A Case Study: Execution of LionWeb nodes in Truffle Language Framework

Erkan Diken / F1RE

F1RE forge dsls

2VFW

ProxyHands

LangDev CON 2024

Seville 17-19 October, 2024

https://langdevcon.org

What

MPS

Meta-model (M2)

Instance model (M1)

LANGUAGE INTERFACES ON THE WEB

LIONWEB

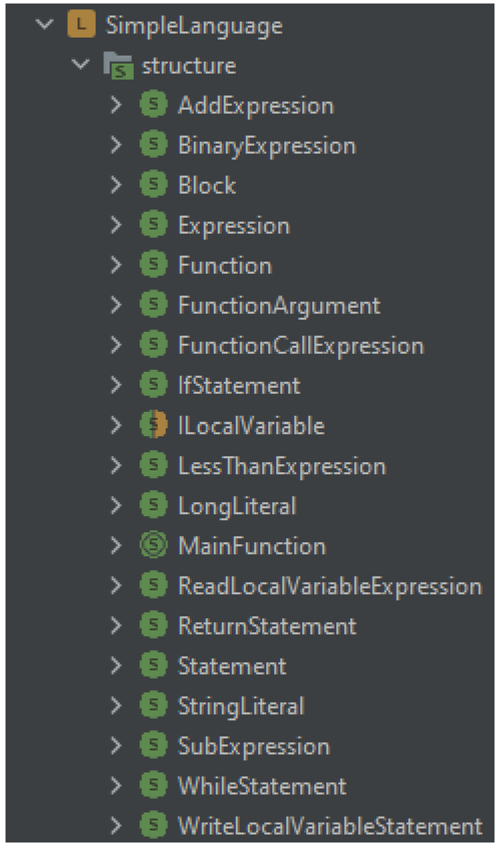
Truffle Language Framework

GraalVM™

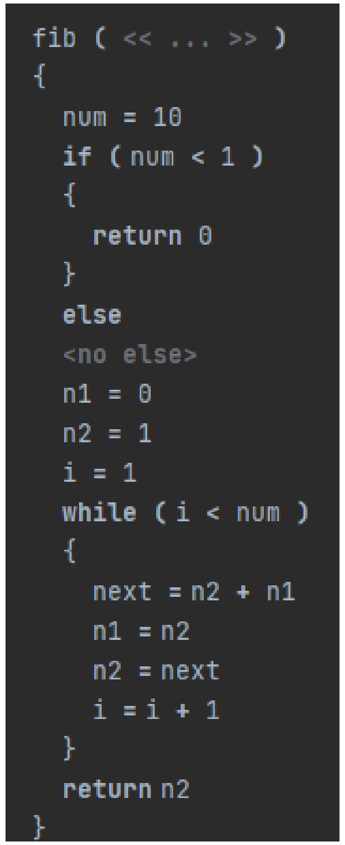
all Java
no MPS

M1.exe

Input models



```
⌄ SimpleLanguage
  ⌄ structure
    > AddExpression
    > BinaryExpression
    > Block
    > Expression
    > Function
    > FunctionArgument
    > FunctionCallExpression
    > IfStatement
    > ILocalVariable
    > LessThanExpression
    > LongLiteral
    > MainFunction
    > ReadLocalVariableExpression
    > ReturnStatement
    > Statement
    > StringLiteral
    > SubExpression
    > WhileStatement
    > WriteLocalVariableStatement
```



```
fib ( << ... >> )
{
  num = 10
  if ( num < 1 )
  {
    return 0
  }
  else
  <no else>
  n1 = 0
  n2 = 1
  i = 1
  while ( i < num )
  {
    next = n2 + n1
    n1 = n2
    n2 = next
    i = i + 1
  }
  return n2
}
```

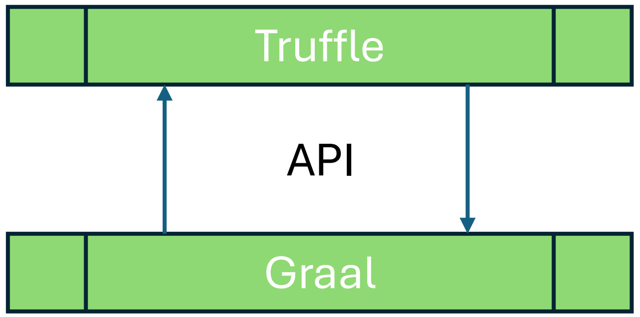
- M2: SimpleLanguage
- M1: nth Fibonacci number

Oracle GraalVM

- GraalVM is another JDK/JVM distribution
- Specifics:
  - Uses open source Graal Compiler as JIT compiler
    - Consumes bytecode and produces machine code
    - Written in Java
  - Truffle language implementation framework
    - A library, written in Java, enables implementation of:
      - new programming language / DSL
      - existing programming language
  - Native-image technology
    - Compiles java code AOT to a binary – a native executable

Truffle Language Implementation Framework

- Truffle framework enables writing AST interpreters for a custom language
  - allows automatic generation of high-performance code from interpreters
    - Graal exposes an API to Truffle and Truffle can ask services from Graal



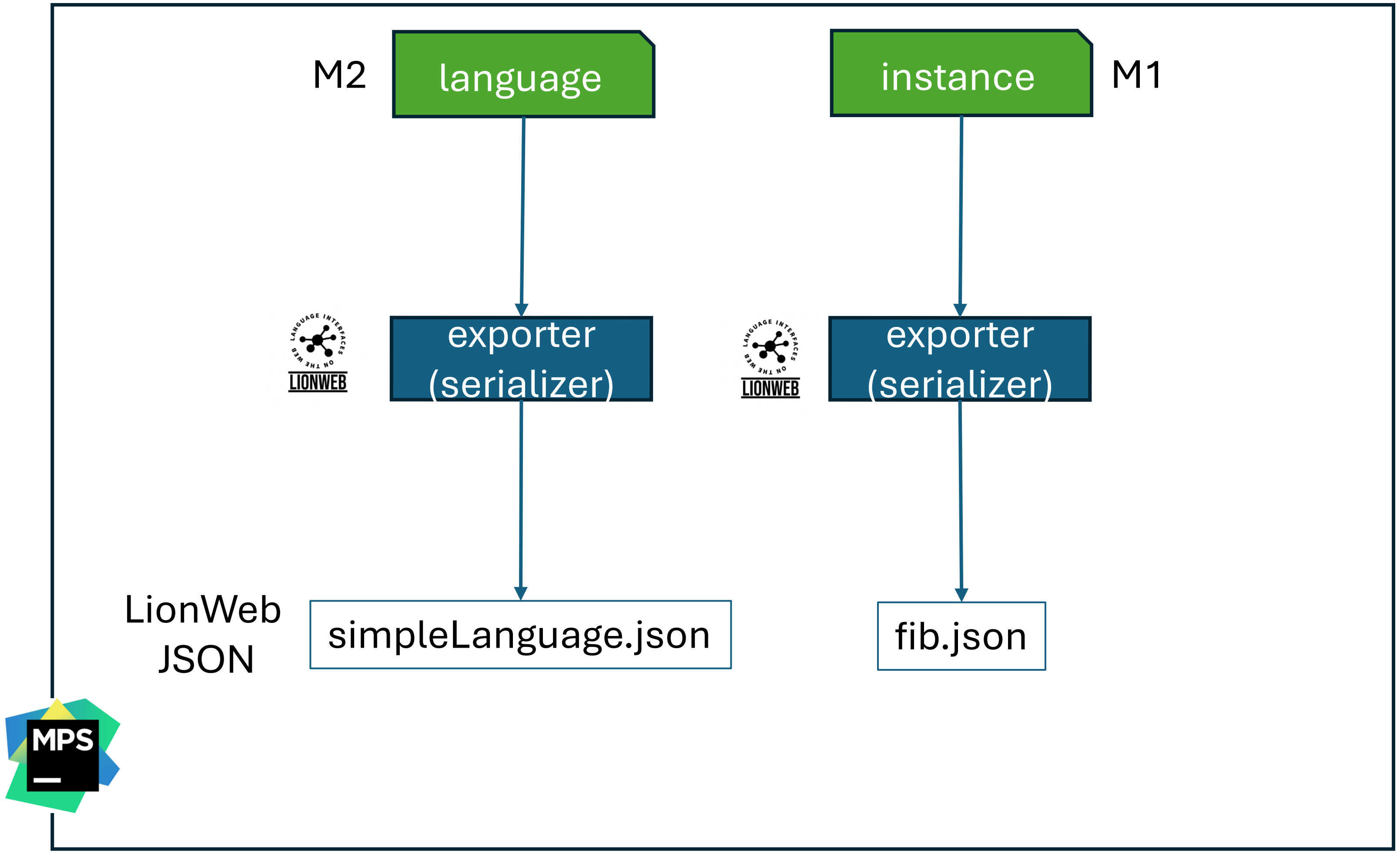
    - AST specialization and partial evaluation of the interpreter with respect to given program and data
    - Compiler optimizations

Motivation

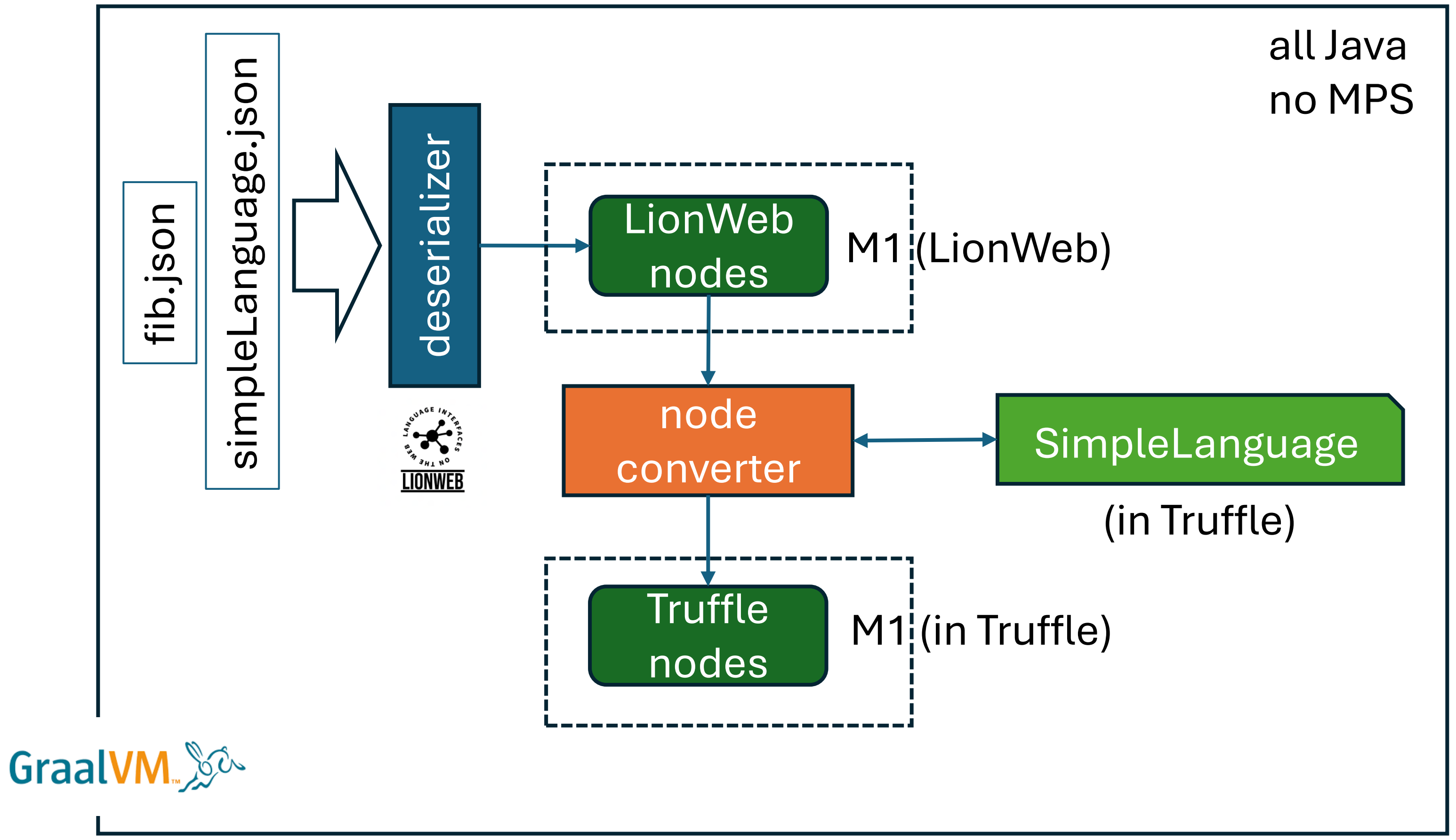
- Add semantic to the model
- Generate production ready target code
- Be independent of any modeling tool
- We use Truffle:
  - To add semantic to the language
  - Target framework for code generation
- Generated Truffle code:
  - Can be run on any JVM
  - AOT-compiled with native-image technology (in GraalVM)
- Use LionWeb to be independent of a modelling tool

Case study

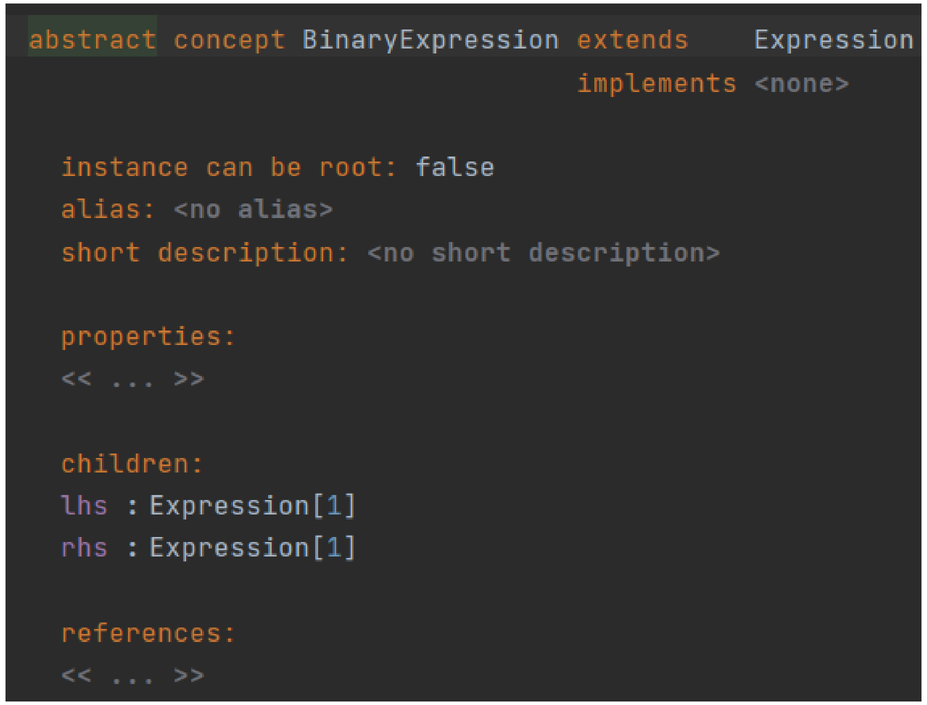
From MPS to LionWeb nodes



From LionWeb to Truffle nodes

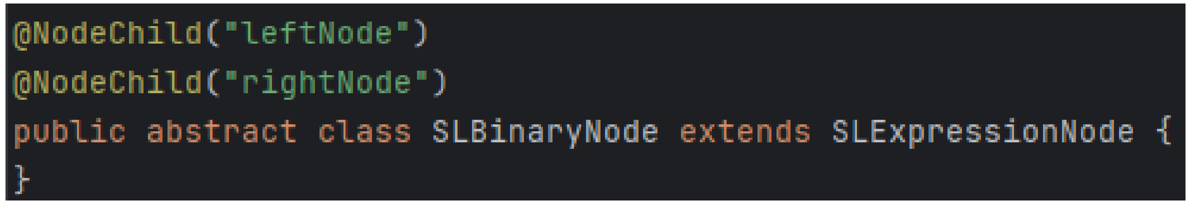


Node definitions in MPS and Truffle BinaryExpression <-> SLBinaryNode



```
abstract concept BinaryExpression extends     Expression
                                  implements <none>

  instance can be root: false
  alias: <no alias>
  short description: <no short description>

  properties:
  << ... >>

  children:
  lhs  : Expression[1]
  rhs  : Expression[1]

  references:
  << ... >>
```

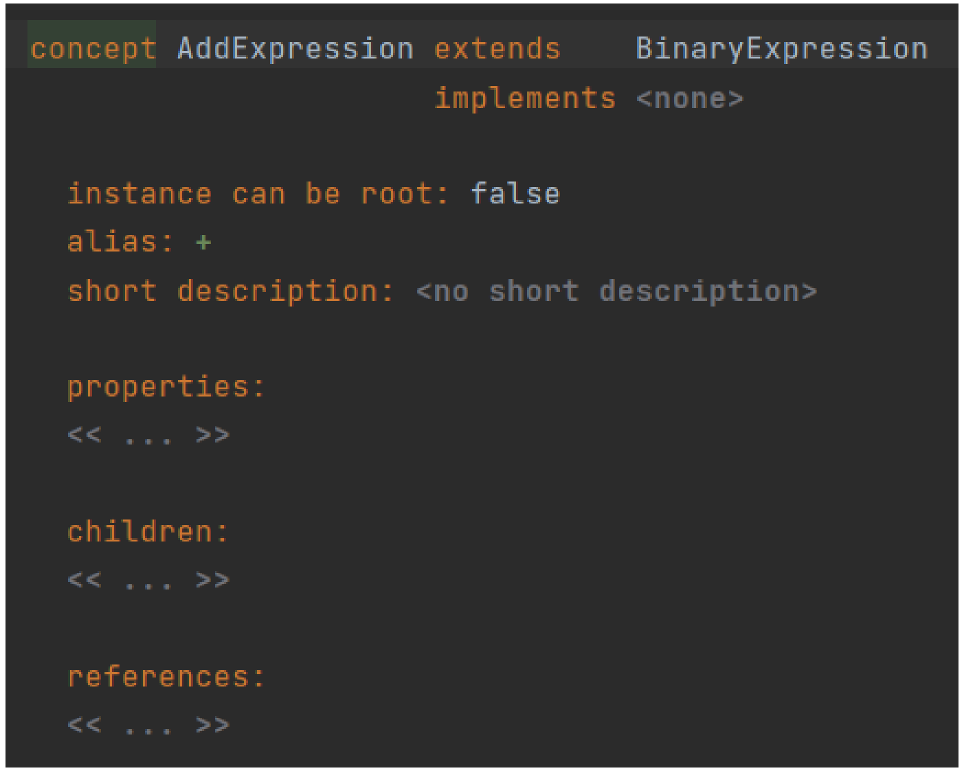
MPS (model AST)



```
@NodeChild("leftNode")
@NodeChild("rightNode")
public abstract class SLBinaryNode extends SLExpressionNode {
}
```

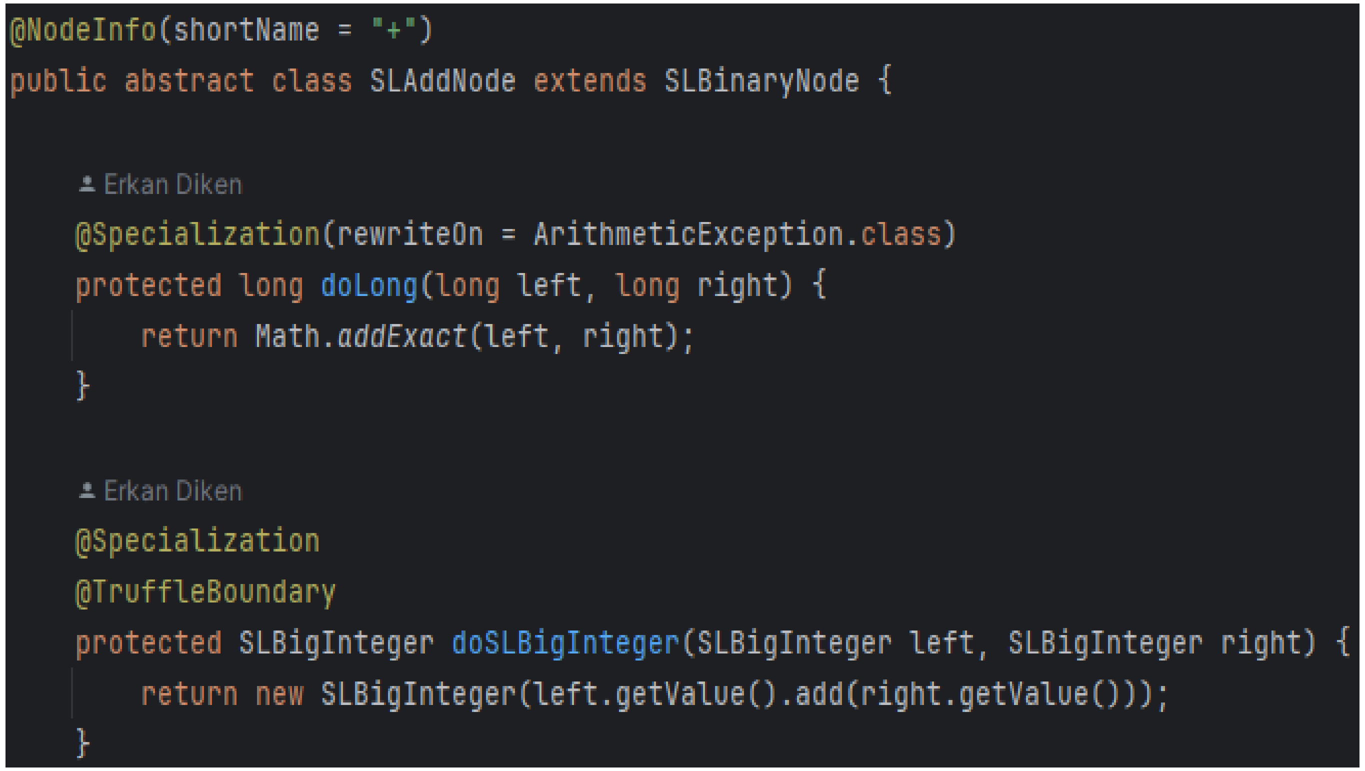
Truffle (interpreter AST)

Node definitions in MPS and Truffle AddExpression <-> SLAddNode



```
concept AddExpression extends    BinaryExpression
                      implements <none>

  instance can be root: false
  alias: +
  short description: <no short description>

  properties:
  << ... >>

  children:
  << ... >>

  references:
  << ... >>
```

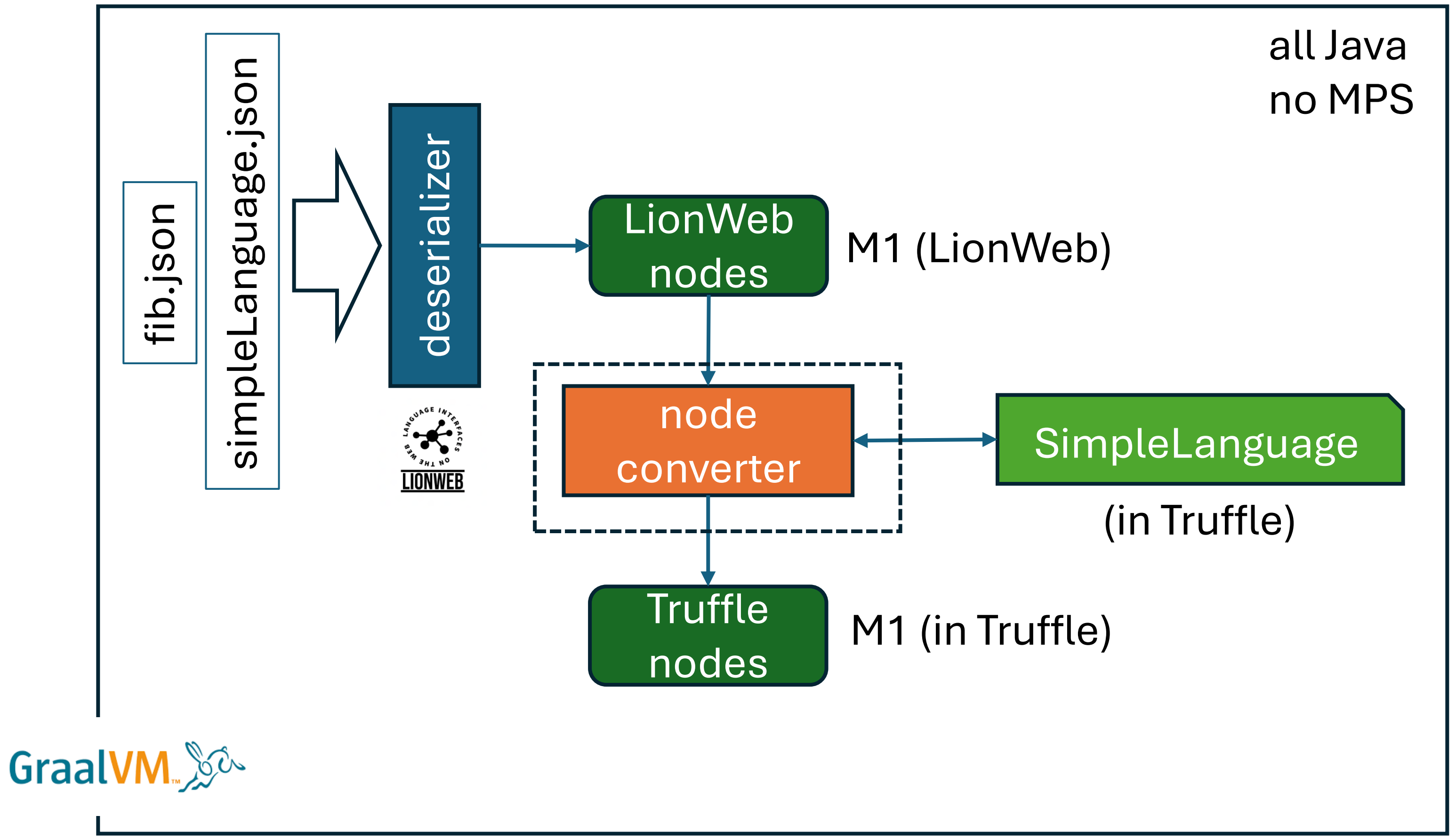
MPS (model AST)



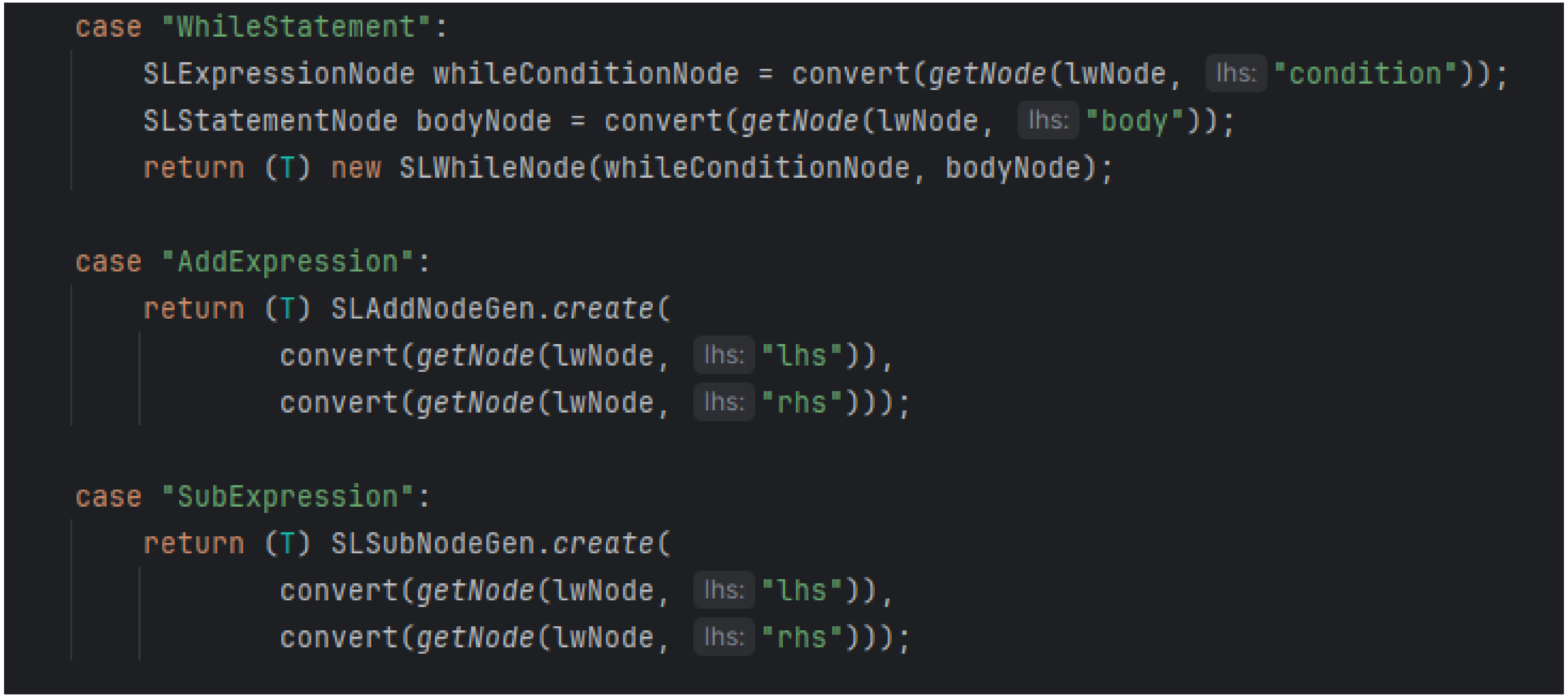
```
@NodeInfo(shortName = "+")
public abstract class SLAddNode extends SLBinaryNode {

    Erkan Diken
    @Specialization(rewriteOn = ArithmeticException.class)
    protected long doLong(long left, long right) {
        return Math.addExact(left, right);
    }

    Erkan Diken
    @Specialization
    @TruffleBoundary
    protected SLBigInteger doSLBigInteger(SLBigInteger left, SLBigInteger right) {
        return new SLBigInteger(left.getValue().add(right.getValue()));
    }
```

Truffle (interpreter AST)

From LionWeb to Truffle nodes



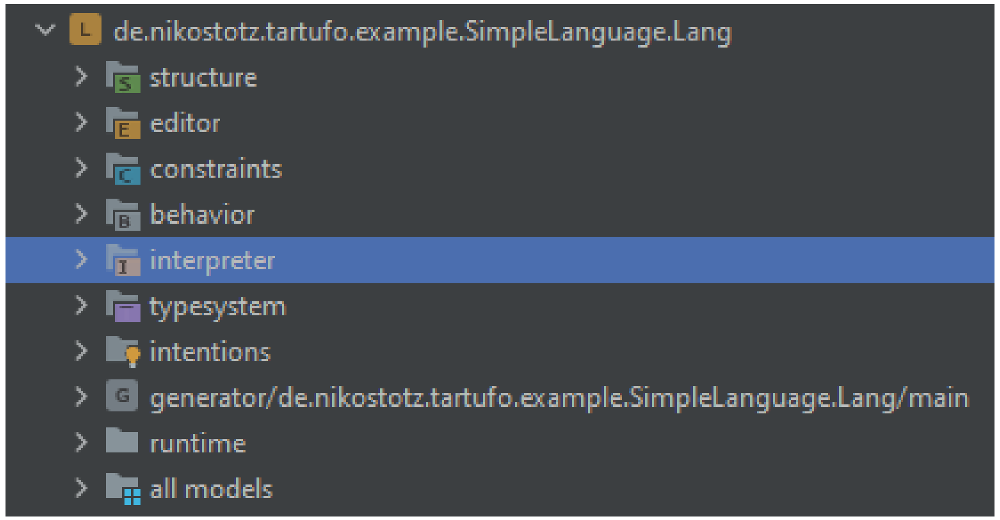
Node conversion



```
case "WhileStatement":
    SLExpressionNode whileConditionNode = convert(getNode(lwNode, lhs: "condition"));
    SLStatementNode bodyNode = convert(getNode(lwNode, lhs: "body"));
    return (T) new SLWhileNode(whileConditionNode, bodyNode);

case "AddExpression":
    return (T) SLAddNodeGen.create(
            convert(getNode(lwNode, lhs: "lhs")),
            convert(getNode(lwNode, lhs: "rhs")));

case "SubExpression":
    return (T) SLSubNodeGen.create(
            convert(getNode(lwNode, lhs: "lhs")),
            convert(getNode(lwNode, lhs: "rhs")));
```

Alternative: From LionWeb/MPS to Truffle nodes

- Integrating interpreter with Truffle in MPS:
  - MPS interpreter language
  - Tartufo: implementation of Truffle in MPS *
- Automates:
  - MPS concepts -> Truffle classes
  - Node conversion

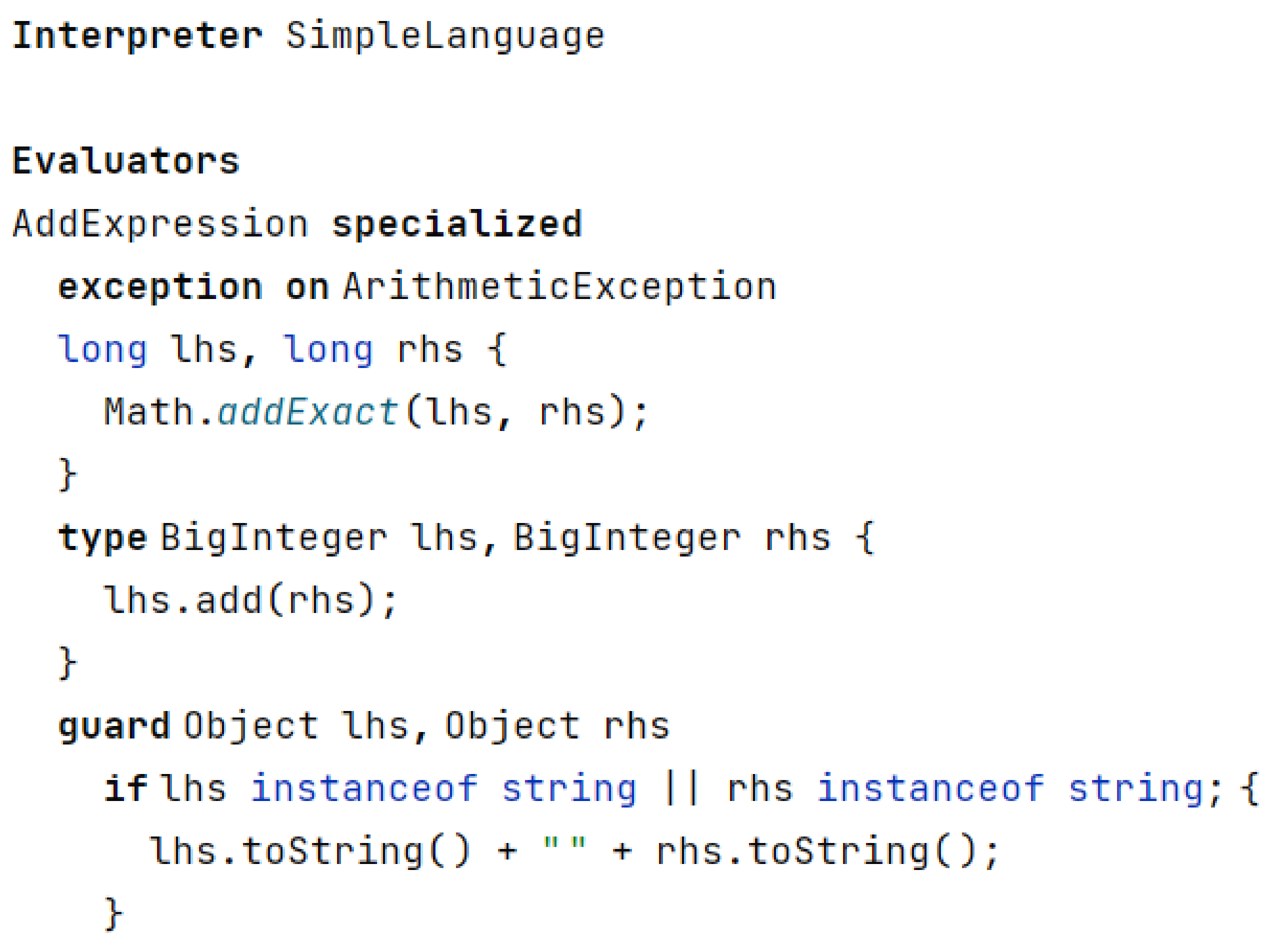


```
de.nikostotz.tartufo.example.SimpleLanguage.Lang
  structure
  editor
  constraints
  behavior
  interpreter
  typesystem
  intentions
  generator/de.nikostotz.tartufo.example.SimpleLanguage.Lang/main
  runtime
  all models
```

Interpreter aspect in MPS language

* "Fast, integrated and debuggable Interpreters in MPS and beyond" (link) by Niko Stotz

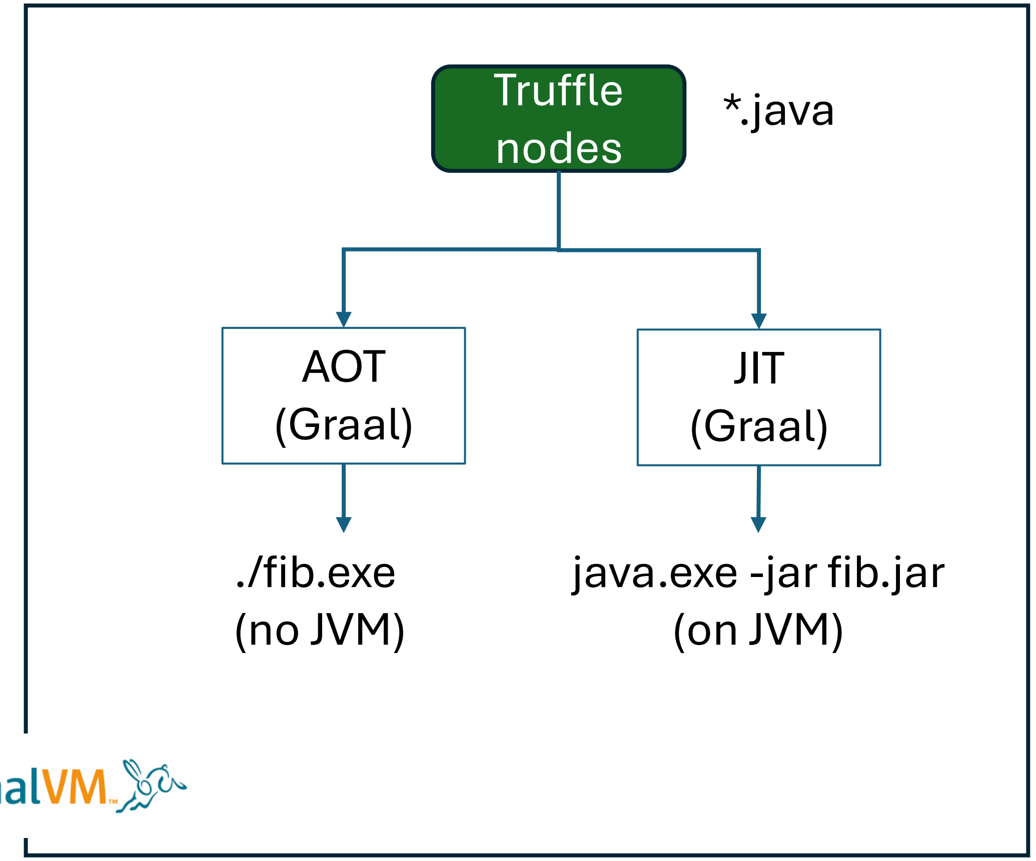
Interpreter Language in MPS



```
Interpreter SimpleLanguage

Evaluators
AddExpression specialized
  exception on ArithmeticException
  long lhs, long rhs {
    Math.addExact(lhs, rhs);
  }
  type BigInteger lhs, BigInteger rhs {
    lhs.add(rhs);
  }
  guard Object lhs, Object rhs
    if lhs instanceof string || rhs instanceof string; {
      lhs.toString() + "" + rhs.toString();
    }
```

From Truffle nodes to executable code

- Building for a JVM
  - Java bytecode
- Building a native image (AOT)
  - Machine code



Demo

https://github.com/erk-dik/langdev24-mps-lionweb-demo
https://github.com/erk-dik/langdev24-lionweb-truffle-demo

Conclusions and future work

- Executed MPS/LionWeb model outside of MPS
- Truffle is chosen as target execution framework
  - Truffle and Graal technology stack
  - High-performance production ready code generation
- Easy to use and integrate LionWeb tools and libraries
- Future work:
  - Seamless integration of LionWeb ⇔ Truffle:
    - Automate node conversions
    - Abstract way to describe semantics

Thank you

erkan@f1re.io

F1RE forge dsls

2VFW

ProxyHands

LangDev CON 2024

Seville 17-19 October, 2024

https://langdevcon.org

Resources

1. https://www.jetbrains.com/mps/
2. https://github.com/LionWeb-io
3. https://www.graalvm.org/
4. https://www.graalvm.org/latest/graalvm-as-a-platform/language-implementation-framework/