# Just-in-Time Compilation of the Dependently-Typed Lambda Calculus

Jakub Zárybnicky <xzaryb00@stud.fit.vutbr.cz>

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2. 612 66 Brno - Královo Pole
login@fit.vutbr.cz


February 2, 2021

- Writing programming languages

| Compiler | Interpreter |
|---|---|
| **Fast** | Slow |
| Hard(er)-to-create | **Easy-to-create** |

- Writing programming languages

| Compiler | Interpreter |
|---|---|
| **Fast** | Slow |
| Hard(er)-to-create | **Easy-to-create** |

- Third way - JIT compilation, write an interpreter, maybe add runtime optimizations

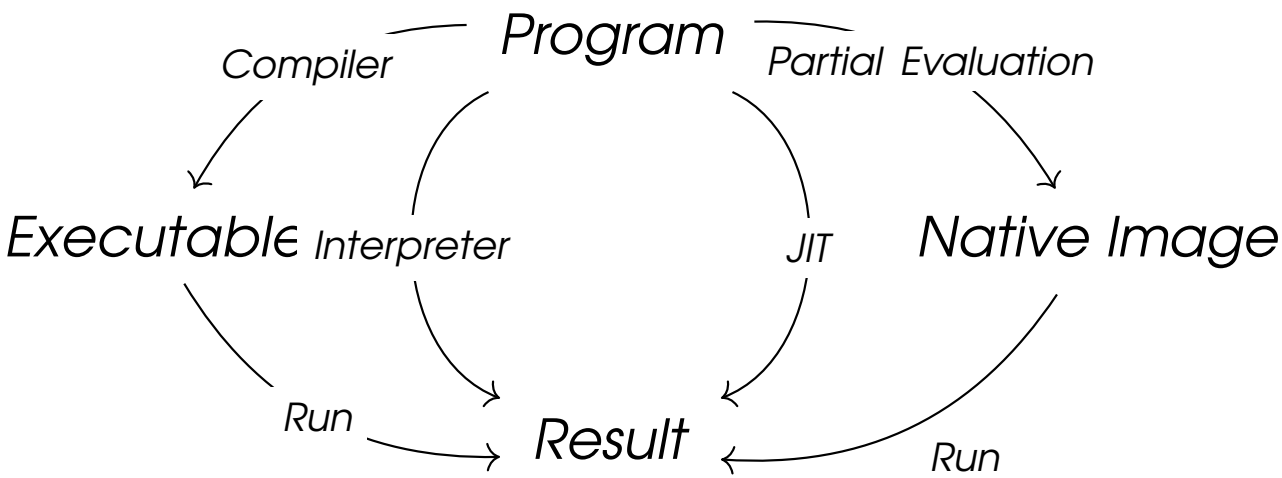


Figure: Methods of program execution

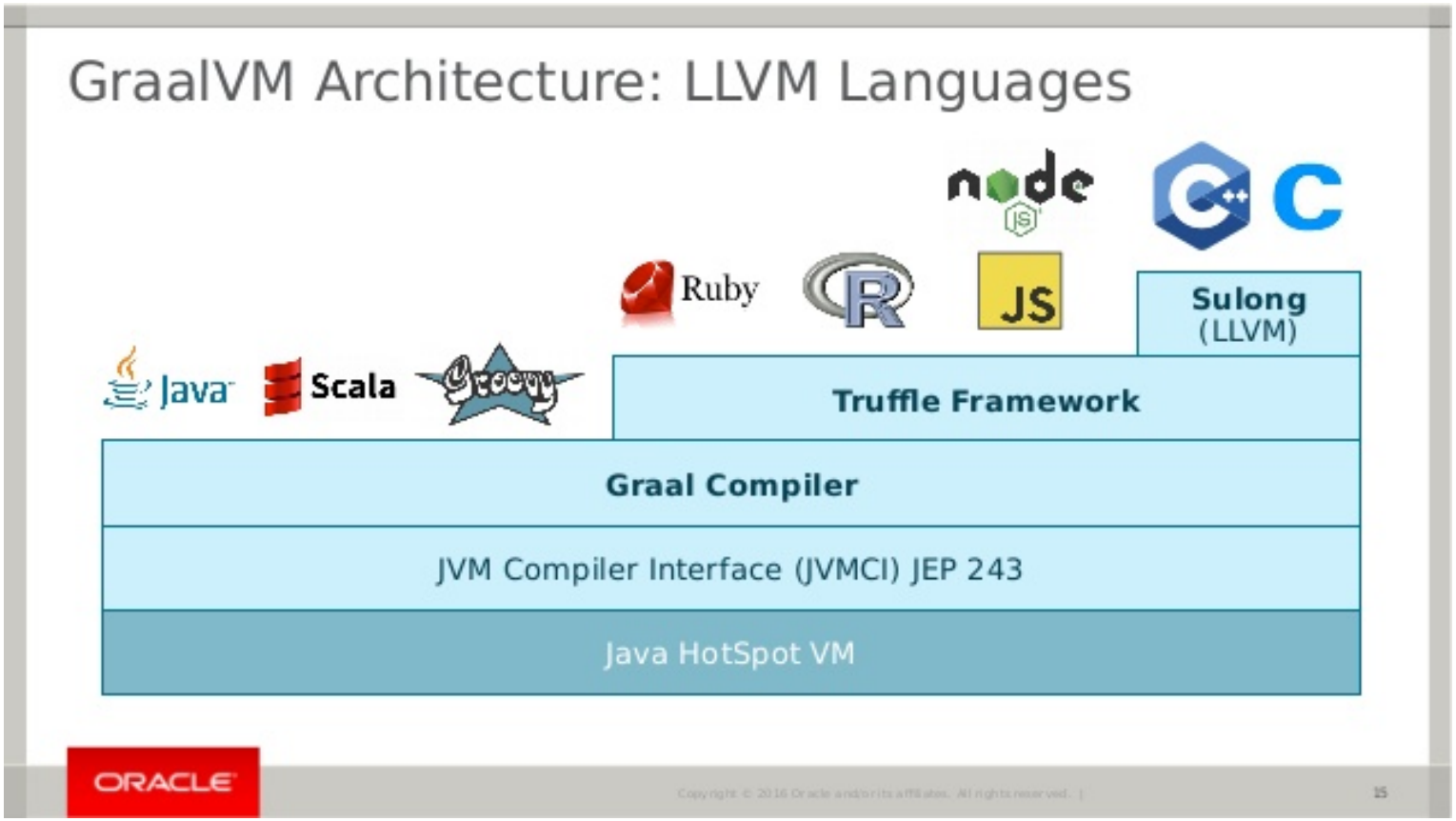


Figure: GraalVM and Truffle (source: oracle.com)

# JIT compilation

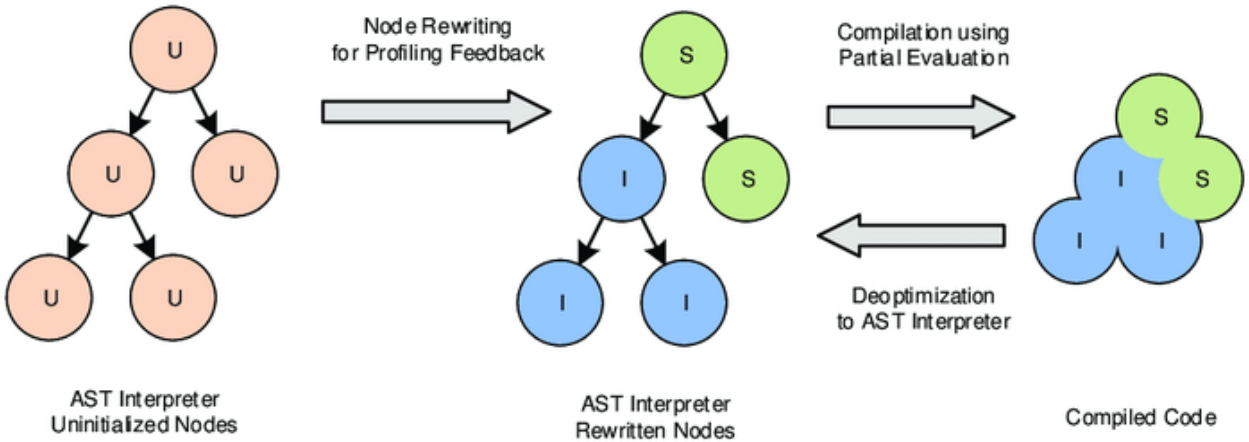


Figure: Node rewriting (src: Truffle DSL, 10.13140/RG.2.2.23639.52646)

$$
\begin{array}{lcll}
e & ::= & x & \text{variable} \\
 & | & e_1\ e_2 & \text{application} \\
 & | & \lambda x.e & \text{abstraction} \\
 & | & x : \tau & \text{annotation} \\
 & | & * & \text{the type of types} \\
 & | & \forall x : \rho.\rho' & \text{dependent function space}
\end{array}
\tag{1}
$$

Figure: Dependently typed lambda calculus

```
let const = (\ a b x y -> x)
    :: forall (a :: *) (b :: *) . a -> b -> a
```

# Assignment

1. Investigate dependent types, simply-typed and dependently-typed lambda calculus, and their evaluation models (push/enter, eval/apply).
2. Get familiar with the Graal virtual machine and the Truffle language implementation framework.
3. Create a parser, and an interpreter for a selected language based on dependently-typed lambda calculus.

# Implementations

- Interpreter (plain Kotlin), 100% done
- JIT compiler (Kotlin + Truffle), 10% done
- LLVM compiler (Kotlin + LLVM bindings), 5% done

Thank You For Your Attention !