Fine-grained Language Composition: A Case Study

Edd Barrett

Carl Friedrich Bolz

Lukas Diekmann

Laurence Tratt

Software Development Team
2016-07-22
“The ability to write a computer program in a mix of programming languages.”
Why Compose Languages?

- Choose the best language for the job.
- Access to a broader set of libraries.
- Language migration.
Beyond the FFI

Lang X

call

Lang Y

return
Beyond the FFI

my_composed_program
Breaking it Down

PL X

PL Y

PL Z
Breaking it Down

Diagram showing the relationship between syntax and runtime for PL X, PL Y, and PL Z.
Runtime composition

Engineering Effort

Interpreters

Performance
Runtime composition

Engineering Effort

Interpreters

Performance

JITs
Our Approach

Language Boxes + Meta-tracing
Language Boxes to Compose Syntax

- The best bits from Syntax Directed Editing (SDE)
- Palatable editing experience
Meta-tracing to Compose Runtimes

PL X
Interpreters
Glue

PL Y

Metatracing

PL Z
Interpreter
Tracing JIT
PyHyp

PHP + Python
Features of PyHyp

- Calling Python functions and methods from PHP
- Calling PHP functions and methods from Python
- Transparent type conversions
- Arbitrary nesting of foreign functions
- Python expressions in PHP
- “Embedding” Python methods inside PHP classes
- Adds support for references to Python
- Cross-language scoping
- Cross-language exceptions
## PyHyp Performance

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>HippyVM</th>
<th>PyHypPHP</th>
<th>PyHypPy</th>
<th>PyPy</th>
</tr>
</thead>
<tbody>
<tr>
<td>instchain</td>
<td>0.912</td>
<td>1.000</td>
<td>1.360</td>
<td>0.675</td>
</tr>
<tr>
<td></td>
<td>±0.0011</td>
<td>±0.0003</td>
<td>±0.0007</td>
<td>±0.0007</td>
</tr>
<tr>
<td>l1a0r</td>
<td>1.368</td>
<td>1.000</td>
<td>1.303</td>
<td>1.340</td>
</tr>
<tr>
<td></td>
<td>±0.0004</td>
<td>±0.0016</td>
<td>±0.0106</td>
<td>±0.0022</td>
</tr>
<tr>
<td>l1a1r</td>
<td>1.306</td>
<td>1.000</td>
<td>1.140</td>
<td></td>
</tr>
<tr>
<td></td>
<td>±0.0017</td>
<td>±0.0016</td>
<td>±0.0022</td>
<td>±0.0022</td>
</tr>
<tr>
<td>total_list</td>
<td>0.864</td>
<td>1.000</td>
<td>1.508</td>
<td>0.587</td>
</tr>
<tr>
<td></td>
<td>±0.0002</td>
<td>±0.0004</td>
<td>±0.0003</td>
<td>±0.0003</td>
</tr>
<tr>
<td>walk_list</td>
<td>0.779</td>
<td>1.000</td>
<td>1.601</td>
<td>1.080</td>
</tr>
<tr>
<td></td>
<td>±0.0011</td>
<td>±0.0026</td>
<td>±0.0015</td>
<td>±0.0026</td>
</tr>
<tr>
<td>deltableblue</td>
<td>4.325</td>
<td>1.000</td>
<td>0.457</td>
<td></td>
</tr>
<tr>
<td></td>
<td>±0.0212</td>
<td></td>
<td>±0.0026</td>
<td></td>
</tr>
<tr>
<td>fannkuch</td>
<td>1.848</td>
<td>1.000</td>
<td>1.005</td>
<td></td>
</tr>
<tr>
<td></td>
<td>±0.0007</td>
<td>±0.0004</td>
<td>±0.0004</td>
<td></td>
</tr>
<tr>
<td>mandel</td>
<td>0.921</td>
<td>1.000</td>
<td>0.999</td>
<td></td>
</tr>
<tr>
<td></td>
<td>±0.0005</td>
<td>±0.0003</td>
<td>±0.0003</td>
<td></td>
</tr>
<tr>
<td>Richards</td>
<td>0.853</td>
<td>1.000</td>
<td>0.488</td>
<td></td>
</tr>
<tr>
<td></td>
<td>±0.0010</td>
<td></td>
<td>±0.0005</td>
<td></td>
</tr>
<tr>
<td>Geometric Mean</td>
<td>1.222</td>
<td>1.000</td>
<td>0.963</td>
<td>0.813</td>
</tr>
<tr>
<td></td>
<td>±0.0006</td>
<td>±0.0003</td>
<td>±0.0007</td>
<td>±0.0007</td>
</tr>
</tbody>
</table>

**Worst case: 2.6x overhead**
Implementing desired behaviour: relatively easy

Deciding the correct behaviours: hard

“Semantic friction”
### Semantic Friction: Collection Types

<table>
<thead>
<tr>
<th>Sequence type</th>
<th>PHP</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>array</td>
<td>list</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mapping type</th>
<th>PHP</th>
<th>Python</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>array</td>
<td>dict</td>
</tr>
</tbody>
</table>
Semantic Friction: Collection Types

```
["a", "b", "c"]
```

Semantic Friction: Collection Types

PHP <-> Language Threshold <-> Python

array <-> ? <-> list
array <-> ? <-> dict

HTTP://SOFT-DEV.ORG/
Semantic Friction: Collection Types

PHP ↔ Language Threshold ↔ Python

- **array** (PHP) ↔ **list** (Python)
  - Int keys

- **array** (PHP) ↔ **dict** (Python)
  - Mixed keys
Semantic Friction: Collection Types

PHP  ↔  Language Threshold  ↔  Python

$a = \text{array}

int keys

$a["x"] = 4

array

mixed keys

List

array

18/23 HTTP://SOFT-DEV.ORG/
Semantic Friction: Collection Types

Language Threshold

PHP

$a = \text{array}

\text{int keys}

$a["x"] = 4

\text{array}

mixed keys

Python

list

\text{array}

Inconsistent list!

18/23 HTTP://SOFT-DEV.ORG/
Semantic Friction: Collection Types

PHP

Language Threshold

Python

array

list
dict

as_list()
Conclusions

Language boxes and Meta-tracing:

▶ Fine-grained language composition
▶ Good editing experience
▶ Good Performance
▶ Relatively small engineering effort.

Qualitative outcomes:

▶ Implementing x-lang behaviours is easy.
▶ Designing x-lang behaviours is hard.
Future Work

- Tools for composed programs
  - Debugging
  - Profiling
  - Version control
  - ...

- Statically typed/functional languages.

- Compositions with >2 languages involved.
References

- *Parsing Composed Grammars with Language Boxes* Lukas Diekmann, Laurence Tratt.
- *Eco: A Language Composition Editor* Lukas Diekmann, Laurence Tratt.
- *Unipycation: A Case Study in Cross-language Tracing*, Edd Barrett, Carl Friedrich Bolz and Laurence Tratt
- *Approaches to Interpreter Composition*, Edd Barrett, Carl Friedrich Bolz and Laurence Tratt
- *Fine-grained Language Composition: A Case Study*, Edd Barrett, Carl Friedrich Bolz, Lukas Diekmann, Laurence Tratt
- *Making an Embedded DBMS JIT-friendly*, Carl Friedrich Bolz, Darya Kurilova, Laurence Tratt
Language Boxes + Meta-tracing