GraalVM

Compiler and JVM Research at JKU

JOHANNES KEPLER
UNIVERSITY LINZ

SSW

Oracle Labs

30.03.2017

Josef Eisl

About Me

- Finished Master's studies January 2014
  - Master's Thesis about CACAO VM
  - Supervised by Prof. Krall
- Since March 2014
  - PhD Student, Research & University Assistant
    @ Institute for System Software, JKU Linz
- Working on the Graal Compiler
  - Backend Optimization, Register Allocation

Who Are We?

Oracle Labs

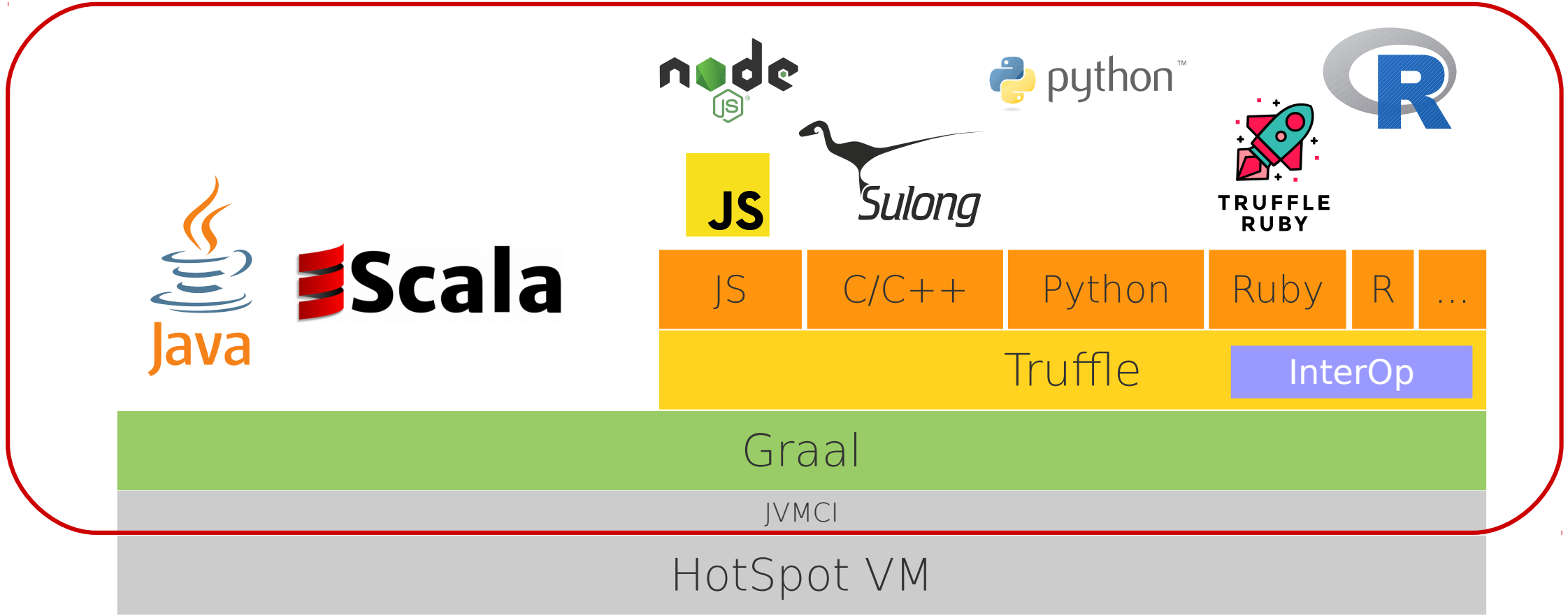
Institute for System Software

Sun/Oracle Collaboration

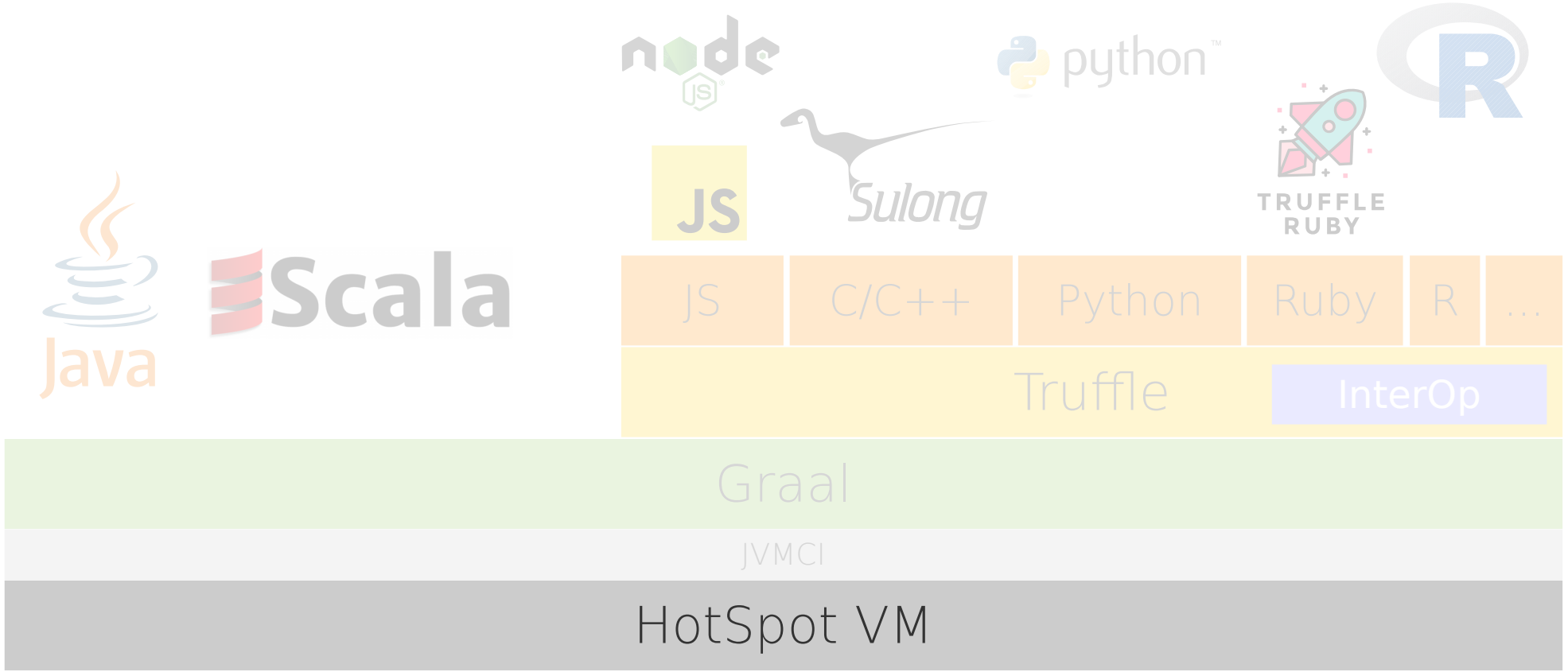
- Started 2001 by Prof. Hanspeter Mössenböck
  - Sun Microsystems, later Oracle Labs
  - Dynamic compiler and JVM research
- Many artifacts now part of Java
  - Client Compiler: Escape Analysis, Register Allocator, …
  - **JVM Compiler Interface in Java 9**
- Current research
  - Graal Compiler
  - Truffle Language Implementation Framework

Oracle Labs

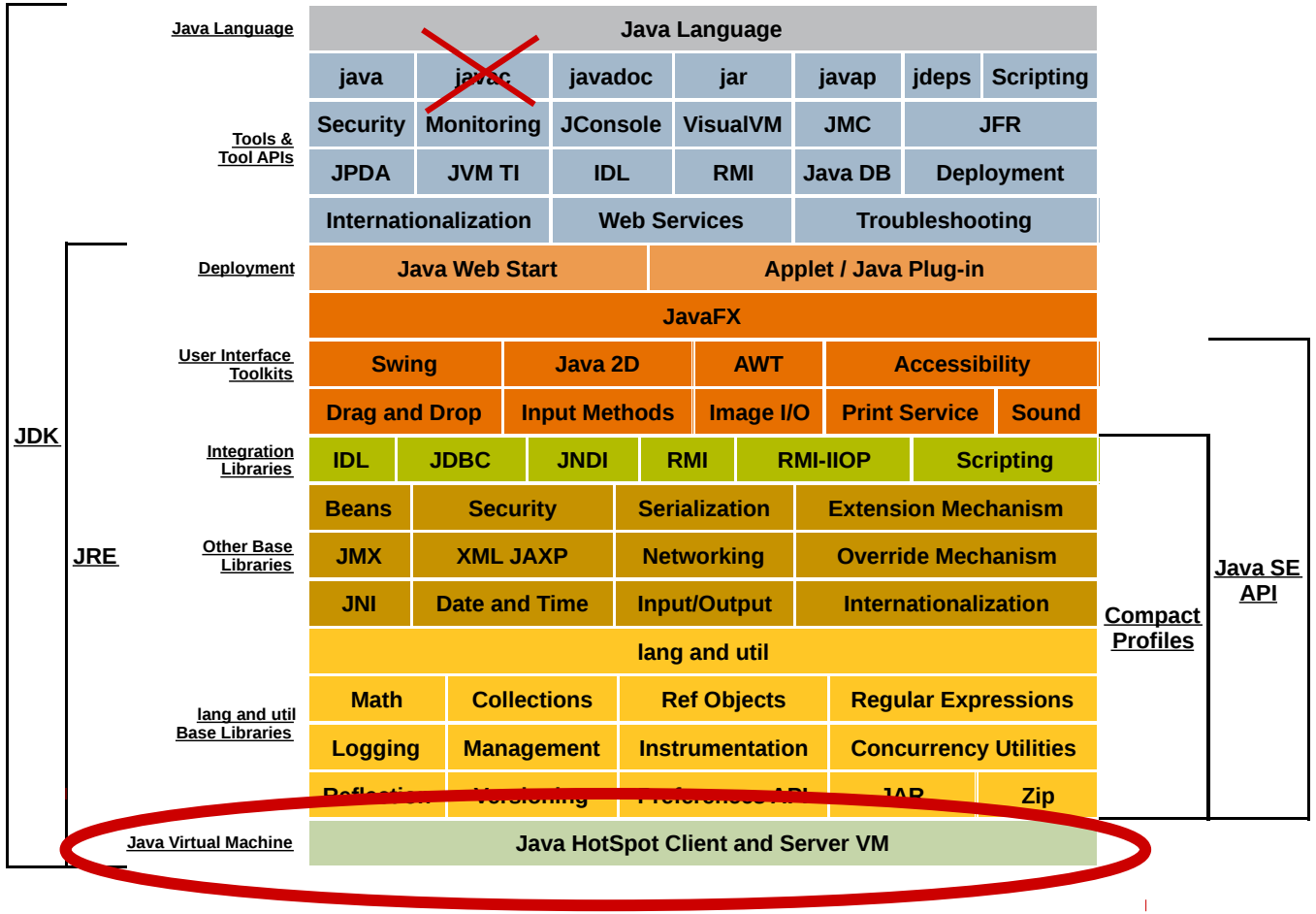
GraalVM Stack



HotSpot VM



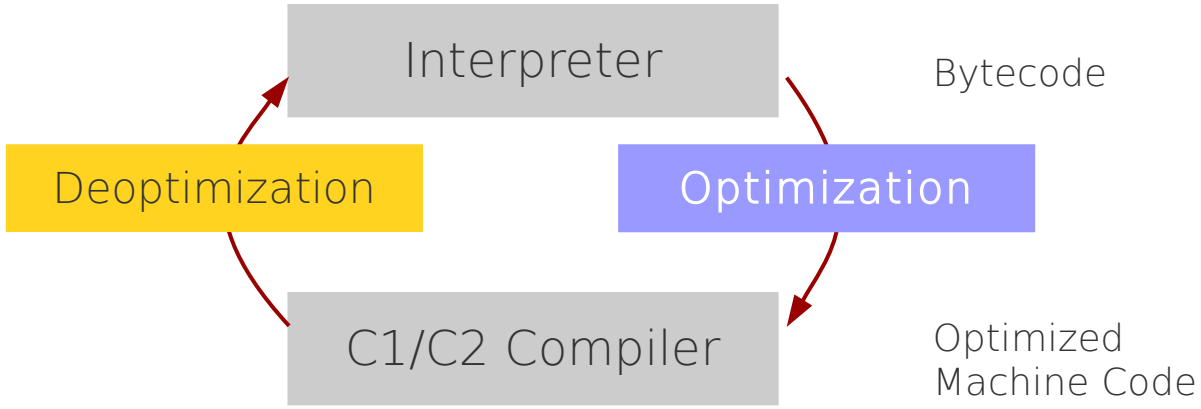
Java Platform



http://docs.oracle.com/javase/8/docs/index.html

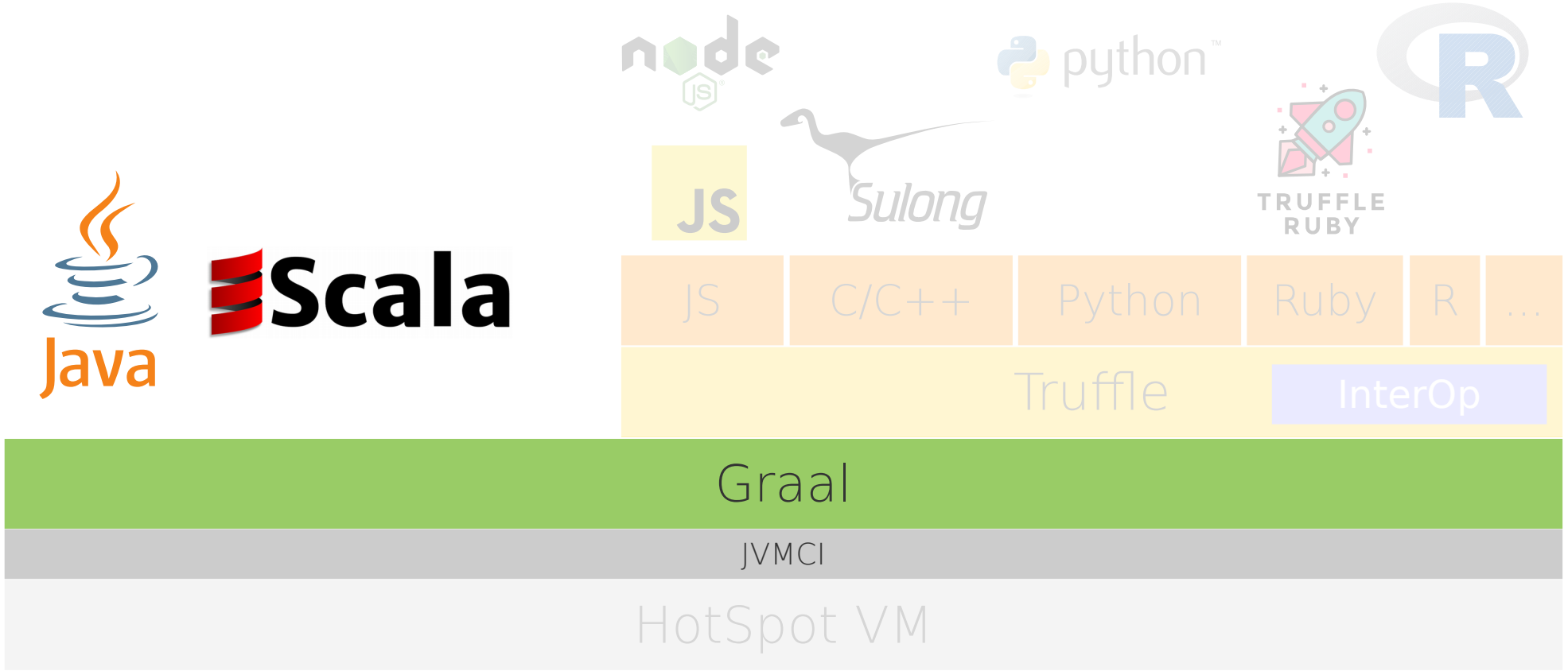
HotSpot VM

- Virtual Machine for Java
  - Execute Java Bytecode (class files)
  - Runtime (Garbage Collection, …)
  - Written in C++
- Execution modes
  - Bytecode Interpreter
    - Immediate start up, slow execution
  - Just-in-Time Compiler (JIT)
    - Important Methods only
    - Slow start up, fast execution



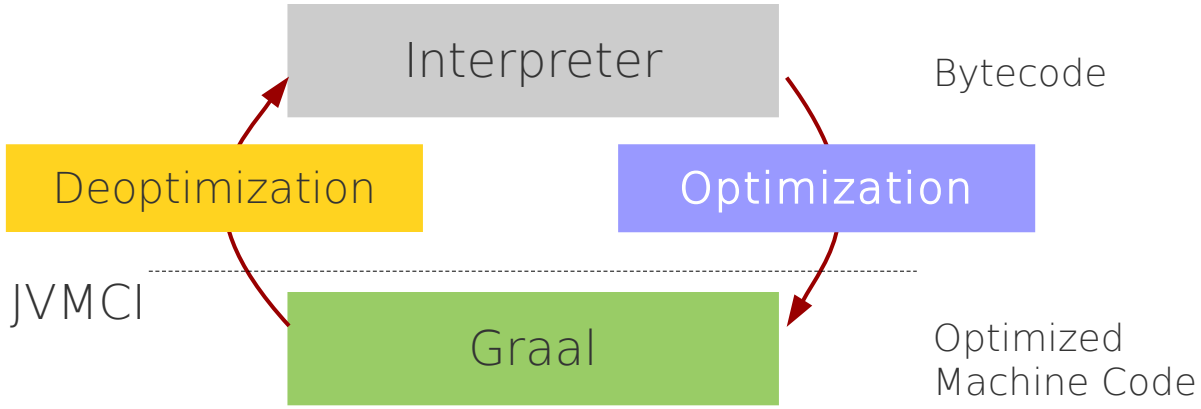

```
$ java -version

java version "1.8.0"
Java(TM) SE Runtime Environment
Java HotSpot(TM) 64-Bit Server VM
```

Graal Compiler



Graal Compiler

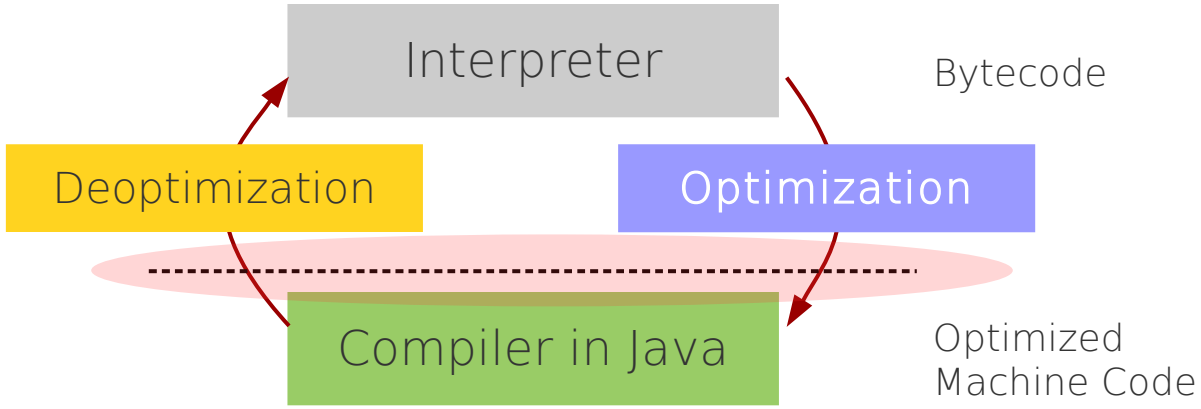
- JIT Compiler for HotSpot
  - GitHub Project: https://github.com/graalvm/graal-core
- Written in Java
  - IDE support
  - Annotations
  - Meta-programming
- Meta-circular



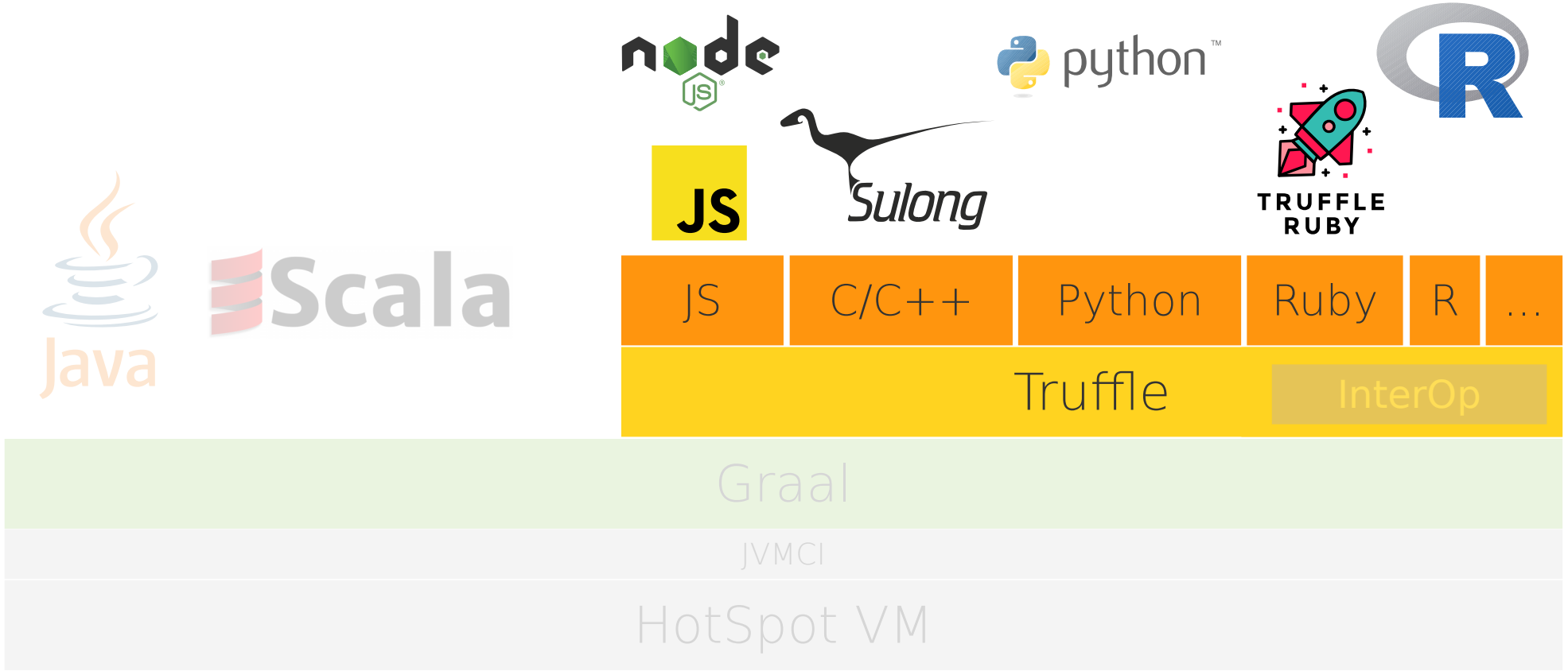

```
$ git clone https://github.com/graalvm/graal-core.git
$ cd graal-core
$ mx build
$ mx vm -version

java version "1.8.0"
Java(TM) SE Runtime Environment
Java HotSpot(TM) 64-Bit Server VM (jvmci-0.25)
```

Java-Level JVM Compiler Interface (JEP 243)

- Compiler **for** Java **in** Java
- Part of Java 9
  - http://openjdk.java.net/jeps/243
- Research Artifact → Product



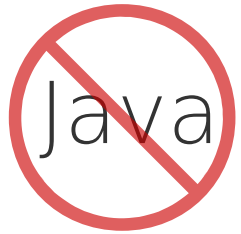
Truffle – Language Implementation Framework



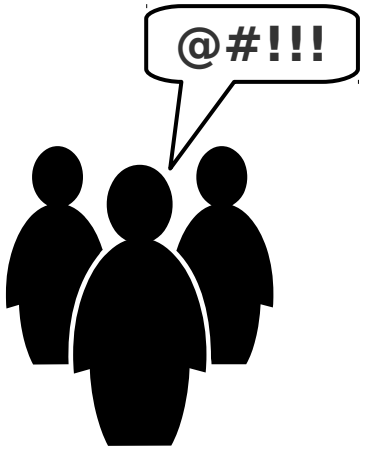
“Write Your Own Language” (Legacy Approach)

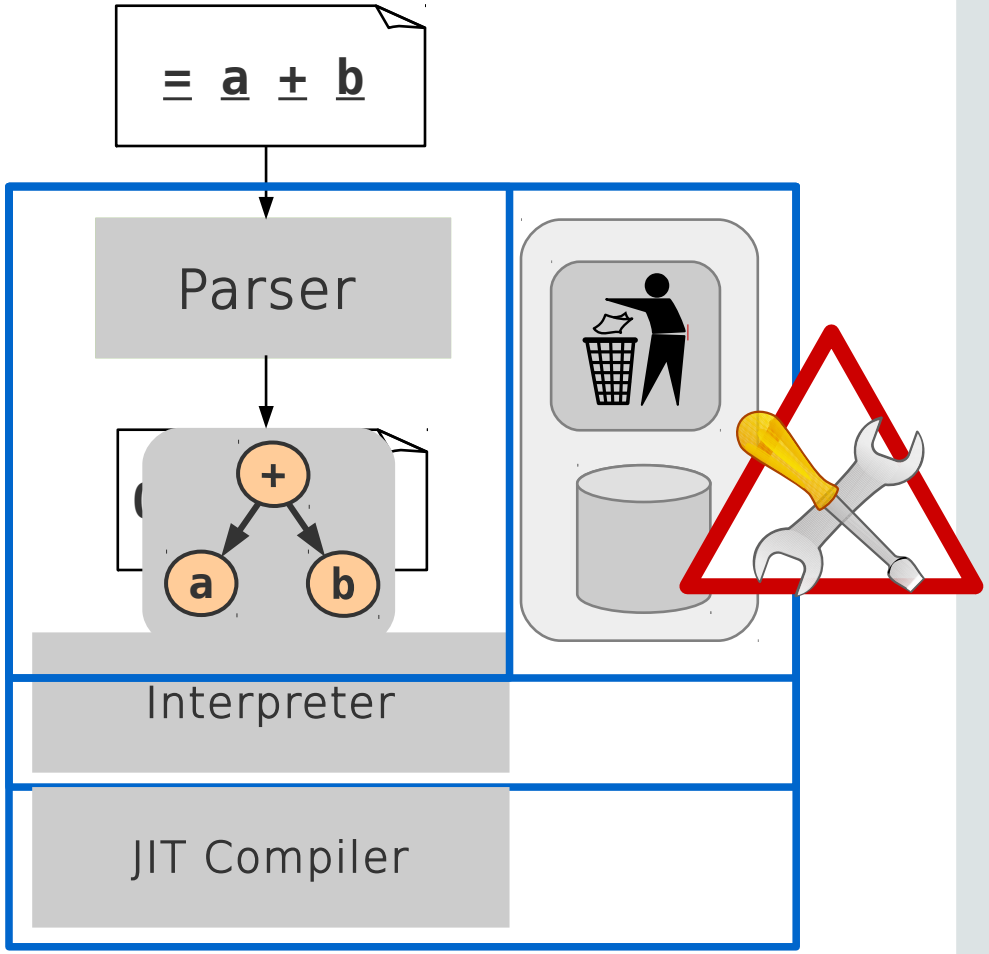
- Parser, Abstract syntax tree (AST) Interpreter
- Write a “real” VM
  - Rewrite in C/C++ (Still using AST interpreter)
  - Add runtime system, Garbage Collection, …
- People start using it

  … and complain about performance
- Define bytecode and write bytecode interpreter
- Performance is still bad
  - Write a JIT compiler
  - Improve the garbage collector, …
  - Repeat



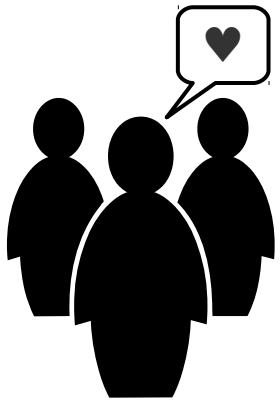
C/C++



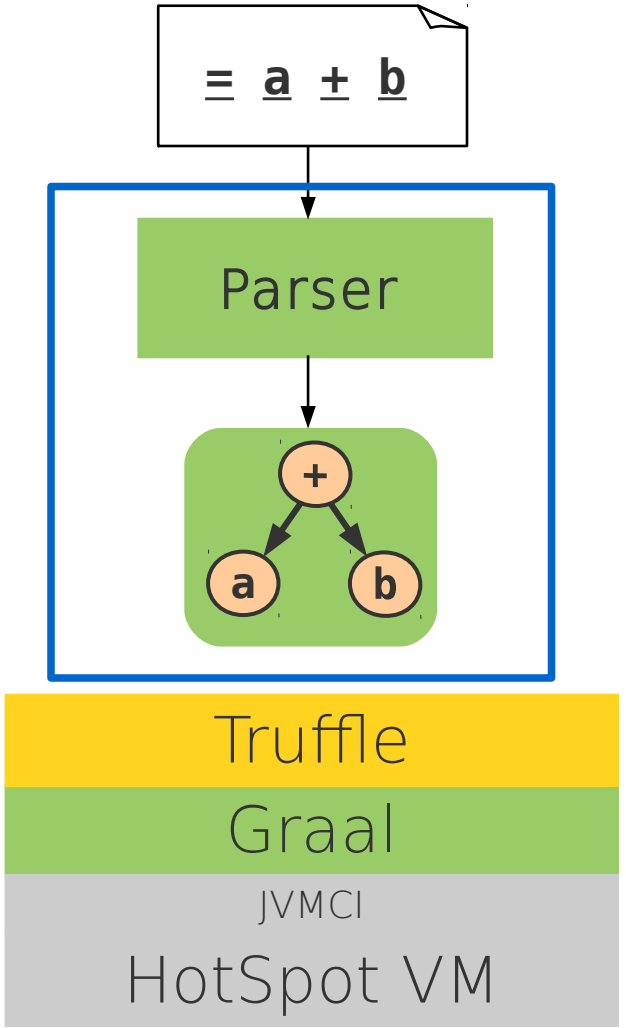


“Write Your Own Language” (Truffle Approach)

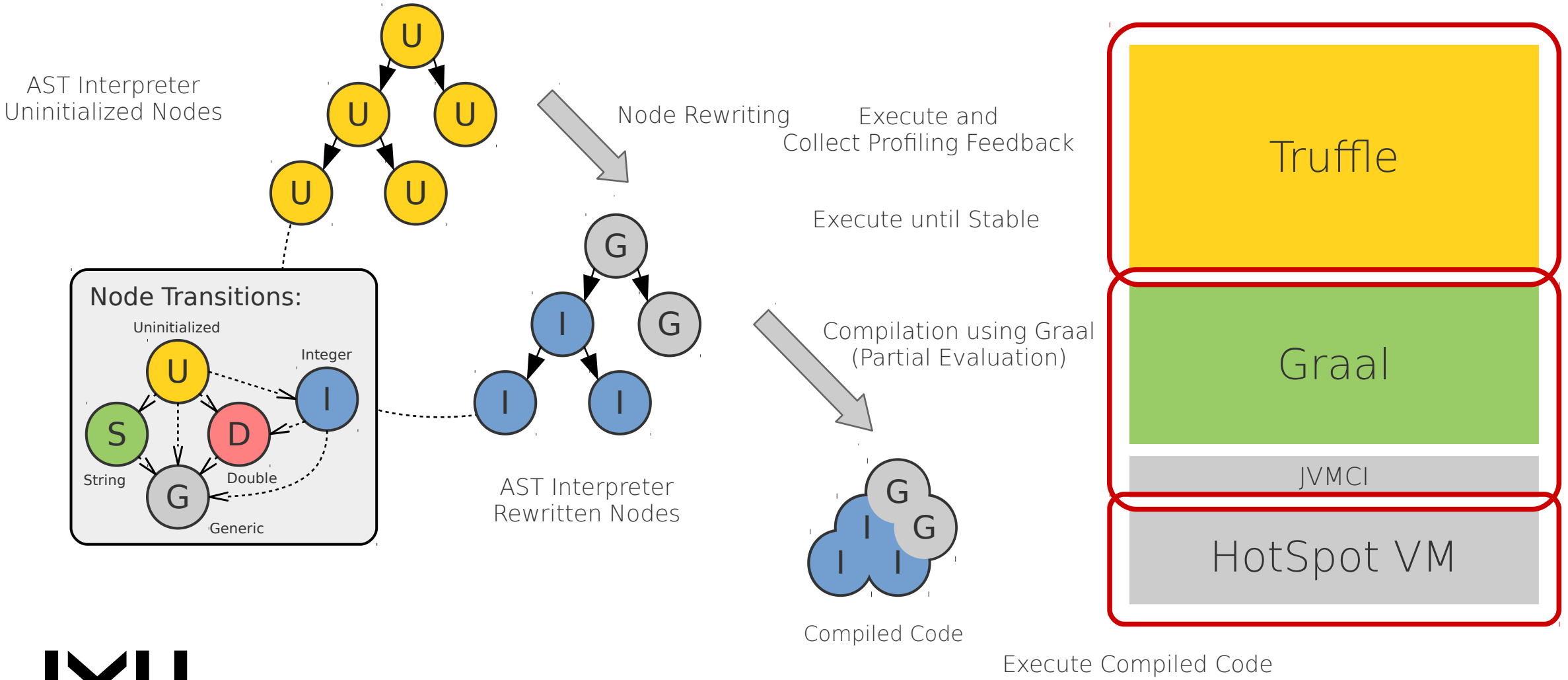
- Parser, Abstract syntax tree (AST) Interpreter
  - Execute it with Truffle/GraalVM
- People start using it
- … and it is already fast!



Java
(Truffle API)



Speculate and Optimize ...



Truffle Implementations

LLVM IR (C/C++/...) - JKU, Oracle Labs

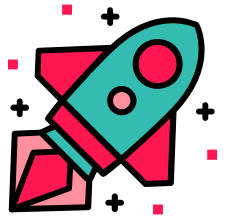
https://github.com/graalvm/sulong

JavaScript – JKU, Oracle Labs

Not Open Sourced (yet)

Oracle Technology Network

(Technology Preview)

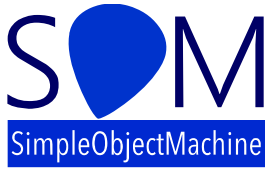


TruffleRuby - JKU, Oracle Labs

https://github.com/graalvm/truffleruby

ZipPy - UCI Irvine

https://bitbucket.org/ssllab/zippy



Smalltalk - JKU (Stefan Marr)

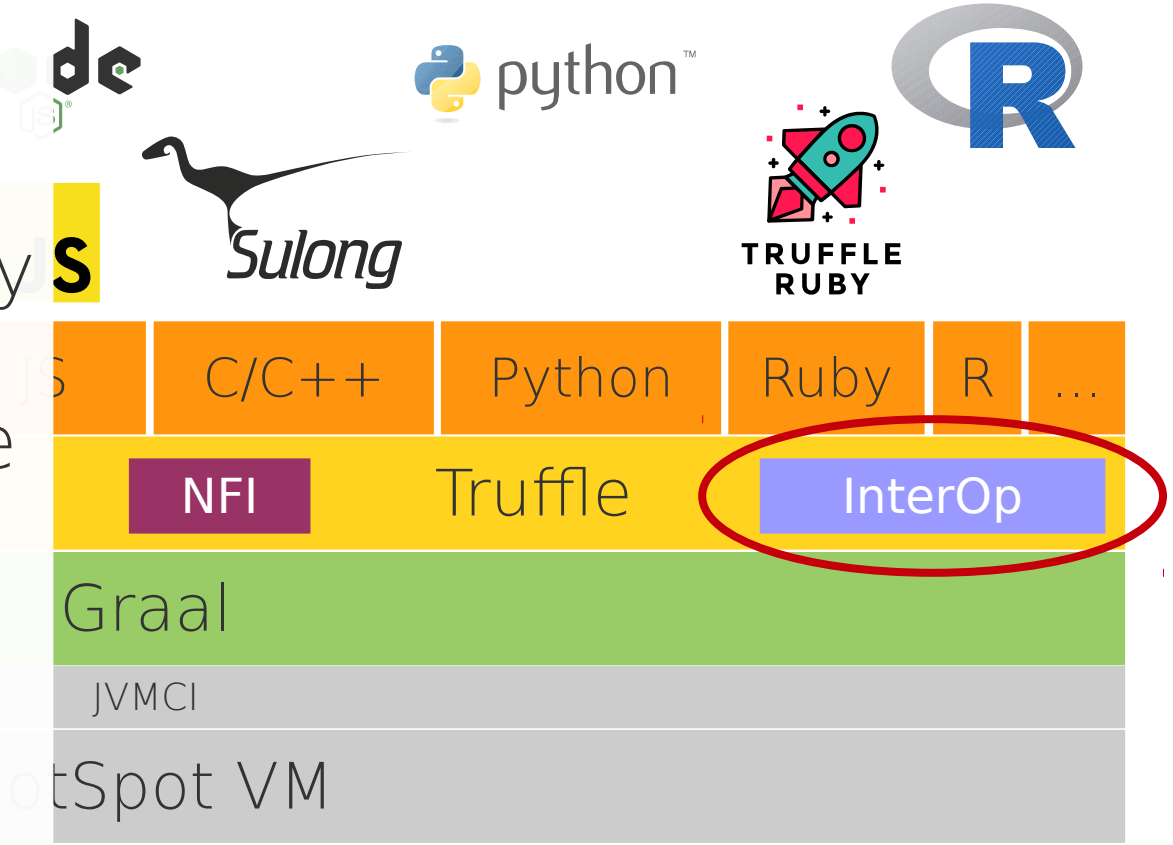
https://github.com/SOM-st/TruffleSOM

FastR - JKU, Oracle Labs

https://github.com/graalvm/fastr

Further Topics

- Language Interoperability
- Native Function Interface
- Substrate VM
  - Paul's Talk



Christian Humer's GraalVM Demo at JavaOne 2015: https://youtu.be/lYGfD2H99Ls

Get in Contact

- Projects, Internships, PhD Positions, …
- Me
  - Mail: josef.eisl@jku.at
  - Twitter: @zapstercc
- GraalVM
  - Github: https://github.com/graalvm

http://goo.gl/cVhX3U