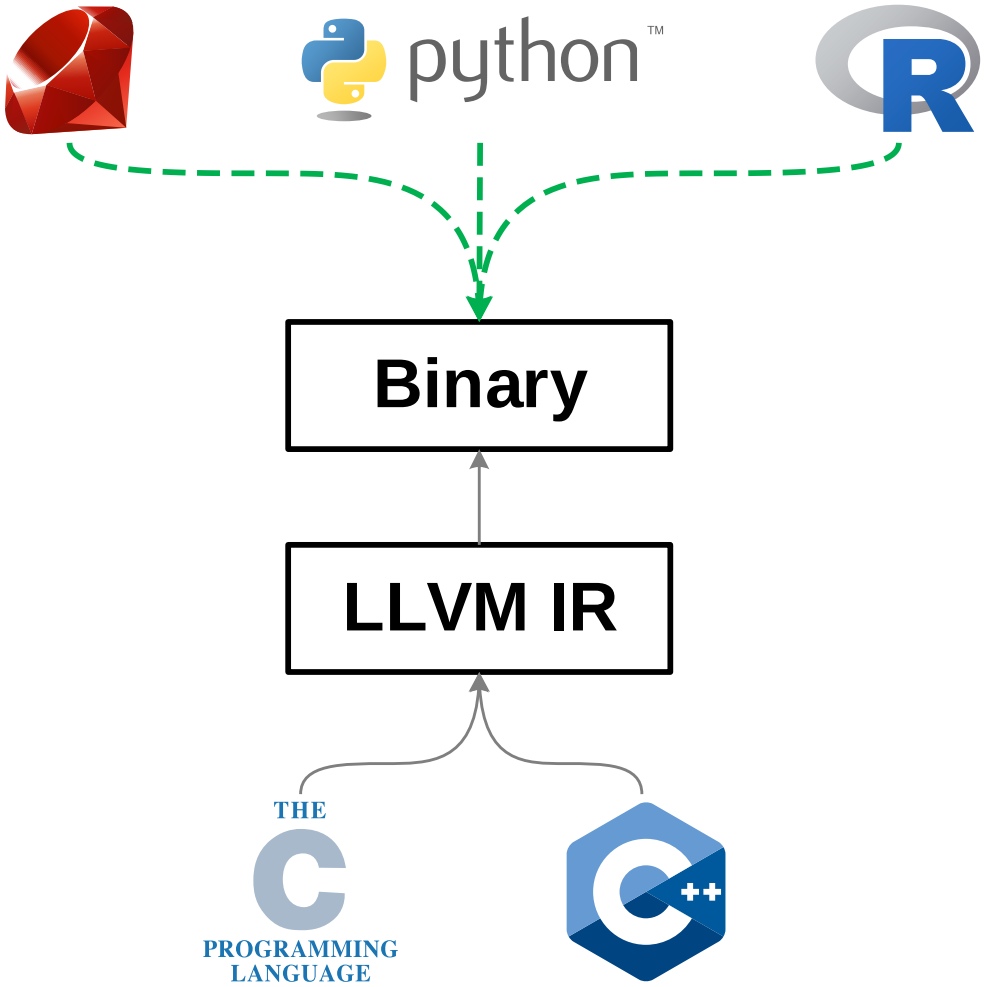
Jacob Kreindl

2019 European LLVM Developers' Meeting, April 8-9, 2019

JOHANNES KEPLER
UNIVERSITY LINZ

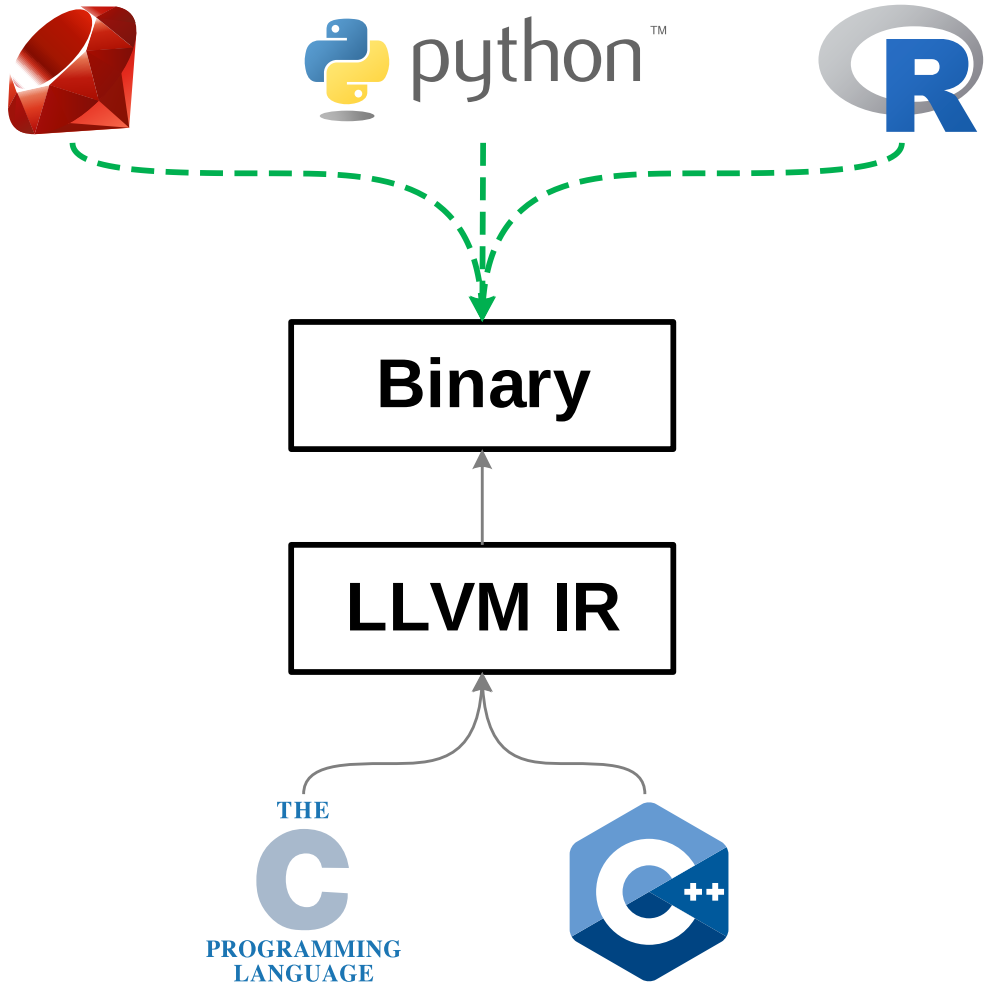
ORACLE
Labs

MOTIVATION



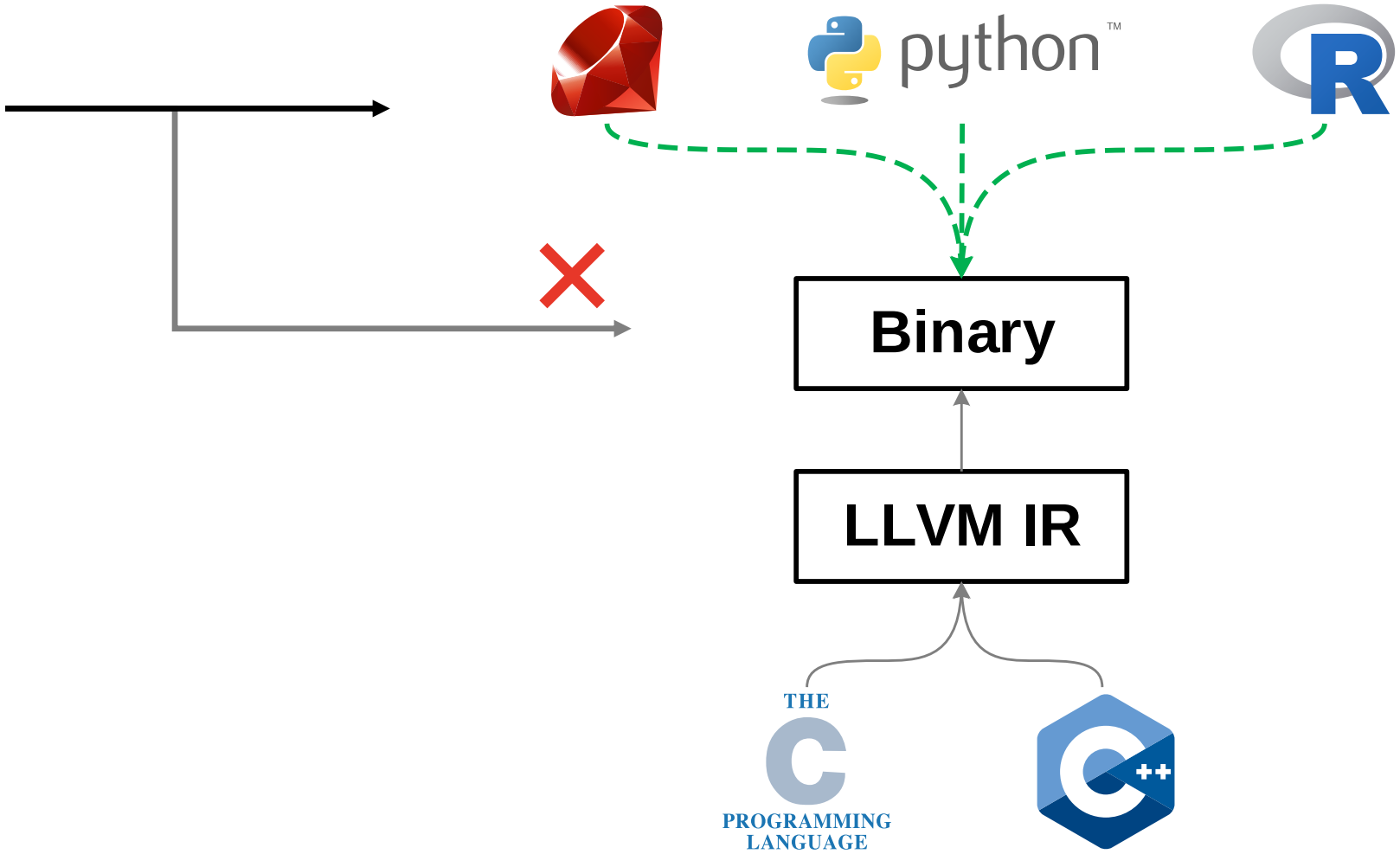
MOTIVATION

Debuggers for
Dynamic Languages



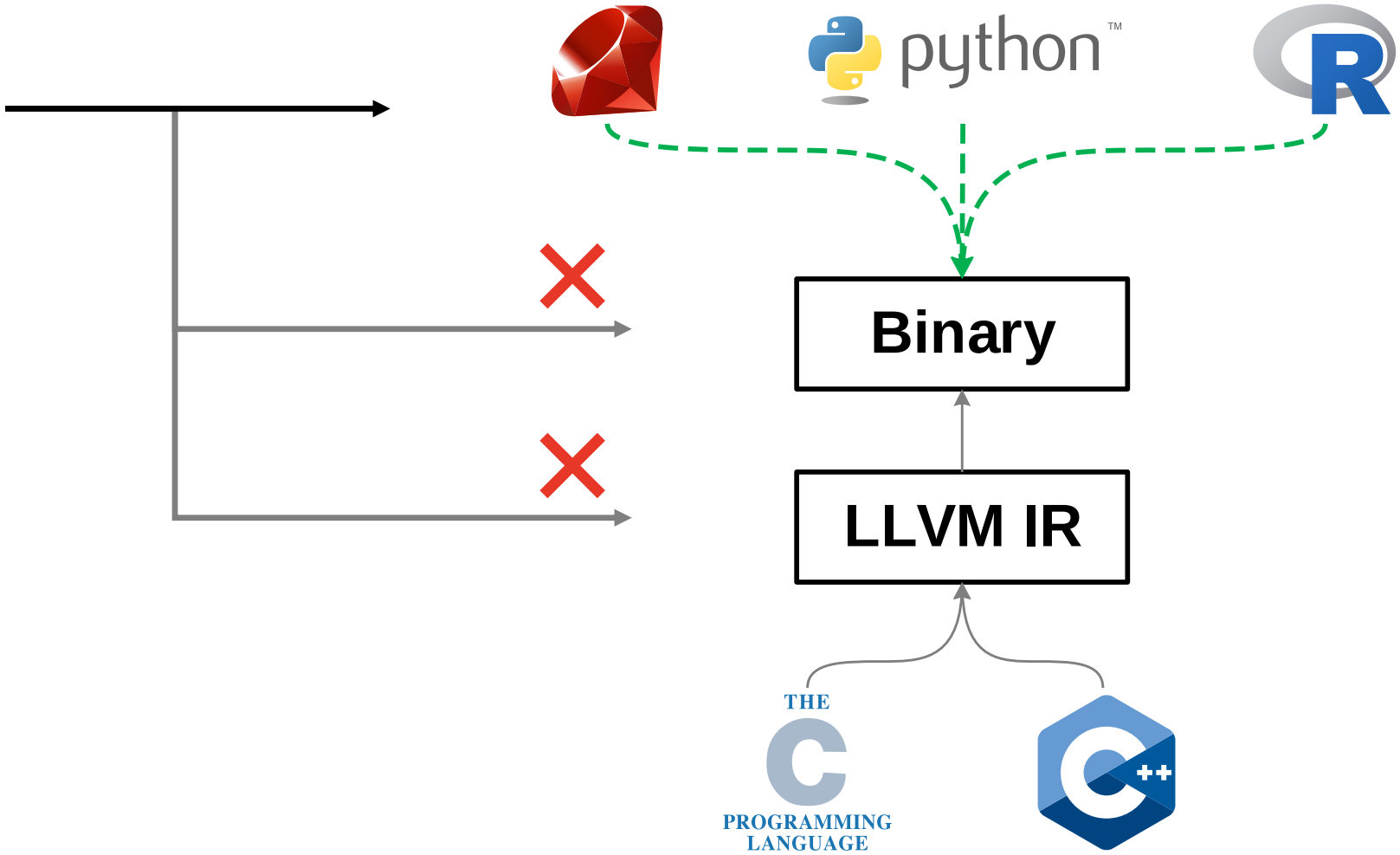
MOTIVATION

Debuggers for
Dynamic Languages



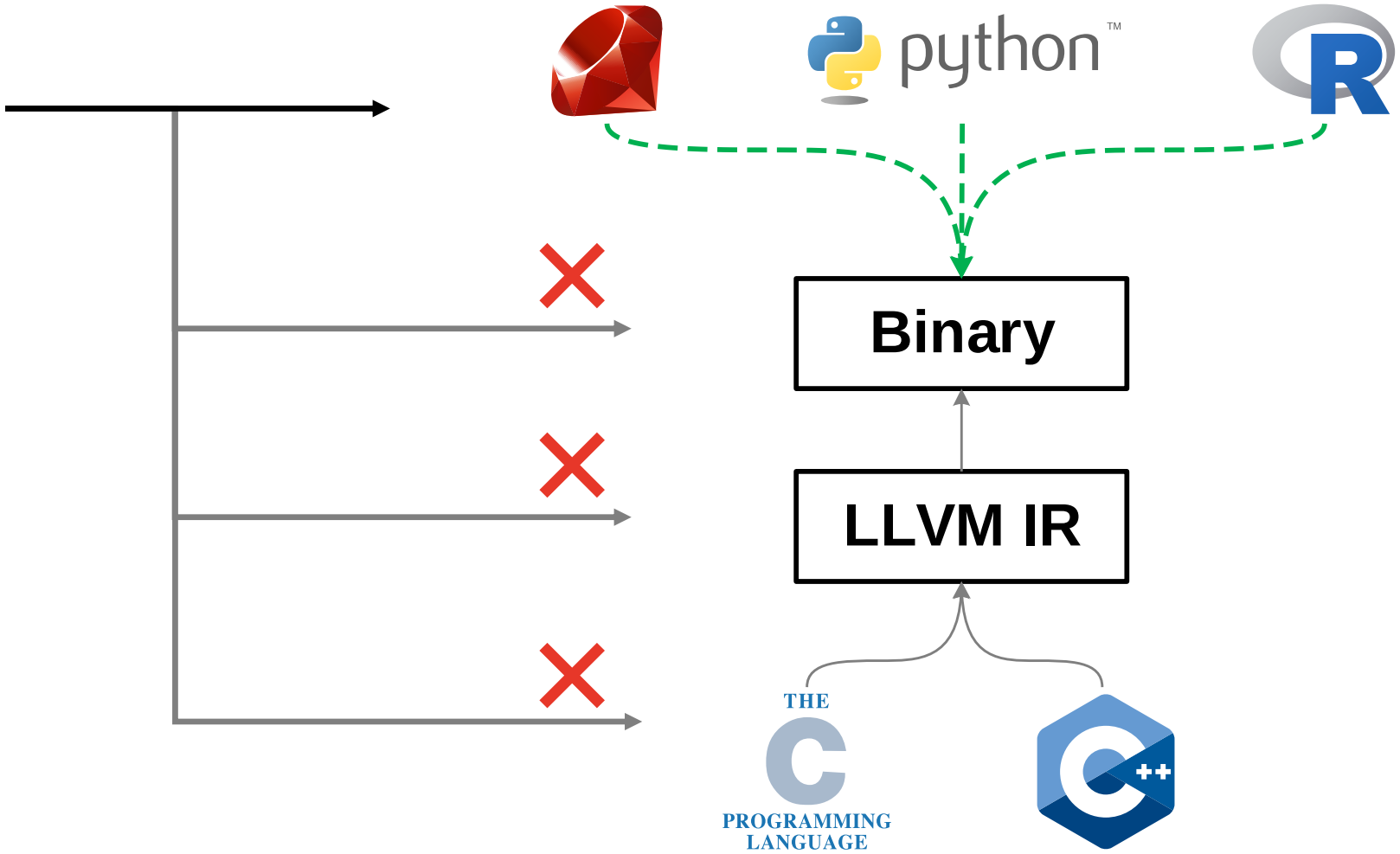
MOTIVATION

Debuggers for
Dynamic Languages



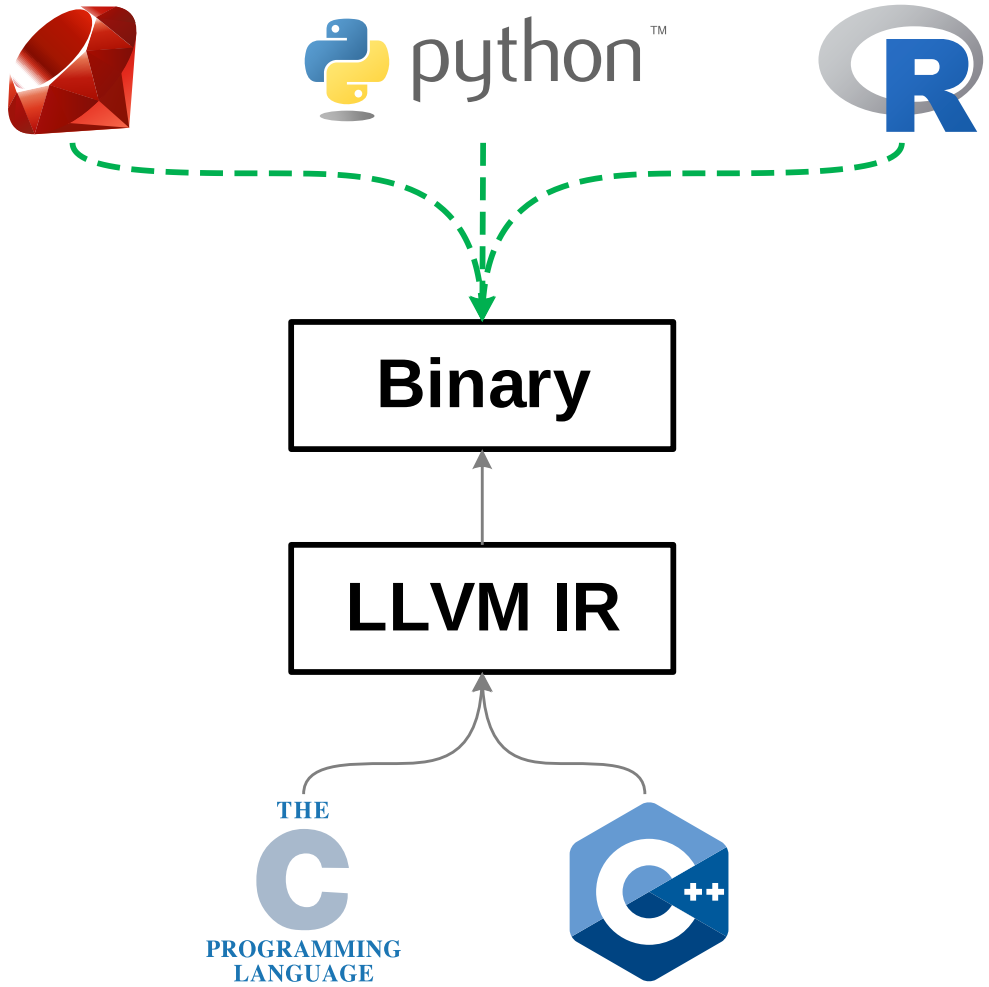
MOTIVATION

Debuggers for
Dynamic Languages



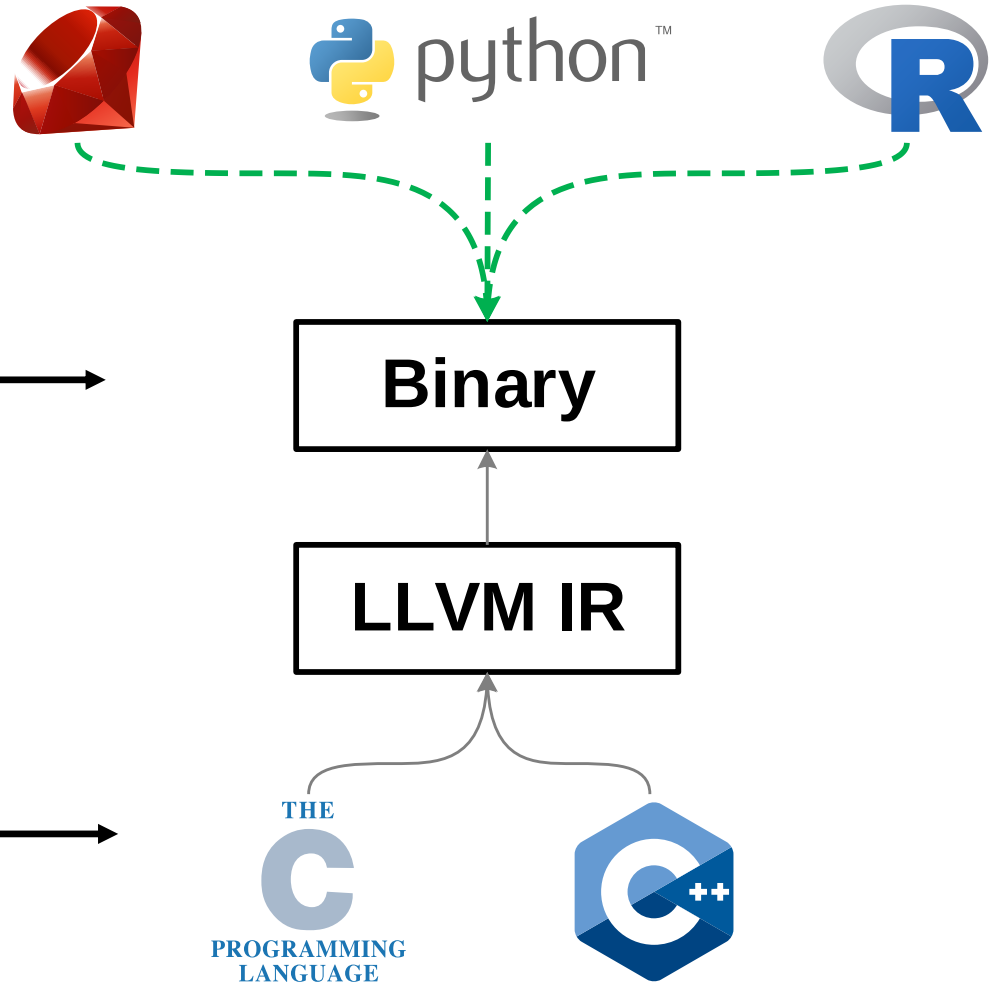
MOTIVATION

Debuggers for
Dynamic Languages



MOTIVATION

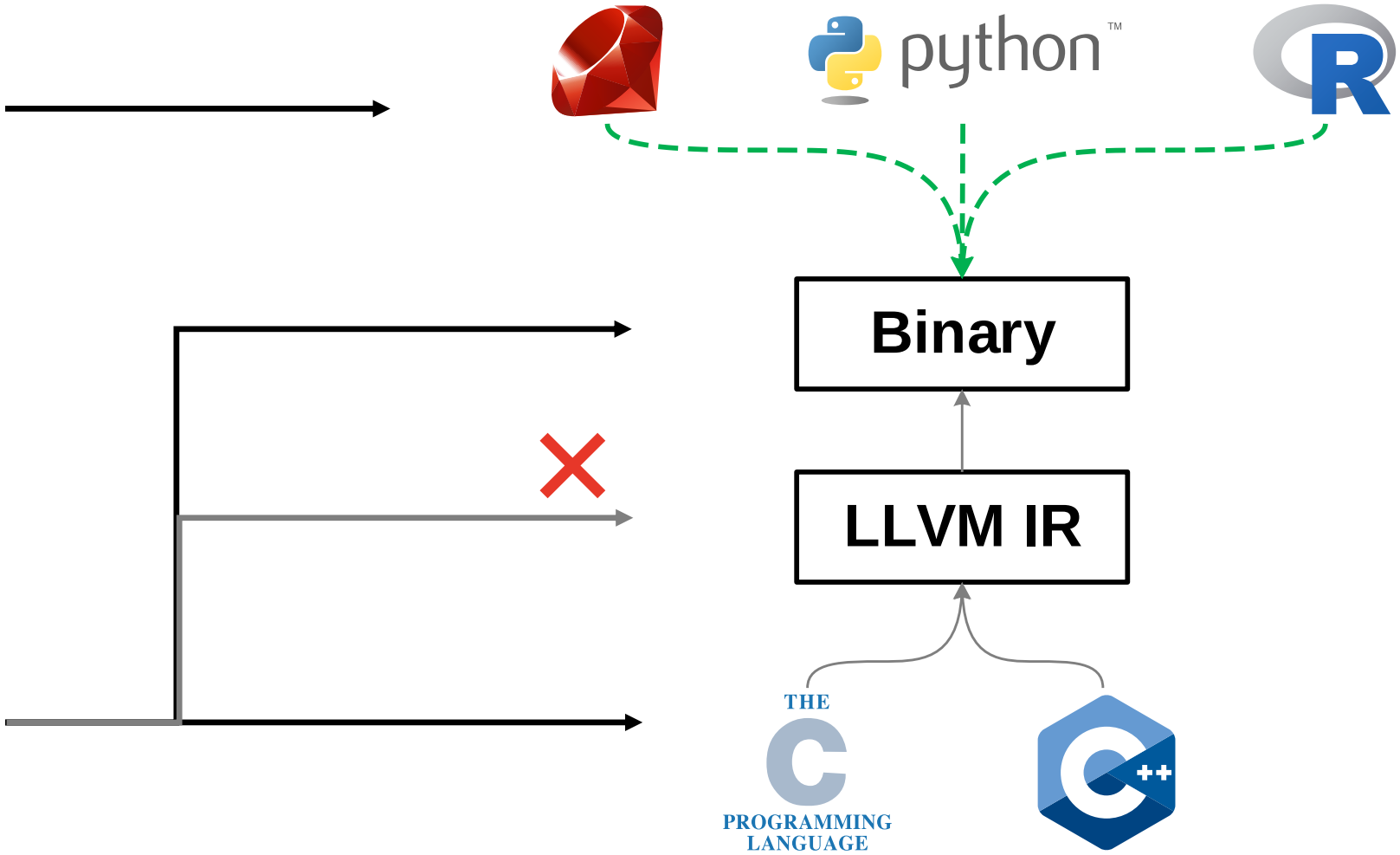
Debuggers for
Dynamic Languages



Debuggers for
Native Code

MOTIVATION

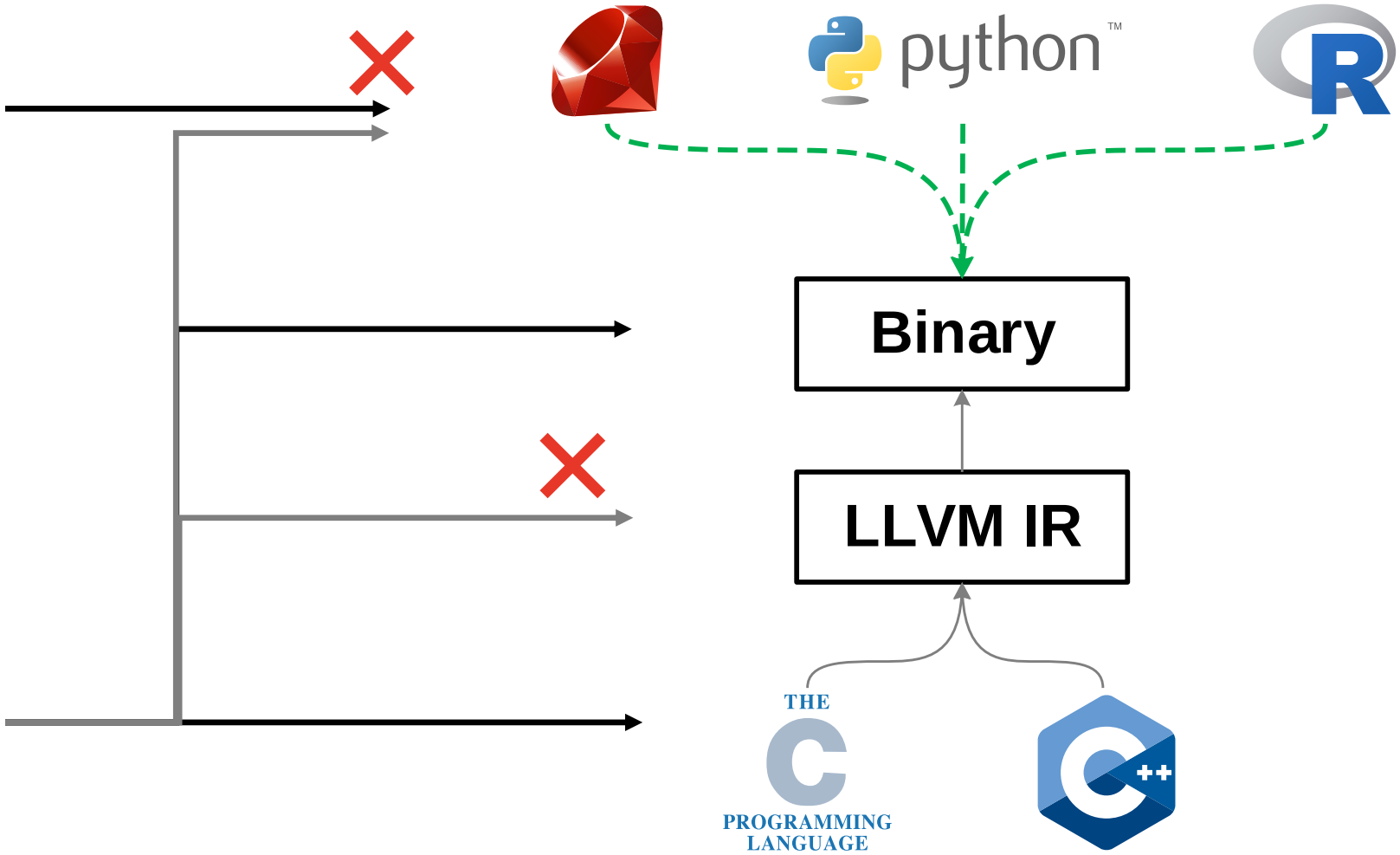
Debuggers for
Dynamic Languages



Debuggers for
Native Code

MOTIVATION

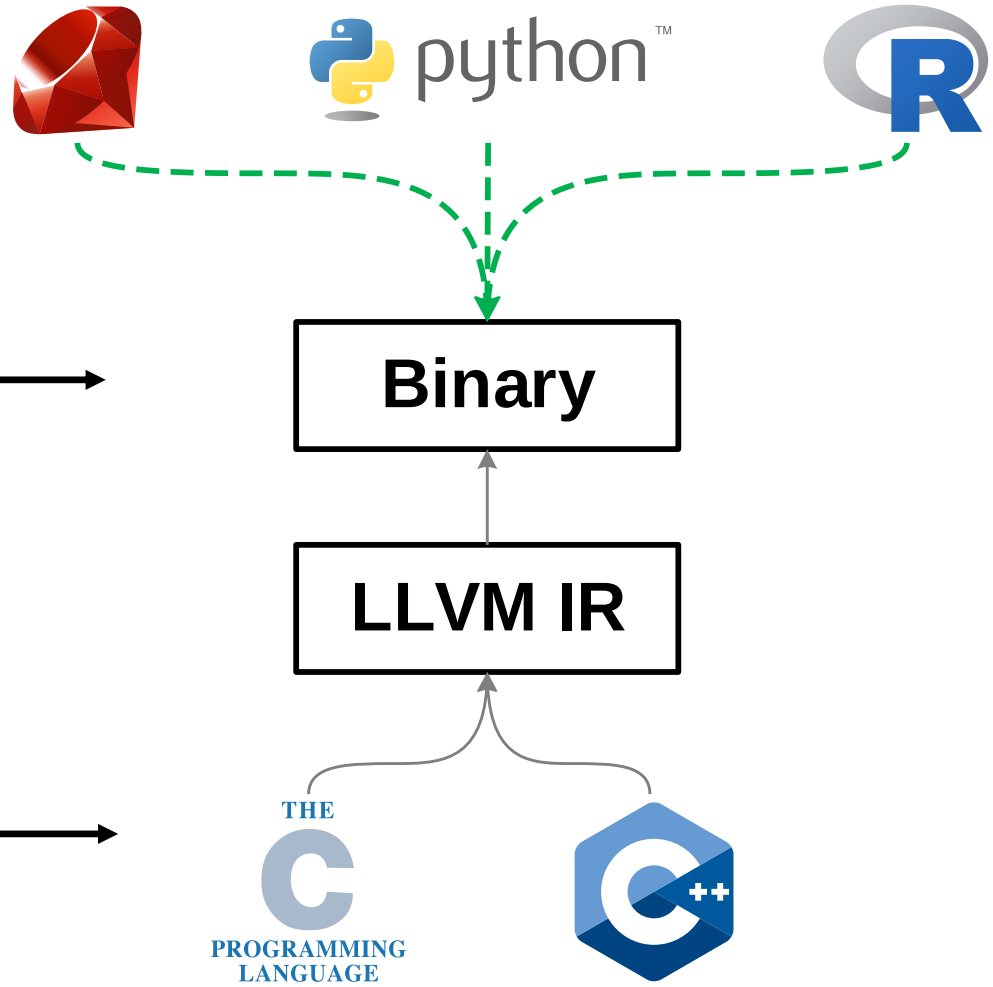
Debuggers for
Dynamic Languages



Debuggers for
Native Code

MOTIVATION

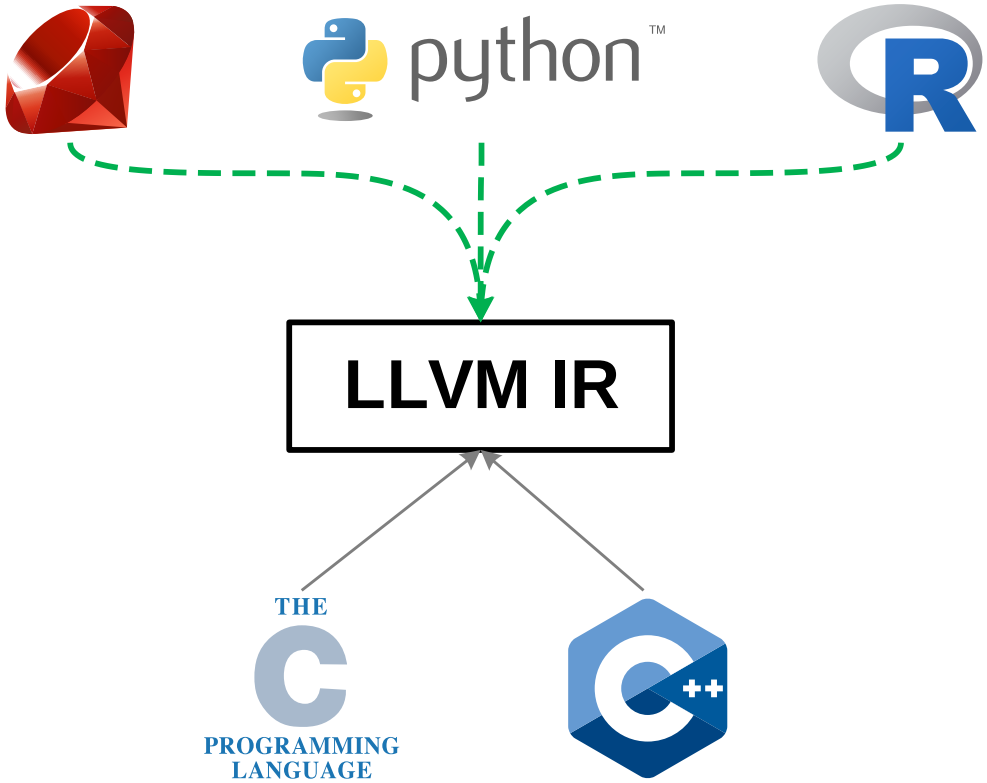
Debuggers for
Dynamic Languages



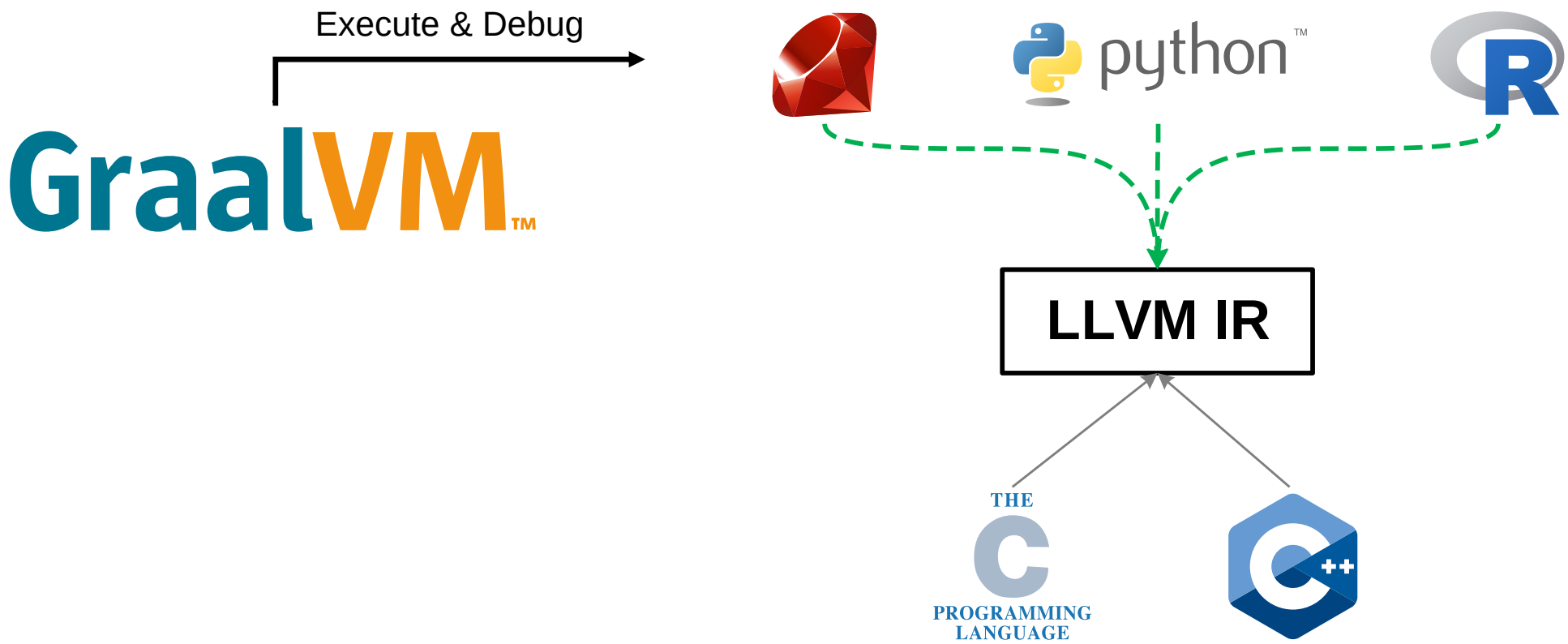
Debuggers for
Native Code

GRAALVM

GraalVM™

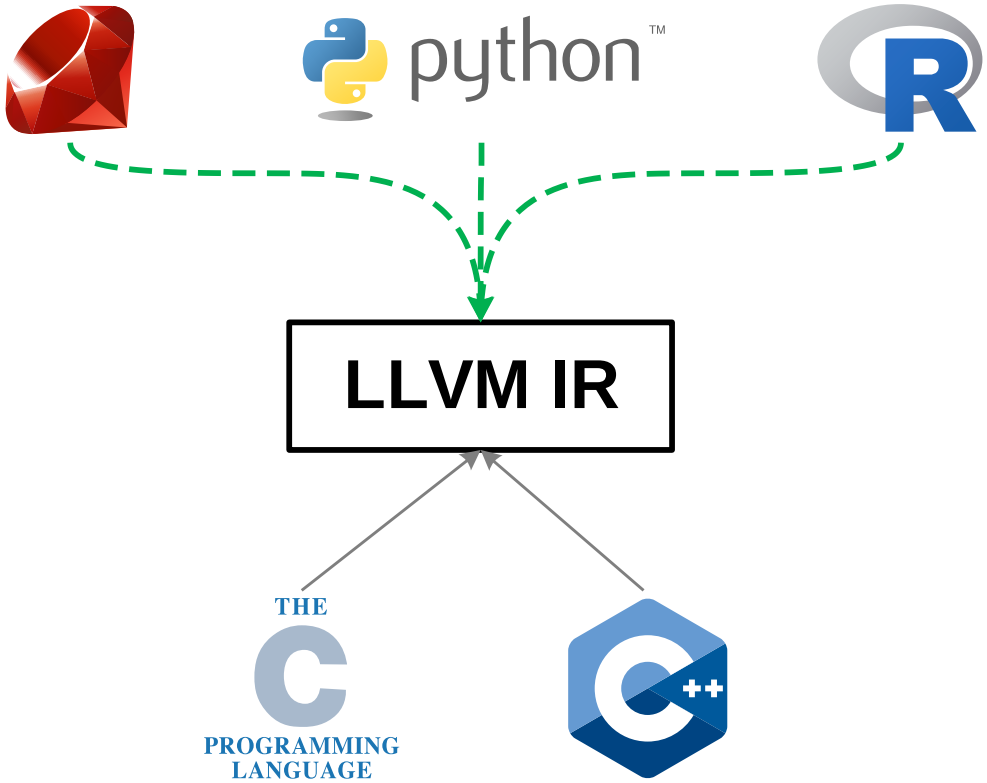


GRAALVM

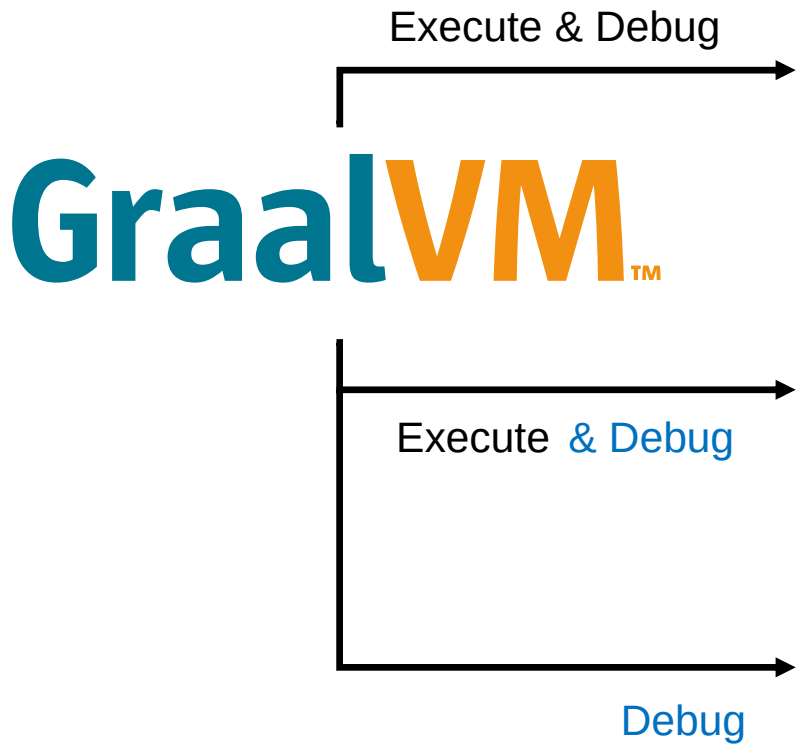

Execute & Debug
GraalVM
python
R
LLVM IR
THE
C
PROGRAMMING
LANGUAGE
C++

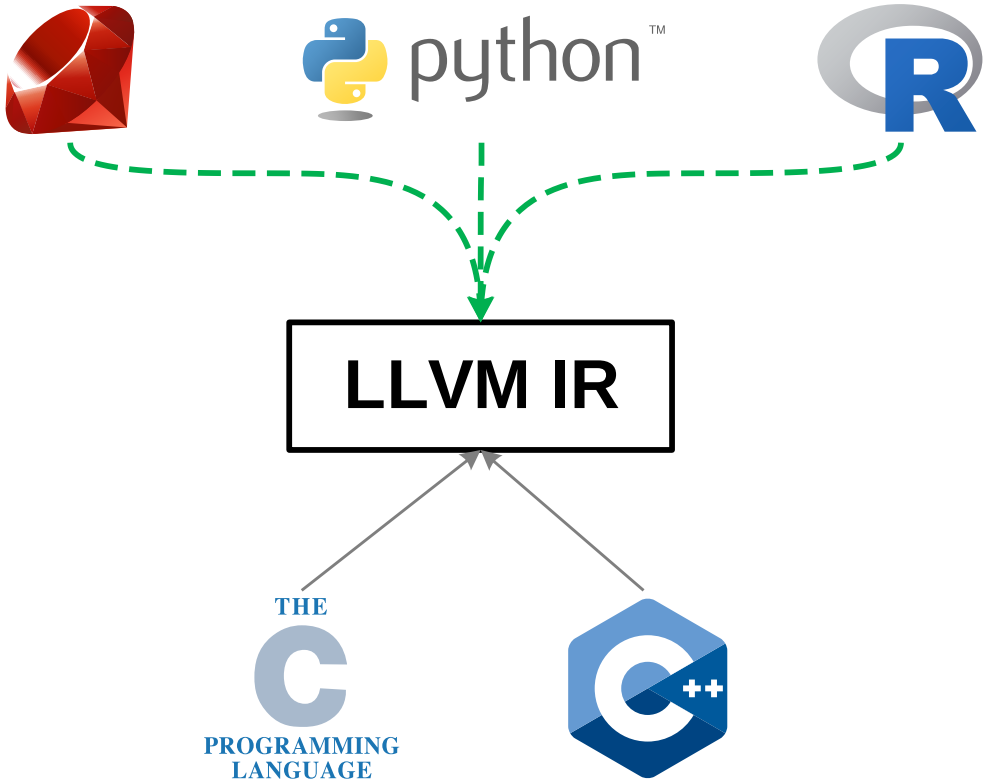
GRAALVM


Execute & Debug
GraalVM™
Execute


python™
R
LLVM IR
THE
C
PROGRAMMING
LANGUAGE
C++

GRAALVM


Execute & Debug
GraalVM™
Execute & Debug
Debug


python™
R
LLVM IR
THE
C
PROGRAMMING
LANGUAGE
C++

GRAALVM FOR DYNAMIC LANGUAGES

GraalVM™

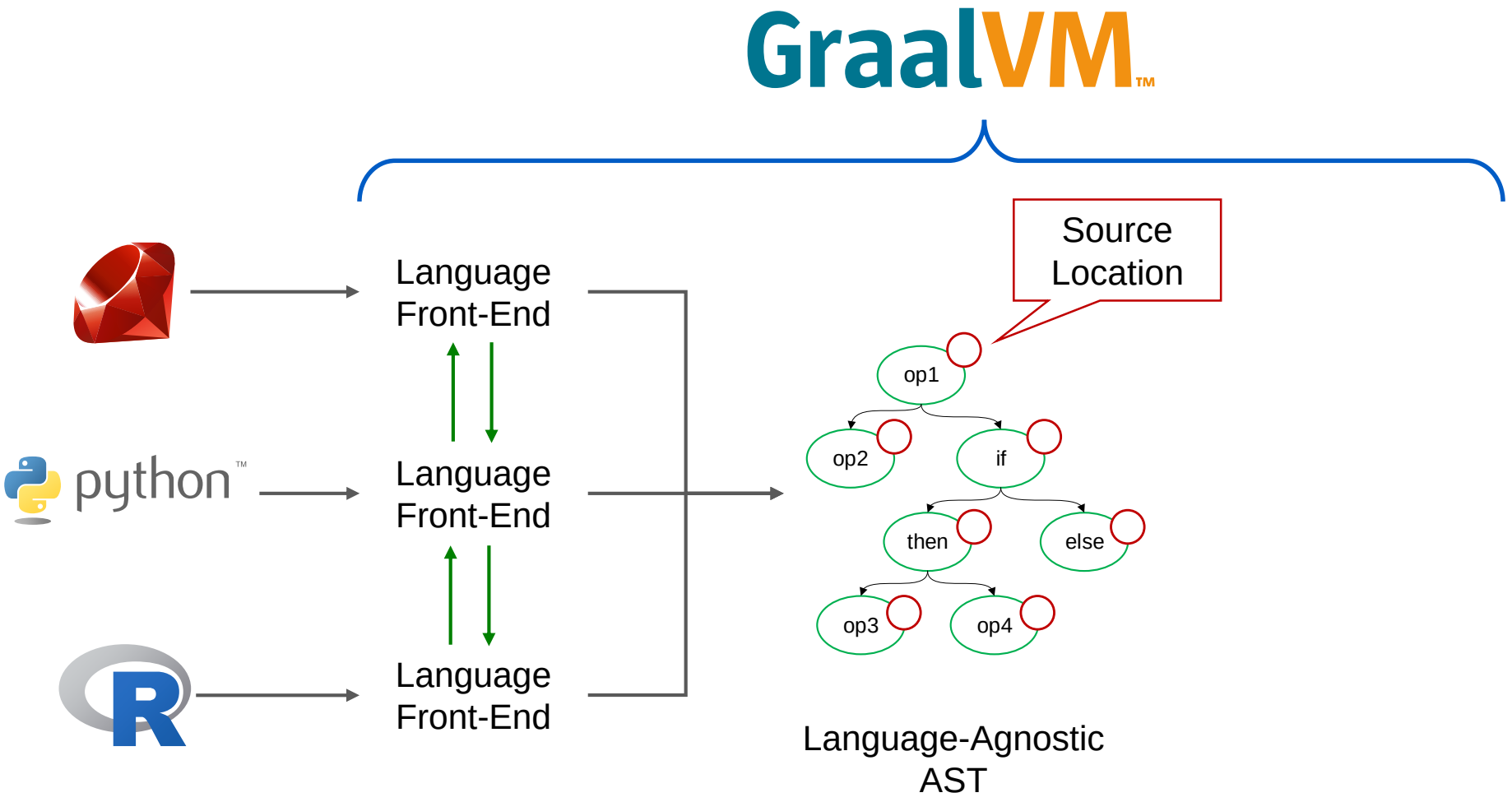
GRAALVM FOR DYNAMIC LANGUAGES

Language
Front-End

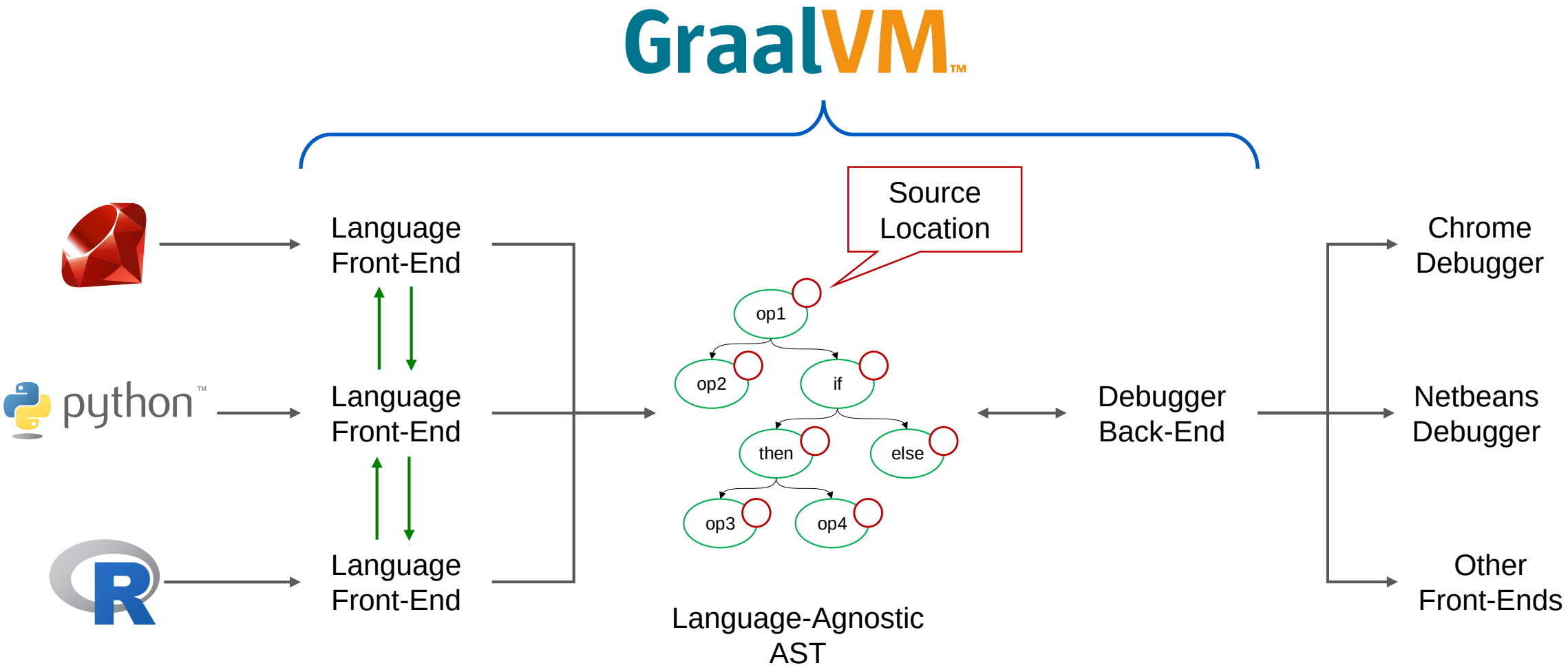
Language
Front-End

Language
Front-End

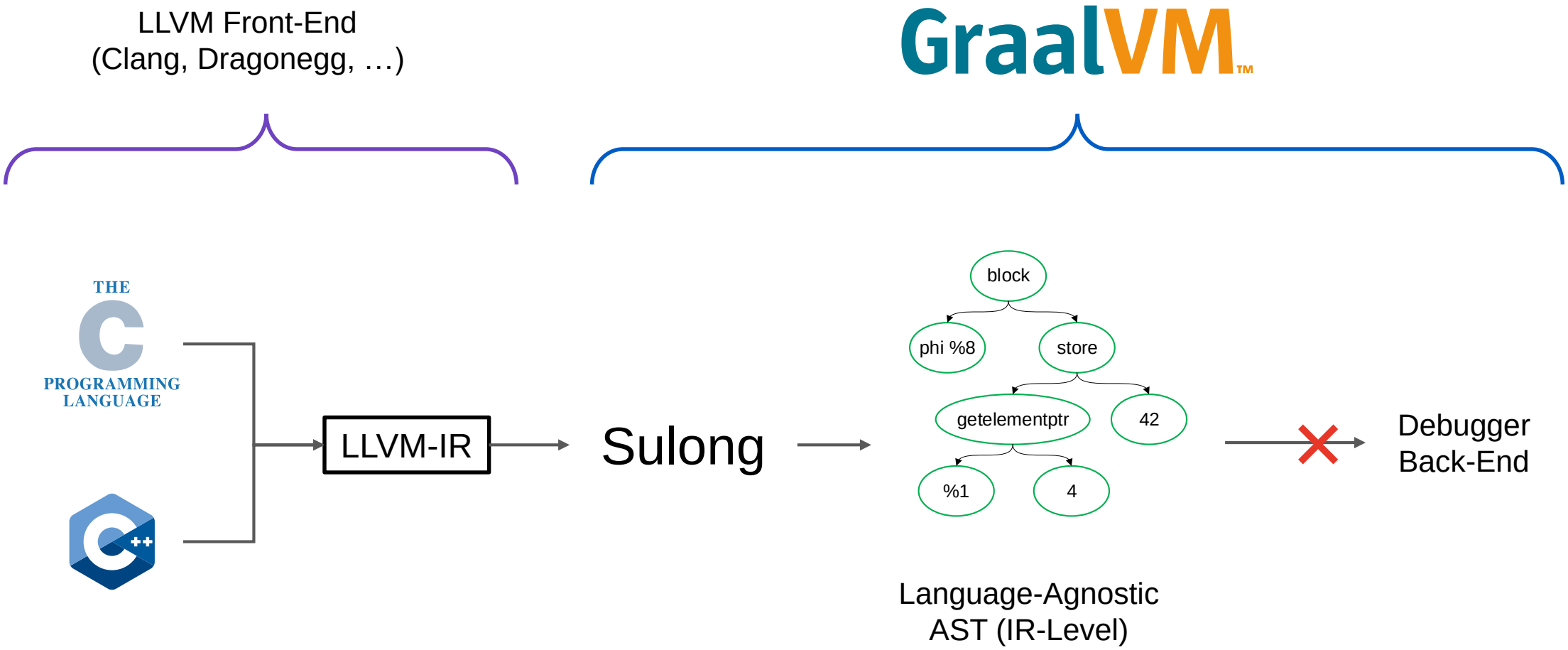
GRAALVM FOR DYNAMIC LANGUAGES



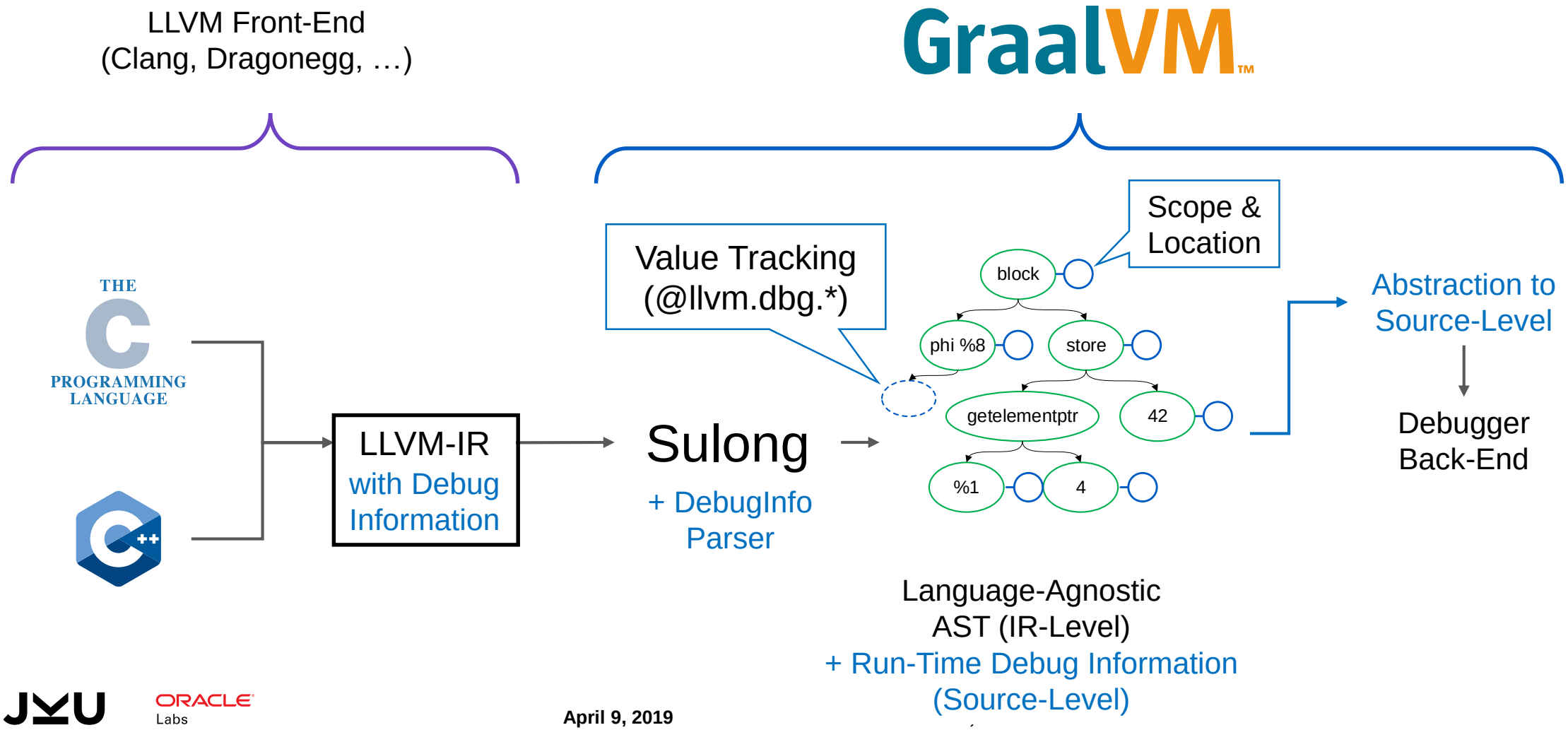
GRAALVM FOR DYNAMIC LANGUAGES



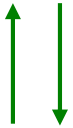
SULONG – THE LLI IN GRAALVM



SOURCE-LEVEL DEBUGGING WITH SULONG

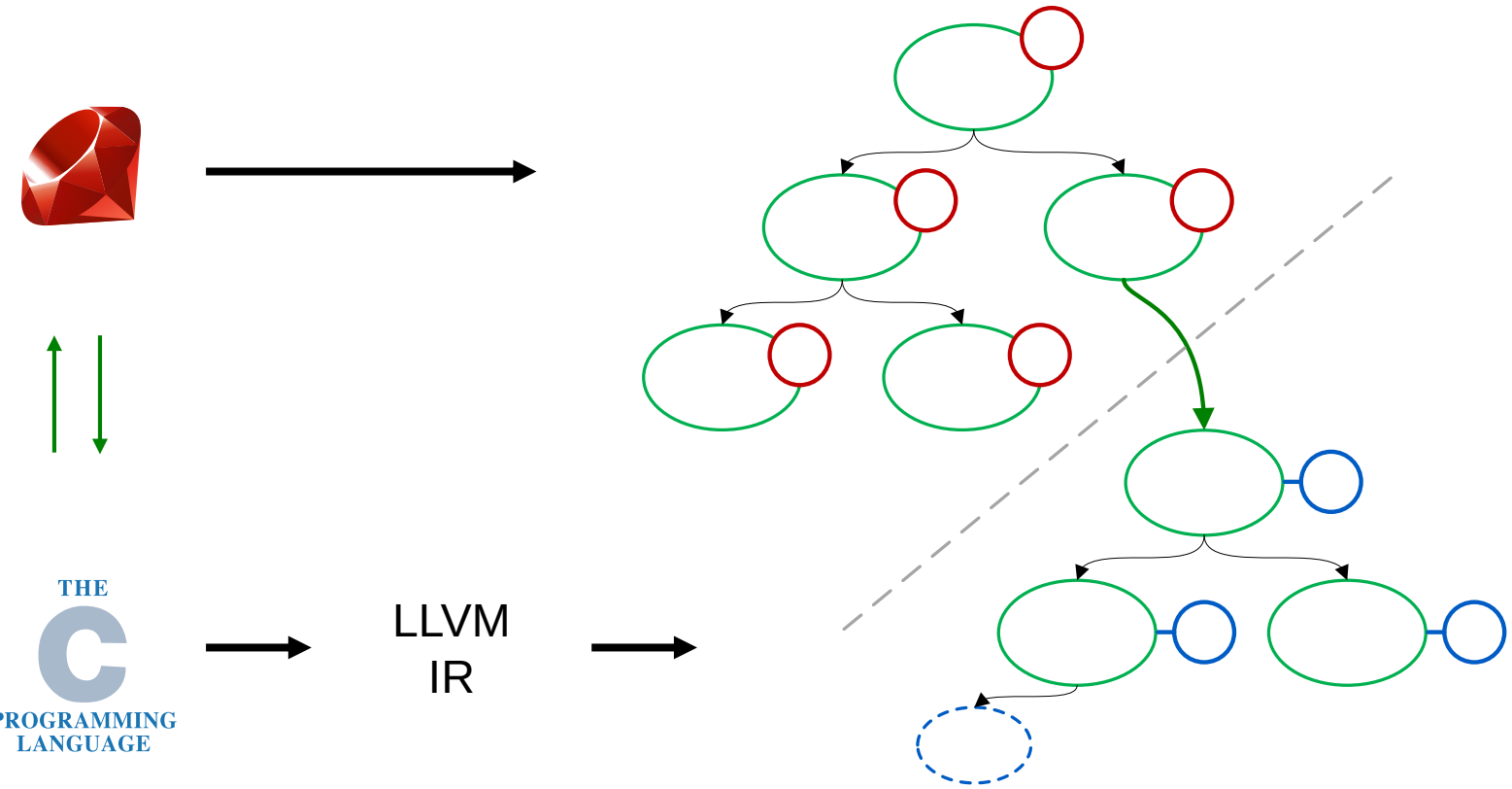


DEBUGGING NATIVE EXTENSIONS OF DYNAMIC LANGUAGE PROGRAMS



Ruby Program
+ Native Extension

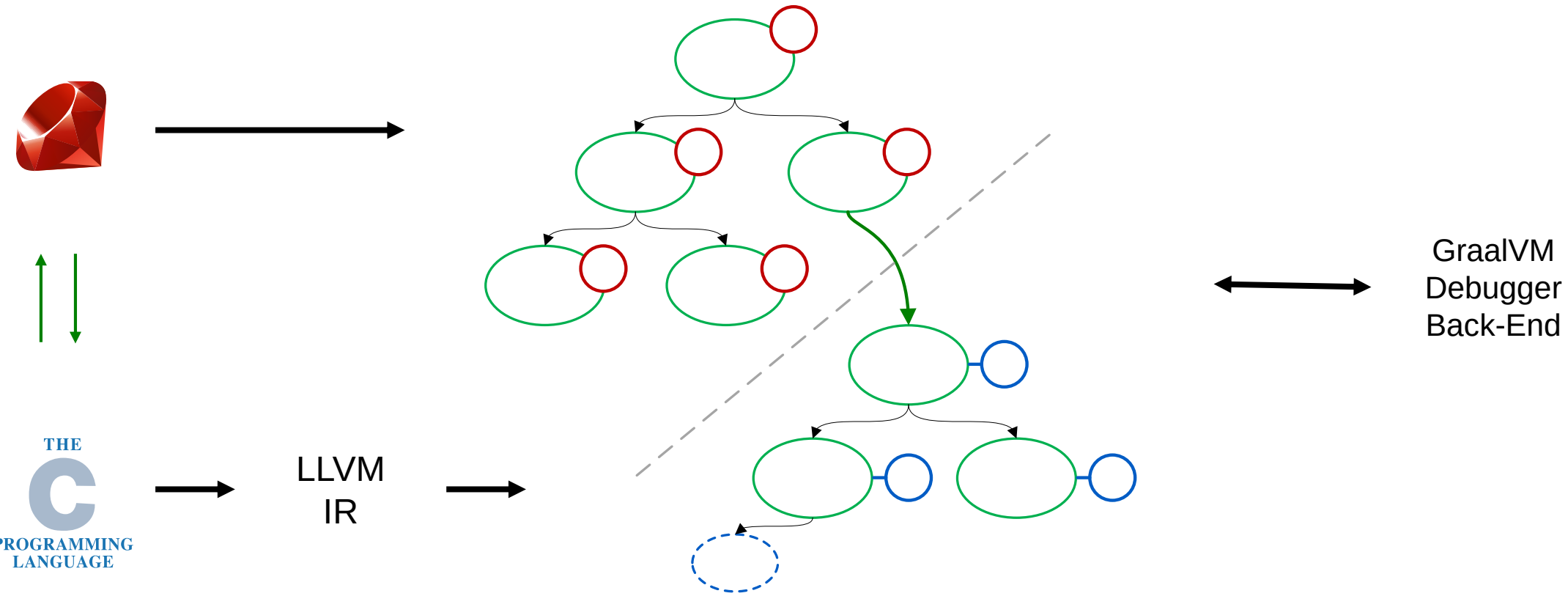
DEBUGGING NATIVE EXTENSIONS OF DYNAMIC LANGUAGE PROGRAMS



Ruby Program
+ Native Extension

One AST
built from multiple language front-ends

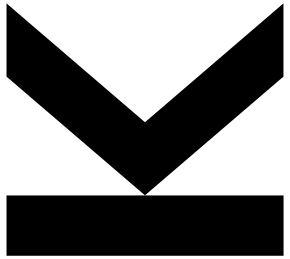
DEBUGGING NATIVE EXTENSIONS OF DYNAMIC LANGUAGE PROGRAMS



Ruby Program
+ Native Extension

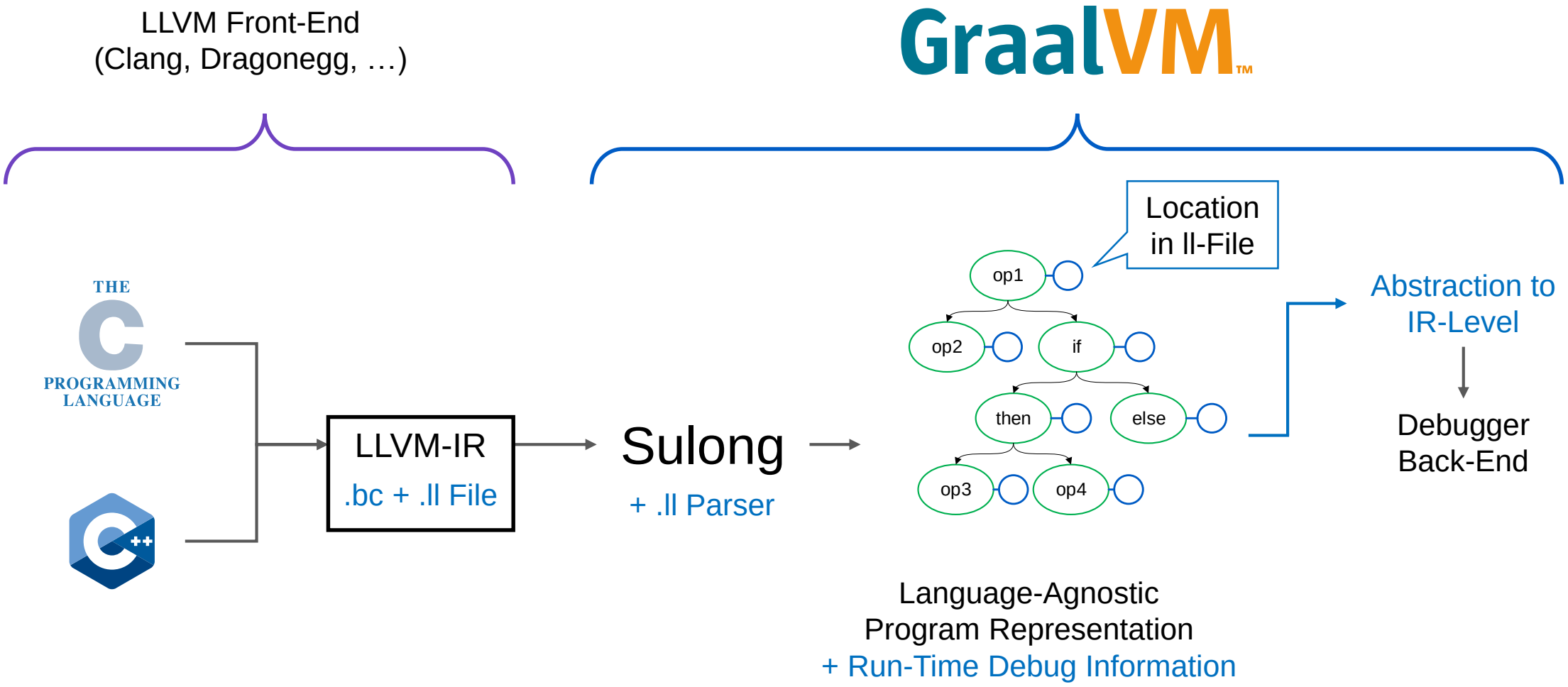
One AST
built from multiple language front-ends

DEMO



Debugging a C-Extension for a Ruby Program with Sulong/TruffleRuby

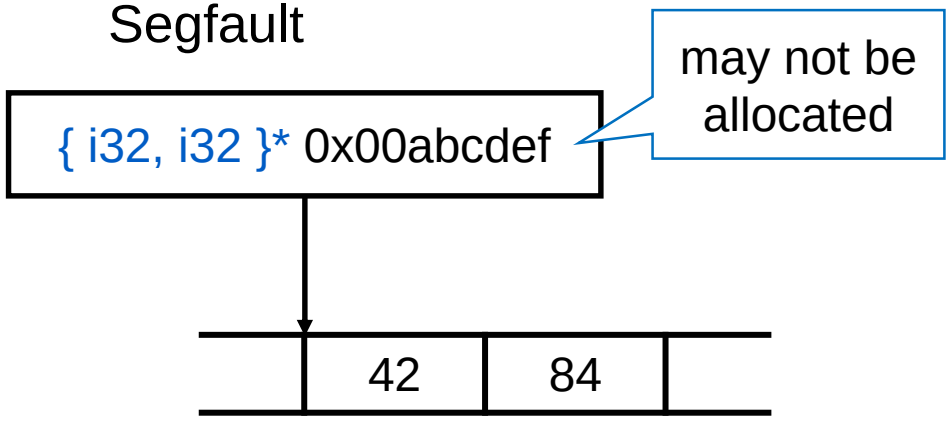
IR-LEVEL DEBUGGING WITH SULONG



MEMORY ACCESS IN SULONG

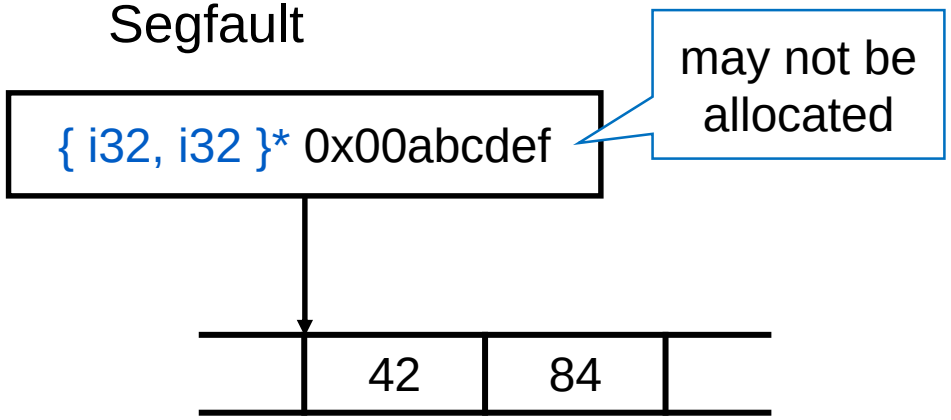
MEMORY ACCESS IN SULONG

- Native Mode
  - All allocations on the native heap
  - Using java.lang.Unsafe
  - Accessing illegal memory leads to Segfault

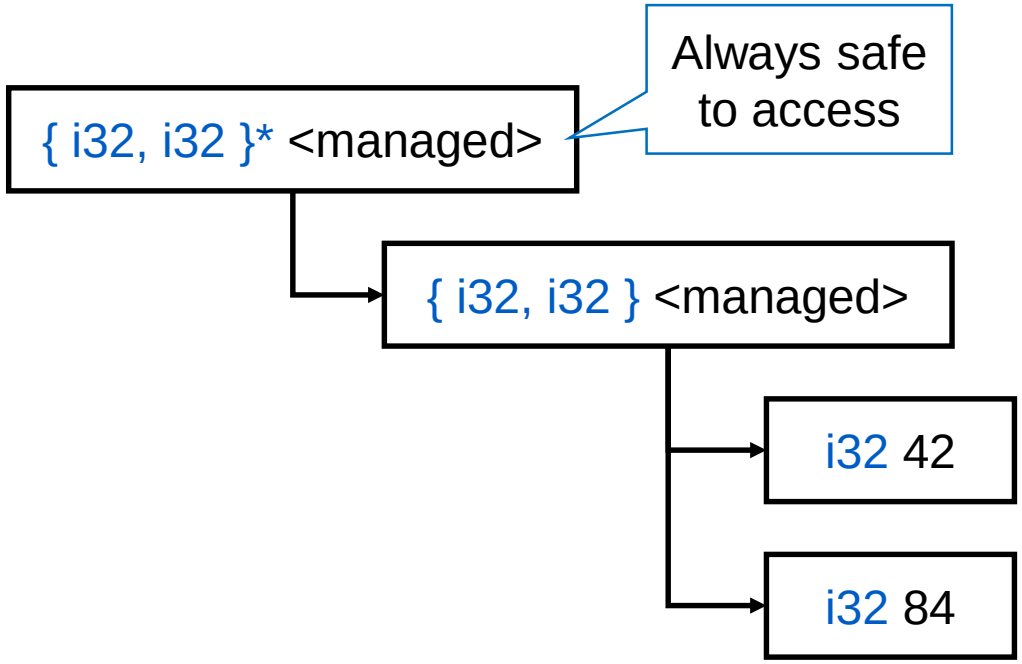


MEMORY ACCESS IN SULONG

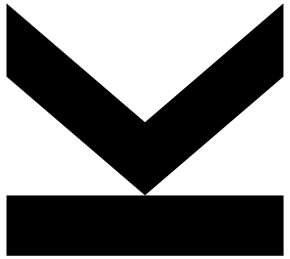
- Native Mode
  - All allocations on the native heap
  - Using java.lang.Unsafe
  - Accessing illegal memory leads to Segfault



- Sandboxed Mode
  - All allocations as Java objects
  - Only in GraalVM Enterprise Edition
  - Guaranteed memory safety



DEMO



Debugging & Tracing a C-Program at IR-Level with Managed Sulong

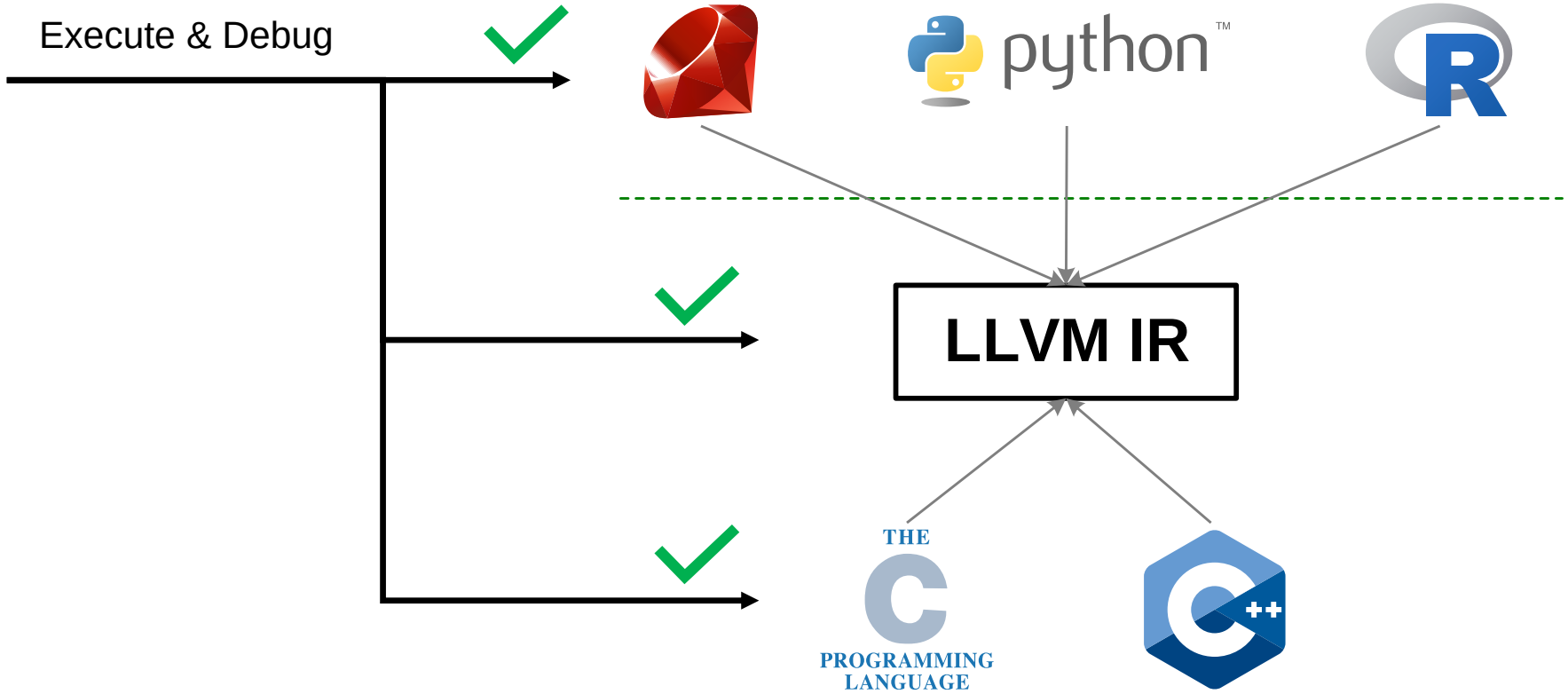
YOU CAN TRY IT YOURSELF

- Download GraalVM at https://www.graalvm.org/

- GraalVM Community Edition is Open Source: https://github.com/oracle/graal

IN CONCLUSION

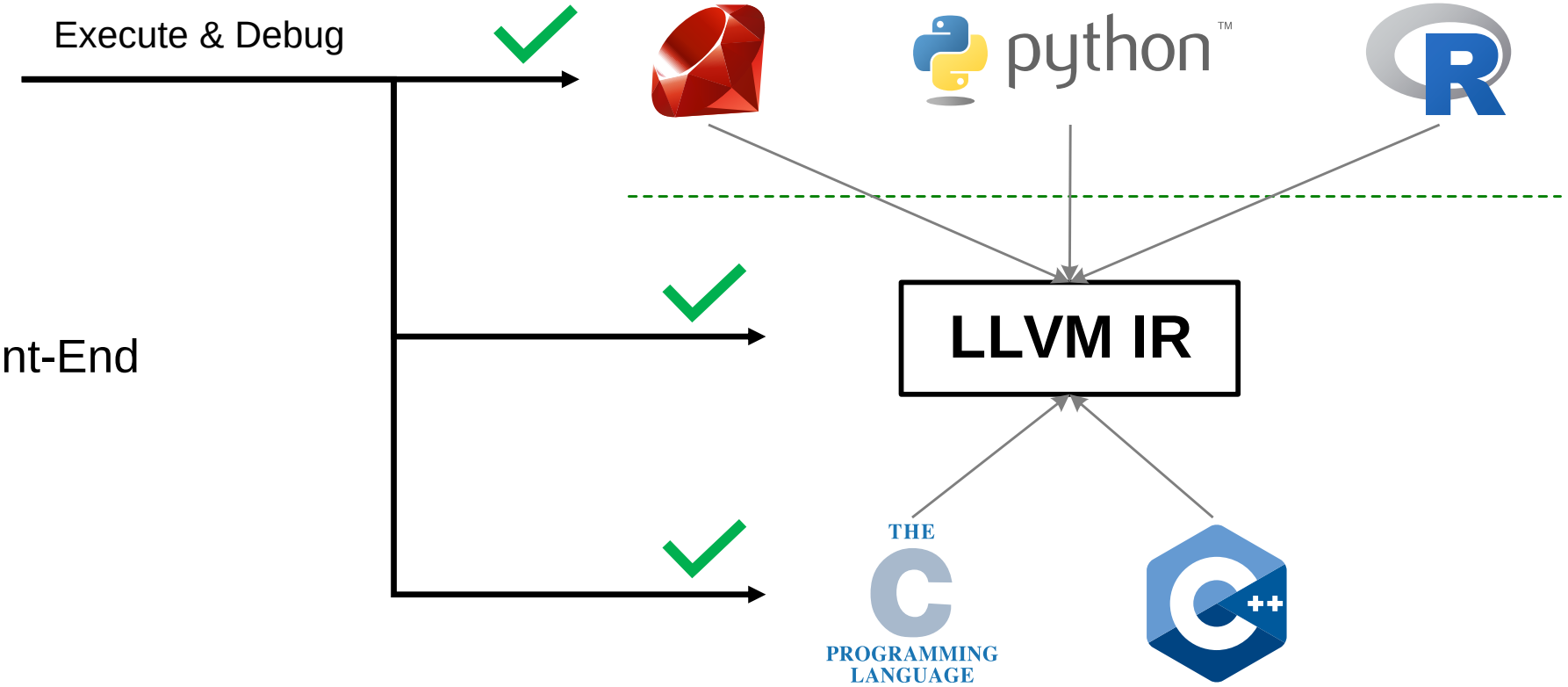
GraalVM



IN CONCLUSION

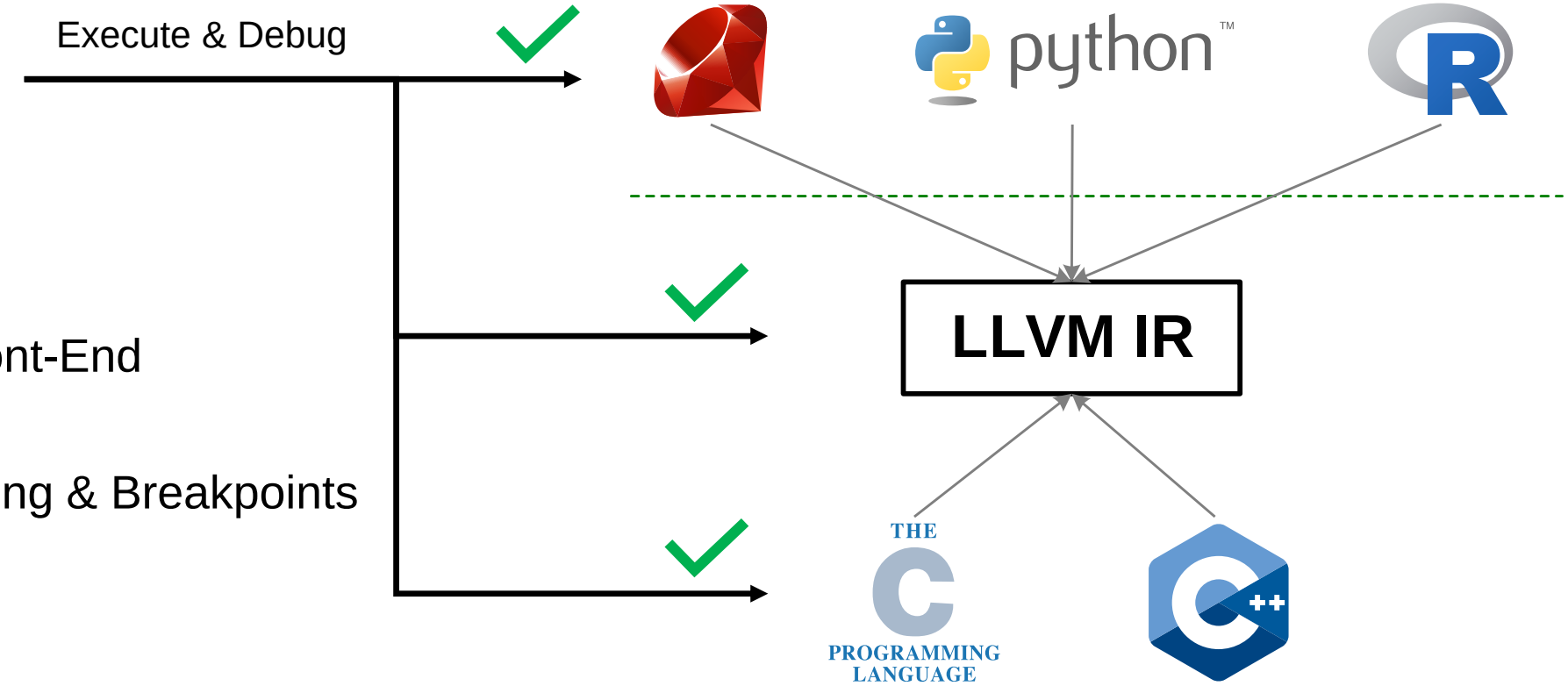
GraalVM™

- Same Debugger Front-End for all Languages



IN CONCLUSION

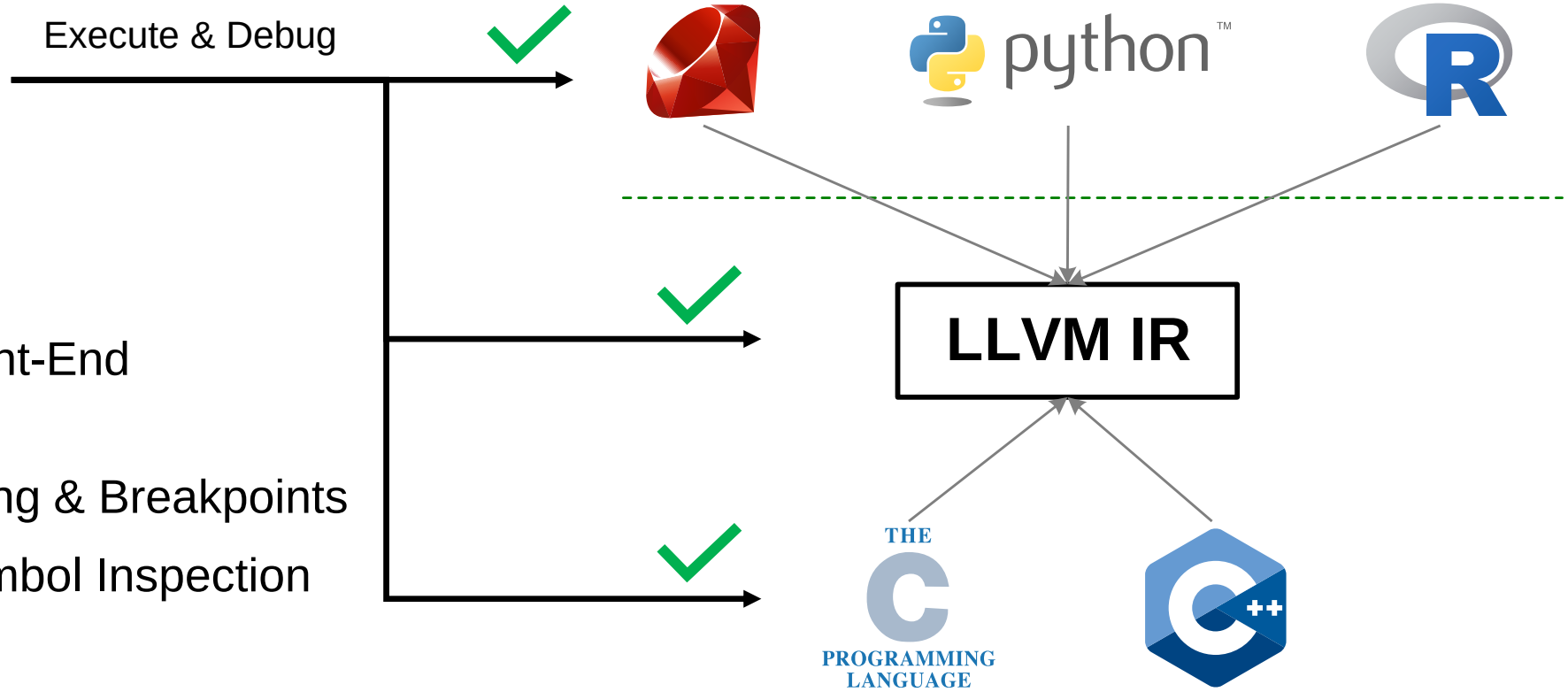
GraalVM



- Same Debugger Front-End for all Languages
- Source-Level Stepping & Breakpoints

IN CONCLUSION

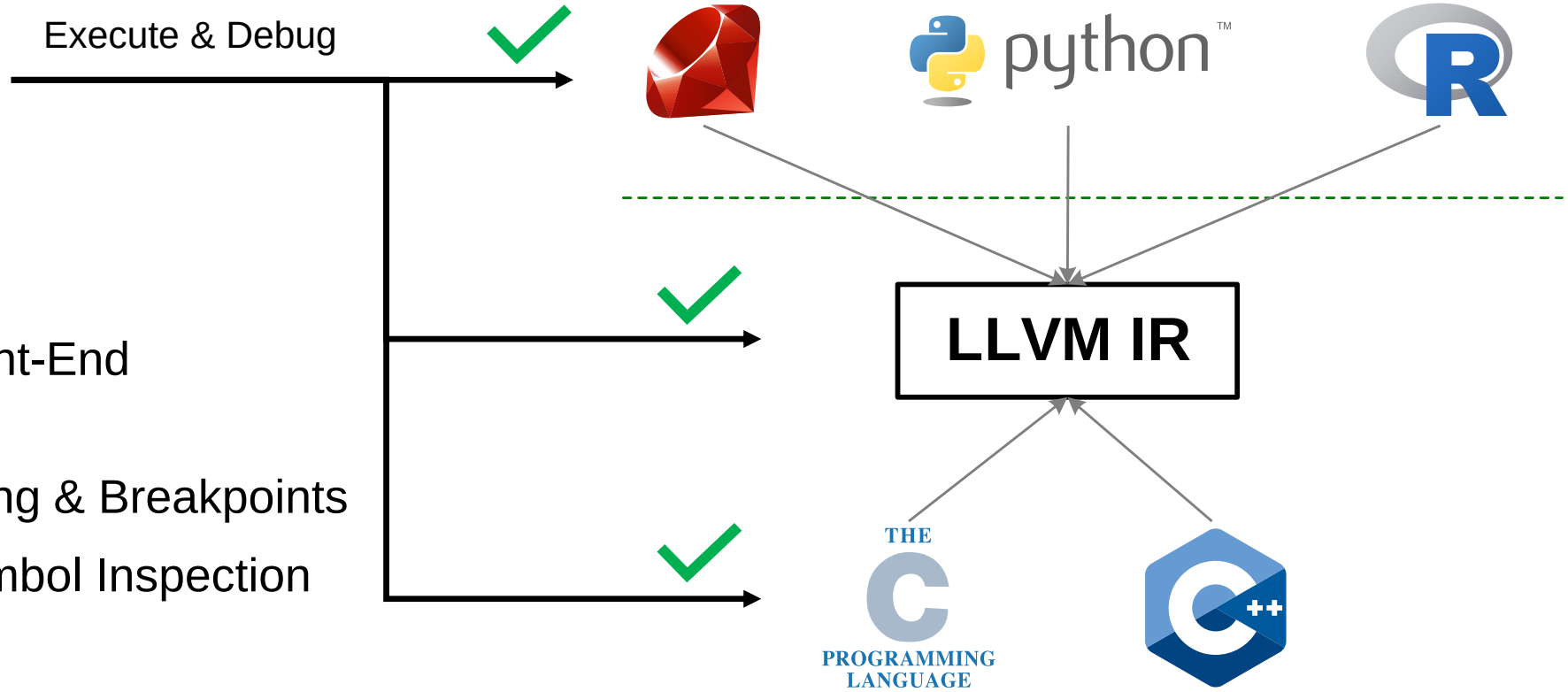
GraalVM



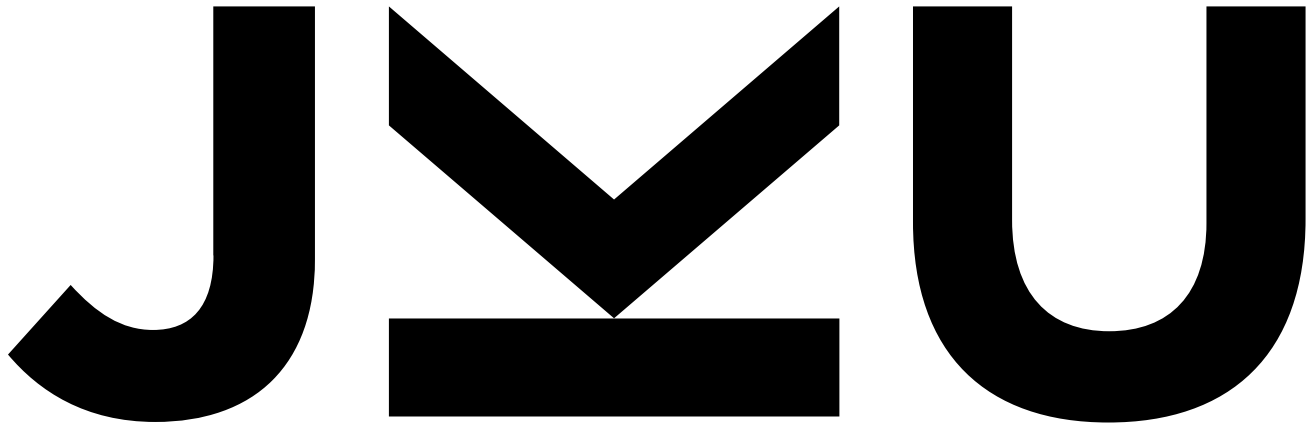
- Same Debugger Front-End for all Languages
- Source-Level Stepping & Breakpoints
- Cross-Language Symbol Inspection

IN CONCLUSION

GraalVM



- Same Debugger Front-End for all Languages
- Source-Level Stepping & Breakpoints
- Cross-Language Symbol Inspection
- IR-Level Debugging


JOHANNES KEPLER
UNIVERSITY LINZ

SYSTEM OVERVIEW