Fine-grained language composition

Edd Barrett, Carl Friedrich Bolz, Lukas Diekmann, Geoff French, Sarah Mount, Jasper Schulz, Laurence Tratt, Naveneetha Krishnan Vasudevan

Software Development Team
2016-05-25
A perfect programming language
Solution
Solution

A new programming language
Background

Reality
Reality

Another imperfect programming language
What to expect from this talk

A

B
What to expect from this talk

A ∪ B
What to expect from this talk

Python ∪ Prolog
What to expect from this talk

Python ∪ PHP
Two levels of challenge

Tooling
Two levels of challenge

Tooling

Language friction
Tooling challenges

- Python
- PHP
- PyHyp

6 / 26

http://soft-dev.org/
Tooling challenges

Python
syntax
runtime

PHP
syntax
runtime

PyHyp
syntax
runtime
Tooling challenges

Python
- Syntax
- Runtime

PHP
- Syntax
- Runtime

PyHyp
- Syntax
- Runtime

Language boxes

6/26 HTTP://SOFT-DEV.ORG/
Tooling challenges

- Language boxes
  - Syntax
    - Python
      - Runtime
    - PHP
      - Runtime
  - PyHyp
    - Syntax
    - Runtime
  - Composed meta-meta-tracing VMs
Syntax composition

**PL X**

```
<grammar>
expr ::= ...
term ::= ...
    | ...
    | ...
func ::= ...
```

**PL Y**

```
<program>
for (j : js) {
    doStuff();
}
...```

http://soft-dev.org/
Syntax composition

```plaintext
<grammar>
expr::= ...
term::= ... |
    | ...
func ::= ...

<program>
for (j : js) {
    doStuff();
}
.
.
.
```

Parser

PL X
<grammar>
expr::= ...
term::= ... |
    | ...
func ::= ...

PL Y
<program>
for (j : js) {
    doStuff();
}
.
.
.

7 / 26 HTTP://SOFT-DEV.ORG/
Syntax composition

PL X
<grammar>
expr::= ...
term::= ...
| ...
| ...
func ::= ...

PL Y
<program>
for (j : js) {
    doStuff();
}
.
.
.
Parser
Parse Tree
The only choice?

SDE
Challenge:
SDE’s power +
a text editor feel?
Runtime composition
Runtime composition

PL X

Interpreters

Glue

PL Y

Meta-tracing

PL Z

Interpreter

Tracing JIT
Meta-tracing translation with RPython

Interpreter
Meta-tracing translation with RPython
Meta-tracing translation with RPython
Meta-tracing translation with RPython

You write this

you write this

RPython translator

Optimised Interpreter

JIT

Interpreter

Optimised

Interpreter

JIT
Meta-tracing translation with RPython

you write this

you get this for free

Optimised Interpreter

JIT

you get this for free
Runtime composition

![Diagram showing the components of the runtime composition process.]

- **PL X** and **PL Y** represent interpreters.
- **PL Z** contains an interpreter and a tracing JIT.
- **Meta-tracing** is the central component connecting the interpreters to the other components.
- **Glue** facilitates communication between the components.
Runtime composition

PyPy
Hippy
Interpreters
Glue
Meta-tracing
PyHyp
Interpreter
Tracing JIT
Performance demo
## Composed Richards vs. other VMs

<table>
<thead>
<tr>
<th>Type</th>
<th>VM</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mono</td>
<td>CPython 2.7.7</td>
<td>9.475 ± 0.0127</td>
</tr>
<tr>
<td></td>
<td>HHVM 3.4.0</td>
<td>4.264 ± 0.0386</td>
</tr>
<tr>
<td></td>
<td>HippyVM</td>
<td>0.250 ± 0.0008</td>
</tr>
<tr>
<td></td>
<td>PyPy 2.4.0</td>
<td>0.178 ± 0.0006</td>
</tr>
<tr>
<td></td>
<td>Zend 5.5.13</td>
<td>9.070 ± 0.0361</td>
</tr>
</tbody>
</table>
## Composed Richards vs. other VMs

<table>
<thead>
<tr>
<th>Type</th>
<th>VM</th>
<th>Time (µs) ± Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mono</td>
<td>CPython 2.7.7</td>
<td>9.475 ± 0.0127</td>
</tr>
<tr>
<td></td>
<td>HHVM 3.4.0</td>
<td>4.264 ± 0.0386</td>
</tr>
<tr>
<td></td>
<td>HippyVM</td>
<td>0.250 ± 0.0008</td>
</tr>
<tr>
<td></td>
<td>PyPy 2.4.0</td>
<td>0.178 ± 0.0006</td>
</tr>
<tr>
<td></td>
<td>Zend 5.5.13</td>
<td>9.070 ± 0.0361</td>
</tr>
<tr>
<td>Composed</td>
<td>PyHyp</td>
<td>0.335 ± 0.0012</td>
</tr>
</tbody>
</table>
Datatype conversion

PHPObject

PHPInt

PHPFunc
Datatype conversion

```
PHPRoot
   /\           /
  PHPObject PHPInt PHPFunc

PyRoot
       /
  PyObject PyInt PyFunc
```
Datatype conversion: primitive types

PHP

Python
Datatype conversion: primitive types

<table>
<thead>
<tr>
<th>PHP</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 : PHPInt</td>
<td></td>
</tr>
</tbody>
</table>
Datatype conversion: primitive types

PHP

2 : PHPInt

Python

2 : PyInt
Datatype conversion: user types

PHP

Python
Datatype conversion: user types

PHP

o : PHPObject

Python
Datatype conversion: user types

- PyRoot
- PyObject
- PyInt
- PyFunc
Datatype conversion: user types

PyRoot
  └── PyObject
  └── PyInt
  └── PyFunc

  └── PyPHPAdapter
Datatype conversion: user types

Diagram:

- PyRoot
  - PyObject
  - PyInt
  - PyFunc
    - PyPHPAdapter
      - php_obj : PHPObjec
Datatype conversion: user types

PHP

```
o : PHPObject
```

Python
Datatype conversion: user types

PHP

`o : PHPObject`

Python

`:PyPHPAdapter`
Datatype conversion: user types

PHP

```
o : PHPObject
```

Python

```
:PyPHPAdapter
```

```
php_obj
```
Datatype conversion: user types

PHP

Python

```
o : PHPObject

:PyPHPAdapter
```

php_obj

**Immutable field**
A good composition needs to reduce *friction*.
A good composition needs to reduce *friction*. Some examples:

- Lexical scoping (or lack thereof) in PHP and Python (semantic friction)
Friction

A good composition needs to reduce friction. Some examples:

- Lexical scoping (or lack thereof) in PHP and Python (semantic friction)
- PHP datatypes are immutable except for references and objects; Python’s are largely mutable (semantic and performance friction)
A good composition needs to reduce friction. Some examples:

- Lexical scoping (or lack thereof) in PHP and Python (semantic friction)
- PHP datatypes are immutable except for references and objects; Python’s are largely mutable (semantic and performance friction)
- PHP has only dictionaries; Python has lists and dictionaries (semantic friction)
Unipycation

PyPy

Hippy

Interpreters

Glue

Meta-tracing

PyHyp

Interpreter

Tracing JIT
Unipycation

PyPy
Interpreters
Glue
Pyrolog

Meta-tracing

Interpreter
Tracing JIT

Software Development Team
http://soft-dev.org/
## Absolute timing comparison

<table>
<thead>
<tr>
<th>VM</th>
<th>Benchmark</th>
<th>Python</th>
<th>Prolog</th>
<th>Python → Prolog</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPython-SWI</td>
<td>SmallFunc</td>
<td>0.125s</td>
<td>±0.007</td>
<td>0.257s ±0.002</td>
</tr>
<tr>
<td></td>
<td>L1A0R</td>
<td>2.924s</td>
<td>±0.284</td>
<td>7.352s ±0.048</td>
</tr>
<tr>
<td></td>
<td>L1A1R</td>
<td>4.184s</td>
<td>±0.038</td>
<td>18.890s ±0.111</td>
</tr>
<tr>
<td></td>
<td>NdL1A1R</td>
<td>7.531s</td>
<td>±0.080</td>
<td>18.643s ±0.197</td>
</tr>
<tr>
<td></td>
<td>TCons</td>
<td>264.415s</td>
<td>±2.250</td>
<td>48.819s ±0.252</td>
</tr>
<tr>
<td></td>
<td>Lists</td>
<td>9.374s</td>
<td>±0.059</td>
<td>25.148s ±0.221</td>
</tr>
<tr>
<td>Unipyication</td>
<td>SmallFunc</td>
<td>0.001s</td>
<td>±0.000</td>
<td>0.006s ±0.001</td>
</tr>
<tr>
<td></td>
<td>L1A0R</td>
<td>0.085s</td>
<td>±0.000</td>
<td>0.086s ±0.000</td>
</tr>
<tr>
<td></td>
<td>L1A1R</td>
<td>0.112s</td>
<td>±0.000</td>
<td>0.114s ±0.000</td>
</tr>
<tr>
<td></td>
<td>NdL1A1R</td>
<td>0.500s</td>
<td>±0.003</td>
<td>0.548s ±0.085</td>
</tr>
<tr>
<td></td>
<td>TCons</td>
<td>6.053s</td>
<td>±0.288</td>
<td>2.444s ±0.003</td>
</tr>
<tr>
<td></td>
<td>Lists</td>
<td>0.845s</td>
<td>±0.002</td>
<td>1.416s ±0.003</td>
</tr>
<tr>
<td>Jython-tuProlog</td>
<td>SmallFunc</td>
<td>0.088s</td>
<td>±0.003</td>
<td>3.050s ±0.053</td>
</tr>
<tr>
<td></td>
<td>L1A0R</td>
<td>1.078s</td>
<td>±0.009</td>
<td>206.590s ±3.846</td>
</tr>
<tr>
<td></td>
<td>L1A1R</td>
<td>2.145s</td>
<td>±0.232</td>
<td>293.311s ±5.691</td>
</tr>
<tr>
<td></td>
<td>NdL1A1R</td>
<td>7.939s</td>
<td>±0.457</td>
<td>1857.687s ±5.169</td>
</tr>
<tr>
<td></td>
<td>TCons</td>
<td>543.347s</td>
<td>±3.289</td>
<td>8014.477s ±17.710</td>
</tr>
<tr>
<td></td>
<td>Lists</td>
<td>5.661s</td>
<td>±0.046</td>
<td>6981.873s ±18.795</td>
</tr>
</tbody>
</table>
## Relative timing comparison

<table>
<thead>
<tr>
<th>VM</th>
<th>Benchmark</th>
<th>Python→Prolog</th>
<th>Prolog→Python</th>
<th>Python→Prolog Unipycation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPython-SWI</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmallFunc</td>
<td>231.770×</td>
<td>±13.136</td>
<td>112.567×</td>
<td>±1.242</td>
</tr>
<tr>
<td>L1A0R</td>
<td>3.184×</td>
<td>±0.300</td>
<td>1.266×</td>
<td>±0.014</td>
</tr>
<tr>
<td>L1A1R</td>
<td>4.987×</td>
<td>±0.049</td>
<td>1.105×</td>
<td>±0.007</td>
</tr>
<tr>
<td>NdL1A1R</td>
<td>88.654×</td>
<td>±1.368</td>
<td>35.814×</td>
<td>±0.554</td>
</tr>
<tr>
<td>TCons</td>
<td>8.264×</td>
<td>±0.101</td>
<td>44.760×</td>
<td>±0.453</td>
</tr>
<tr>
<td>Lists</td>
<td>235.459×</td>
<td>±2.314</td>
<td>87.772×</td>
<td>±1.017</td>
</tr>
<tr>
<td><strong>Unipyction</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmallFunc</td>
<td>1.295×</td>
<td>±0.105</td>
<td>0.182×</td>
<td>±0.054</td>
</tr>
<tr>
<td>L1A0R</td>
<td>1.020×</td>
<td>±0.002</td>
<td>1.012×</td>
<td>±0.002</td>
</tr>
<tr>
<td>L1A1R</td>
<td>1.025×</td>
<td>±0.002</td>
<td>1.002×</td>
<td>±0.003</td>
</tr>
<tr>
<td>NdL1A1R</td>
<td>5.349×</td>
<td>±0.045</td>
<td>4.879×</td>
<td>±0.924</td>
</tr>
<tr>
<td>TCons</td>
<td>5.959×</td>
<td>±0.282</td>
<td>14.756×</td>
<td>±0.092</td>
</tr>
<tr>
<td>Lists</td>
<td>5.982×</td>
<td>±0.045</td>
<td>3.569×</td>
<td>±0.026</td>
</tr>
<tr>
<td><strong>Jython-tuProlog</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SmallFunc</td>
<td>592.904×</td>
<td>±19.517</td>
<td>17.143×</td>
<td>±0.338</td>
</tr>
<tr>
<td>L1A0R</td>
<td>185.460×</td>
<td>±2.818</td>
<td>0.968×</td>
<td>±0.021</td>
</tr>
<tr>
<td>L1A1R</td>
<td>137.427×</td>
<td>±14.537</td>
<td>1.005×</td>
<td>±0.028</td>
</tr>
<tr>
<td>NdL1A1R</td>
<td>250.776×</td>
<td>±14.666</td>
<td>1.072×</td>
<td>±0.009</td>
</tr>
<tr>
<td>TCons</td>
<td>15.096×</td>
<td>±0.106</td>
<td>1.023×</td>
<td>±0.004</td>
</tr>
<tr>
<td>Lists</td>
<td>985.149×</td>
<td>±8.674</td>
<td>0.799×</td>
<td>±0.003</td>
</tr>
</tbody>
</table>
What can we use this for?
What can we use this for?

First-class languages
What can we use this for?

First-class languages

Language migration
Thanks to our funders

- EPSRC: COOLER and Lecture.
- Oracle: various.
Thanks for listening

http://soft-dev.org/