POLYGLOT PROGRAM OPTIMIZATION

Anadi Mitra and Prof. Manas Thakur

IIT Bombay

1. What Is Polyglot Programming

- Writing software using multiple programming languages within a single program.
- These program runs within a common execution environment.

Example 1: Java-Python Polyglot Program

```
void java-python(String anotherScript) {
    //API to manage polyglot environment
    Context context = Context.newBuilder().build();

    Point generatorPoint= new Point(32,54);//cartesian point

    String pythonScript =
        "def pointDoubling(point):\n" +
                "\tslope= (3* point.x* point.x- 3)/(2* point.y)\n" +
                "\tx= slope*slope-2*point.x\n" +
                "\ty= slope*(point.x-x)-point.y\n" +
                "\treturn Point.newPoint(x, y)\n"; //object created

    //context is API for managing polyglot runtime
    context.eval(Source.newBuilder("python", pythonScript).build());

    Value pythonBindings = context.getBindings("python");
    //get the python function in java
    Value pointDoubling = pythonBindings.getMember("pointDoubling");
    //pass the point object to python and store the results
    Value result = pointDoubling.execute(generatorPoint);

    showResults(result);
}
```

2. Be A Polyglot Programmer

- Allows you to leverage the strengths of different languages for specific tasks.
- Enhances flexibility and improve performance.
- Use existing libraries, increases development speed.

Use case: Python for simplicity, Java for performance, and JavaScript for web integration.

3. Run Polyglot Programs

Quite a few polyglot runtimes available:

- GraalVM
- Jupyter Notebooks
- .NET CLR
- Node.js with WebAssembly

4. GraalVM Architecture

- A high-performance runtime that allows you to run polyglot programs.
- Uses JIT compiler to optimize JVM programs.

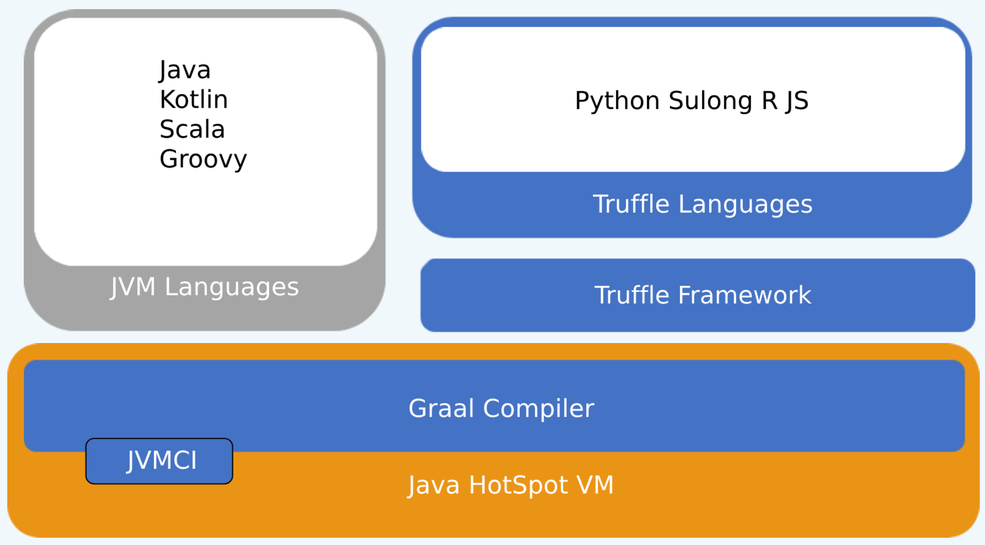

Figure 1. GraalVM

Example 2: Escape Analysis In Java

```
static Point g1;   //some global variable

void foo(MyClass c){

    Point p1= new Point(32,6);   //O1
    Point p2= new Point(16,4);   //O2
    Point p3= new Point(8,3);    //O3

    //passing O1 and O2 to bar
    bar(p1,p2);
    //we don't have implementation of foobar at static time
    c.foobar(p3);    //O3 escapes or not?
}

void bar(Point q1, Point q2){

    //only displaying fields of q1 hence O1 does not escape
    System.out.println("("+q1.x+","+q1.y+")");

    g1=q2;  //p2 is escaping to global space via g1

}
```

5. Problem With Polyglot Programs

- **Polyglot runtimes** are complex, though abstracts language boundaries.
- **Debugging & Readability** of such programs can be challenging.
- **Traditional analysis** tools does not fully support polyglot code analysis.

Challenge

- **Inter-Language Program Analyses**
  Language boundaries in polyglot programs limit the analyses, making optimization challenging.
- **JIT Compilation And Dynamic Languages**
  JIT makes program analysis harder due to runtime dynamism and partially available programs.

6. Key Idea: A Hybrid Approach

Dependencies are generated **statically** for analysis.
Resolve them **dynamically** when whole program is available.

7. Common Interface For Framework

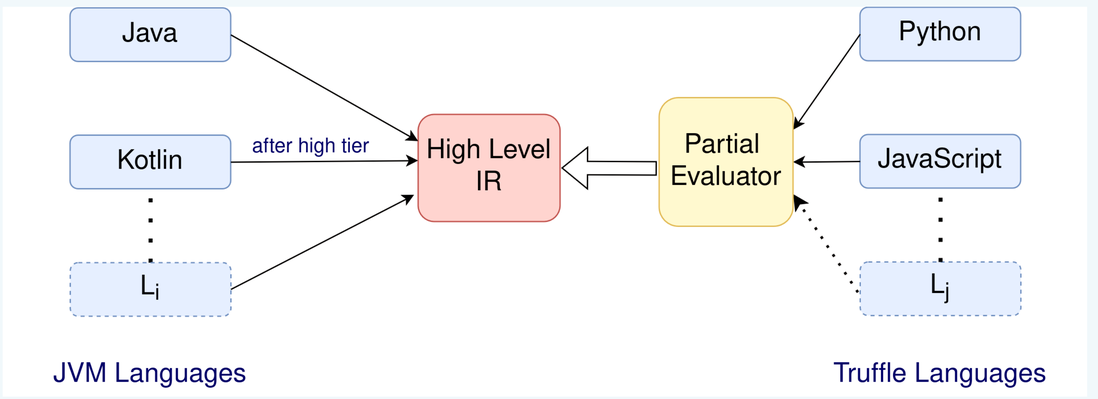

Figure 2. N×M polyglot systems with a common interface

References

[1] Manas Thakur and V. Krishna Nandivada.
PYE: A Framework for Precise-Yet-Efficient Just-In-Time Analyses for Java Programs.
ACM Trans. Program. Lang. Syst., 41(3), July 2019.