Fine-grained Language Composition

Carl Friedrich Bolz
Edd Barrett
Lukas Diekmann
Laurence Tratt

Software Development Team
March 15, 2016
Languages get **better** but also **bigger**.
Language composition
Too coarse and typically high-level to low-level.
Language Composition Challenges

Parsing

Running

<table>
<thead>
<tr>
<th>SUB</th>
<th>AX, AX</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>ES, AX</td>
</tr>
<tr>
<td>SUB</td>
<td>BH, BH</td>
</tr>
<tr>
<td>MOV</td>
<td>BL, INT_NUMBER</td>
</tr>
<tr>
<td>SHL</td>
<td>BX, 1</td>
</tr>
<tr>
<td>SHL</td>
<td>BX, 1</td>
</tr>
<tr>
<td>MOV</td>
<td>DI, ES:[BX]</td>
</tr>
<tr>
<td>MOV</td>
<td>ES, ES:[BX+2]</td>
</tr>
<tr>
<td>ADD</td>
<td>DI, 4</td>
</tr>
<tr>
<td>LEA</td>
<td>SI, TAG</td>
</tr>
<tr>
<td>MOV</td>
<td>CX, TAG_LEN</td>
</tr>
</tbody>
</table>
Parsing compositions

- **LR** undefined
- **Generalised** ambiguous
- **PEG** shadowing
public class Say extends <none> implements <none> {

    private String textchanged;
    <<properties>>
    <<initializer>>
    public Say(String text) {
        <<no statements>>
    }

    <<methods>>

    <<nested classifiers>>
}
The challenge

Challenge:
SDE’s power +
a text editor feel?
Solution: Language Box Editor
Underlying language composition challenges

Parsing

Running

```
SUB     AX,AX
MOV     ES,AX
SUB     BH,BH
MOV     BL,INT_NUMBER
SHL     BX,1
SHL     BX,1
MOV     DI,ES:[BX]
MOV     ES,ES:[BX+2]
ADD     DI,4
LEA     SI,TAG
MOV     CX,TAG_LEN
```

10/24 HTTP://SOFT-DEV.ORG/
Composing Runtimes

RT X

RT Y

RT Z

Easy?
Runtime Composition

PL X

Interpreter

PL Y

Interpreter

C/C++
Runtime Composition

Too slow

C/C++
Runtime Composition

PL X

JIT Compiler

Interpreter

PL Y

JIT Compiler

Interpreter

C/C++
Too much engineering
Runtime Composition

PL X

Interpreter

JVM/CLR

JIT Compiler

PL Y

Interpreter

JIT Compiler

12/24 HTTP://SOFT-DEV.ORG/
Runtime Composition

Poor performance for dynamic languages
Solution: Meta-tracing
Meta-tracing

Interpreter → Meta-tracing

Interpreter

Tracing JIT
Meta-tracing

Python
Interpreter
→
RPython
→
Python
Interpreter
Tracing JIT

pypy
How Does this Apply to VM Composition?

Little Engineering + High Performance

RPython

Interpreters

Glue

Composed VM!

Interpreter

Tracing JIT

PL X

PL Y

PL Z
So far we have:

- **Unipycation**: Python + Prolog
- **PyHyp**: Python + PHP
- **SQPyte**: Python + SQLite
Let’s see one of these VMs in action!

PyHyp = PyPy + HippyVM
How should we measure performance?
E.g. PyHyp composed benchmarks:
- Usually 1-2x slower than mono-language versions.
- In some cases composed benchmarks are faster.

(See our papers for detailed performance analysis)
Qualitative Comments and Conclusion

- Editing composed programs with language boxes.
  - Practical way to parse composed programs.
  - Palatable user experience.

- Implementing composed VMs with meta-tracing.
  - Little engineering effort.
  - Good performance.

- Designing x-language interfaces: Hard!
  - Mapping data structures between languages.
  - Mutability differences.
  - Scoping differences.
  - …
Further Reading


- “Approaches to Interpreter Composition”, Edd Barrett, Carl Friedrich Bolz, Laurence Tratt.


- “Eco: A Language Composition Editor”, Lukas Diekmann, Laurence Tratt.
Thanks for Listening

Discussion / Questions