

# Fine-grained language composition

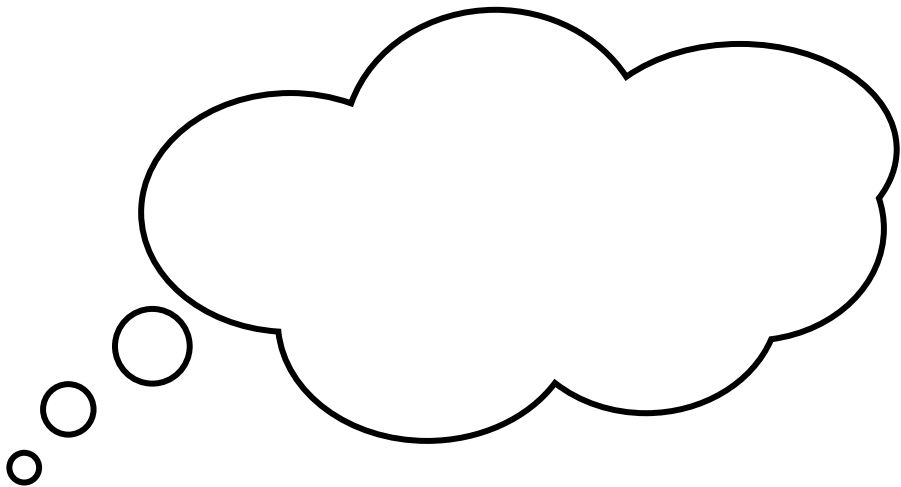


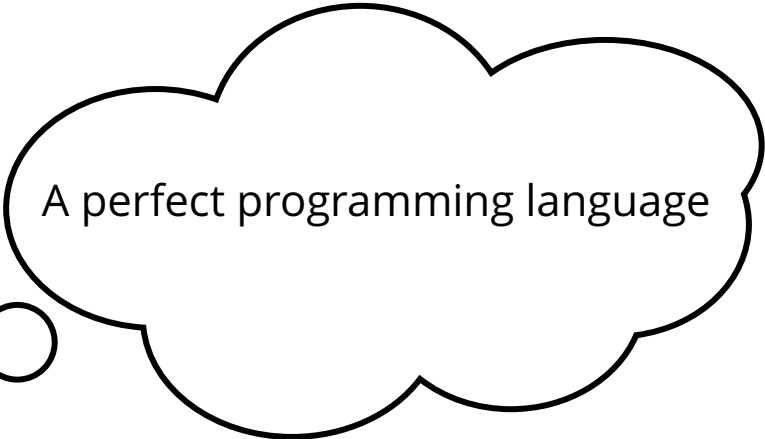
Edd Barrett, Carl Friedrich Bolz, Lukas Diekmann, Geoff French,  
Sarah Mount, Laurence Tratt, Naveneetha Krishnan Vasudevan



Software Development Team  
2016-01-28

# Background





A perfect programming language

*Solution*

## *Solution*

A new programming language

*Reality*

## *Reality*

Another imperfect programming language

# What to expect from this talk

A



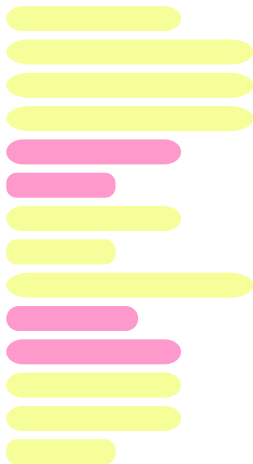
B





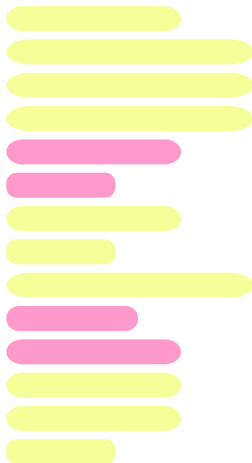
# What to expect from this talk

$A \cup B$



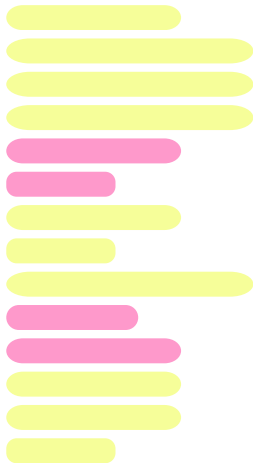
# What to expect from this talk

## Python $\cup$ Prolog



# What to expect from this talk

## Python $\cup$ PHP

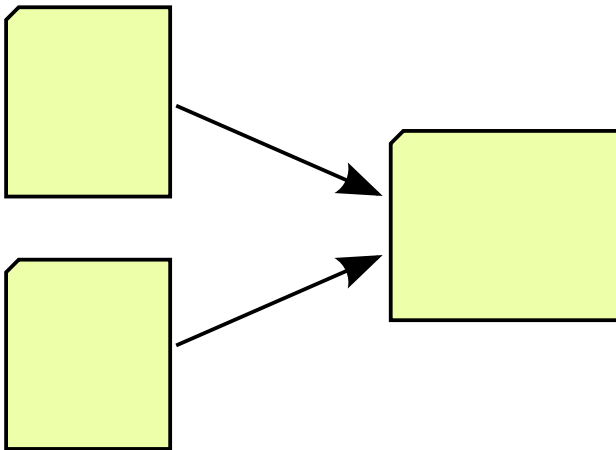


# *Tooling*

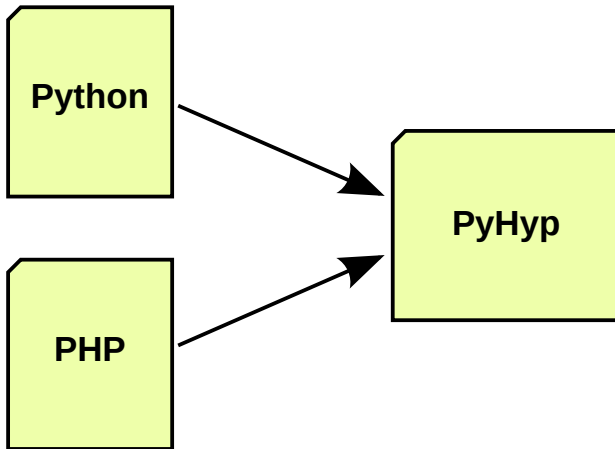
*Tooling*

*Language friction*

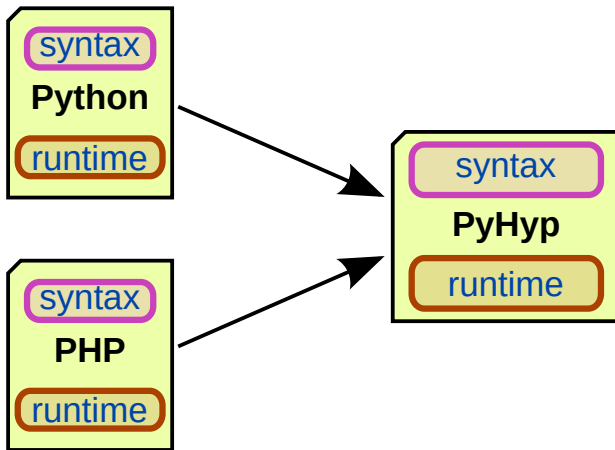
# Tooling challenges



# Tooling challenges

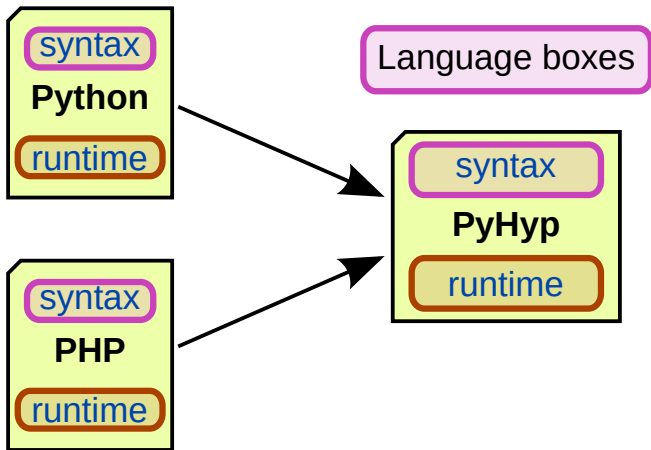


# Tooling challenges

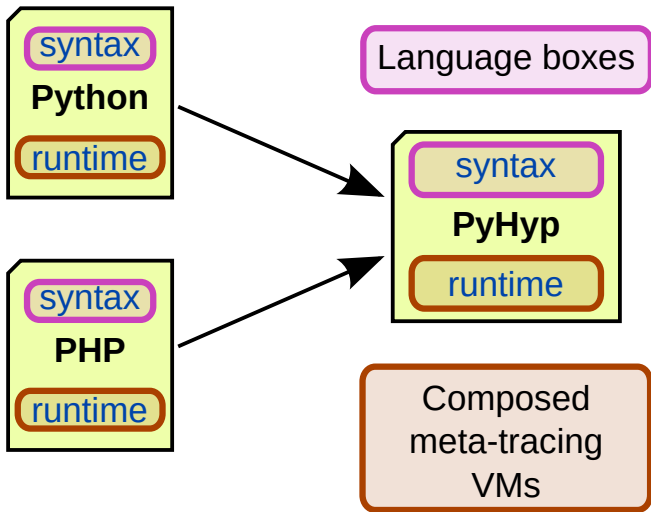




# Tooling challenges



# Tooling challenges



# Parsing: recognising

Input: Bill hits Ben

## Parsing: recognising

Input: Bill hits Ben  
Grammar: noun verb noun

# Parsing: Context-Free Grammars

```
Expr    ::= Expr '+' Term    | Term
Term     ::= Term '*' Factor  | Factor
Factor  ::= '(' Expr ')'     | 'INT'
```

# Parsing: Context-Free Grammars

```
Expr    ::= Expr '+' Term    | Term
Term     ::= Term '*' Factor  | Factor
Factor  ::= '(' Expr ')'     | 'INT'
```

# Parsing: Context-Free Grammars

```
Expr    ::= Expr '+' Term    | Term
Term     ::= Term '*' Factor  | Factor
Factor  ::= '(' Expr ')'     | 'INT'
```



# Parsing: Context-Free Grammars

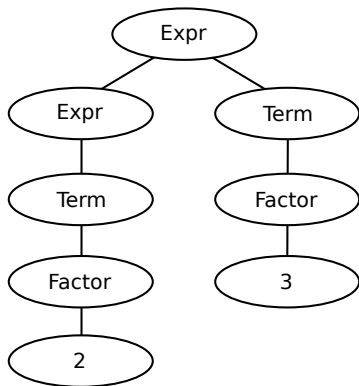
```
Expr    ::= Expr '+' Term    | Term
Term     ::= Term '*' Factor  | Factor
Factor  ::= '(' Expr ')'     | 'INT'
```

# Parsing: Context-Free Grammars

```
Expr    ::= Expr '+' Term    | Term
Term     ::= Term '*' Factor  | Factor
Factor  ::= '(' Expr ')'      | 'INT'
```

# Parsing: Trees

2 + 3



# Syntax composition

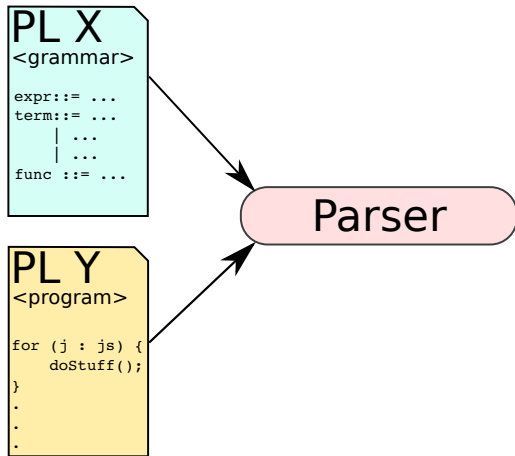
**PL X**  
<grammar>

```
expr ::= ...  
term ::= ...  
      | ...  
      | ...  
func ::= ...
```

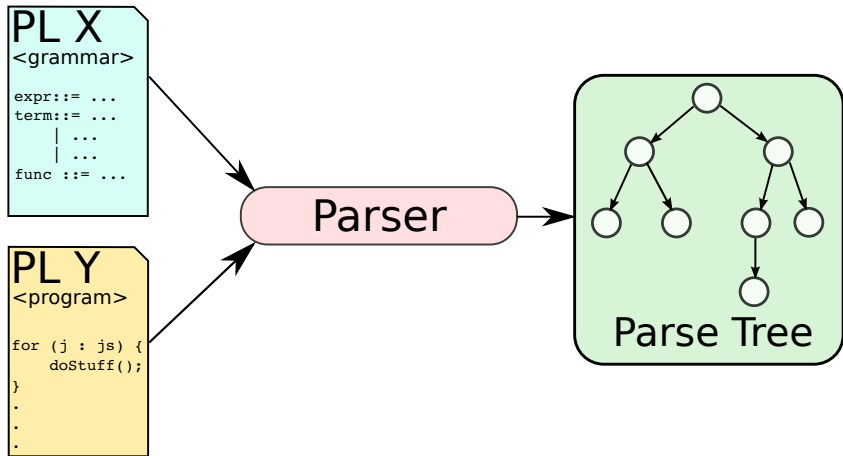
**PL Y**  
<program>

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

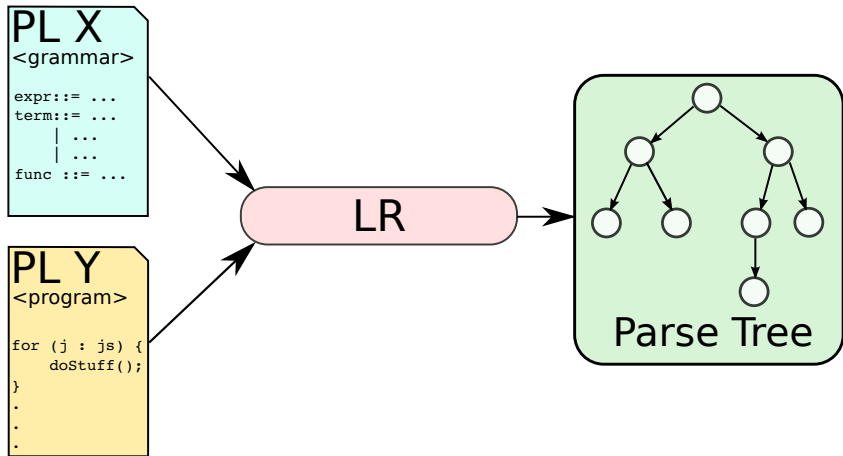
# Syntax composition



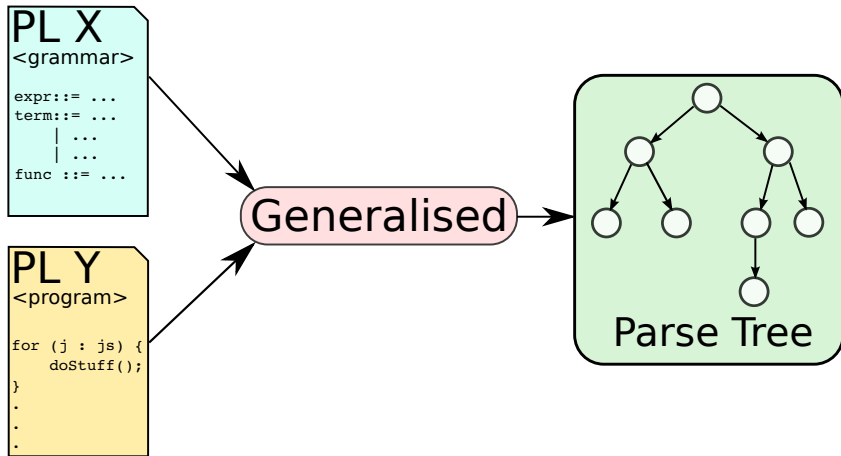
# Syntax composition



# Syntax composition

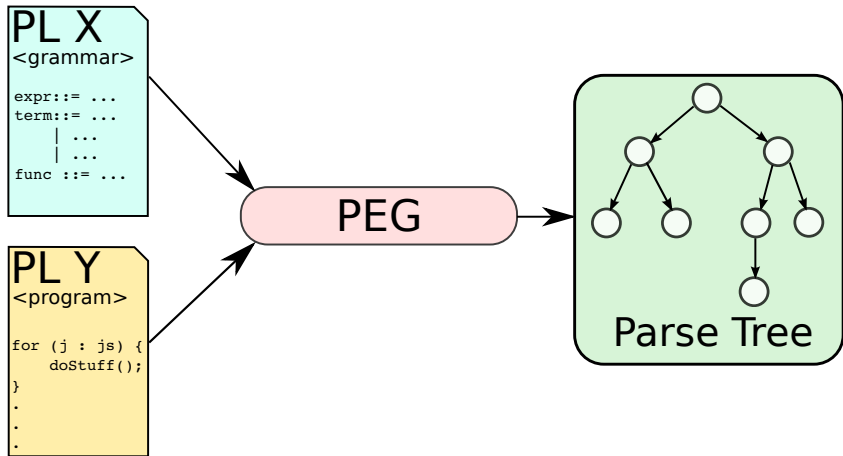


# Syntax composition





# Syntax composition



# The only choice?

The only choice?

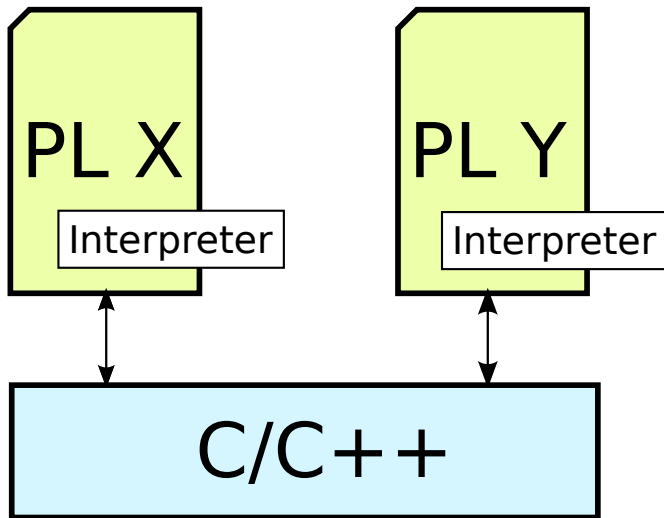
SDE

Challenge:  
SDE's power +  
a text editor feel?

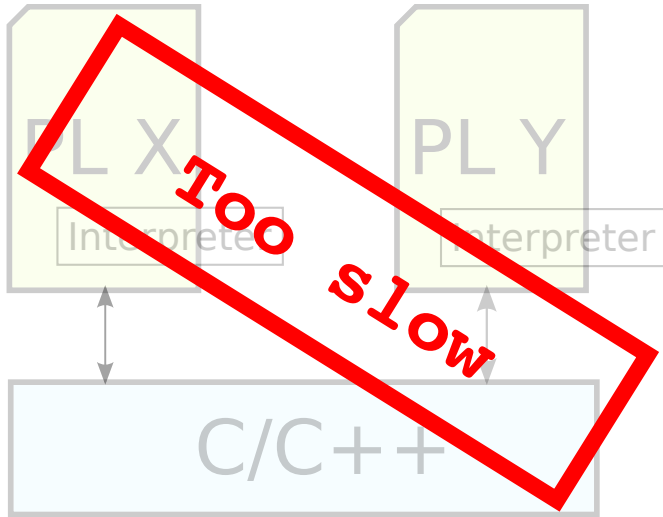
# Eco demo

# Runtime composition

## Runtime composition

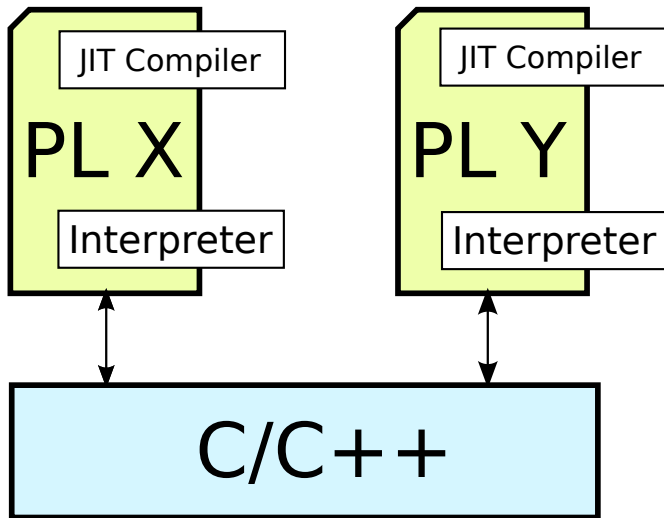


# Runtime composition

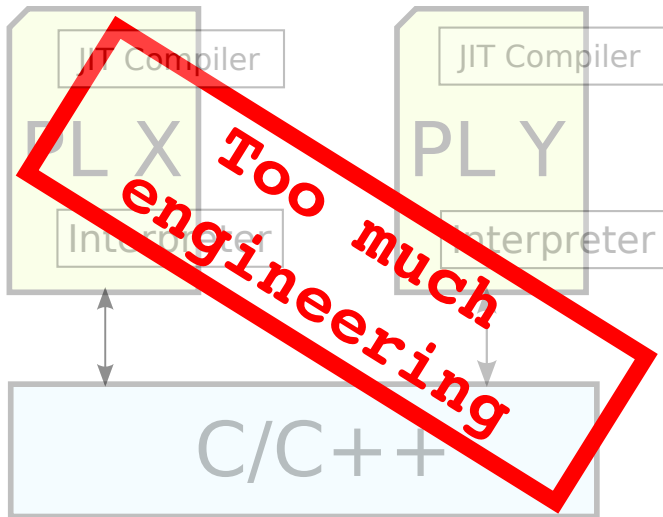




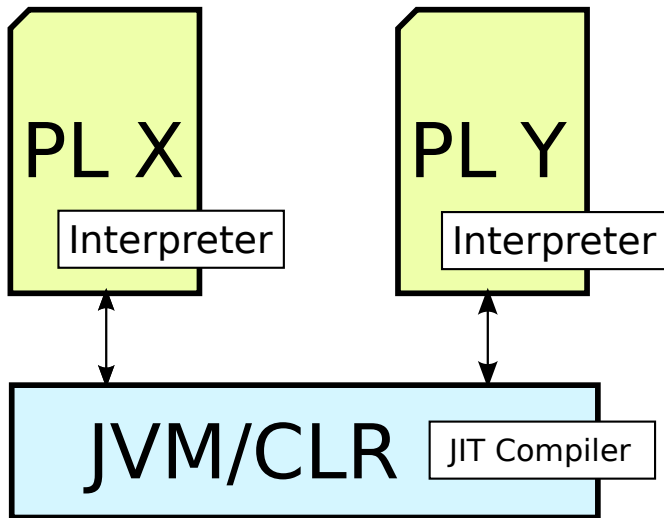
# Runtime composition



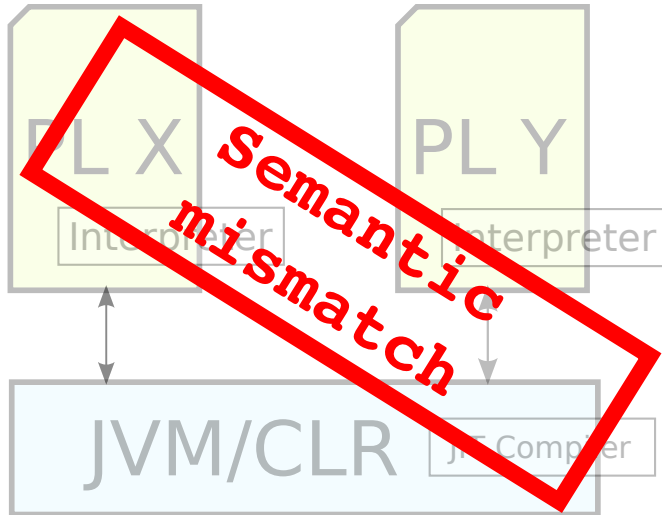
# Runtime composition



## Runtime composition

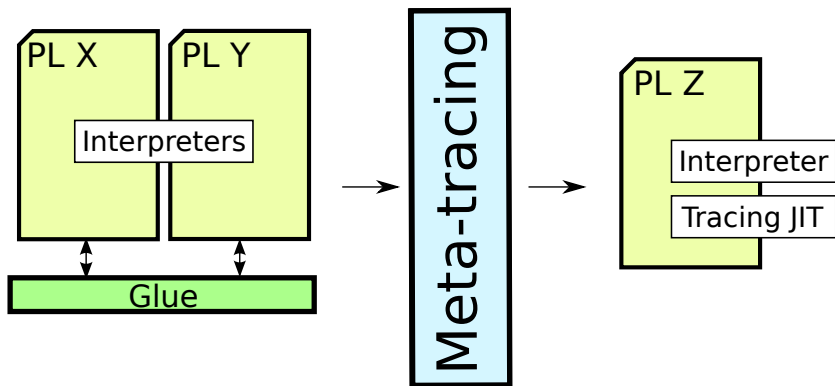


# Runtime composition



# Runtime composition

# Runtime composition

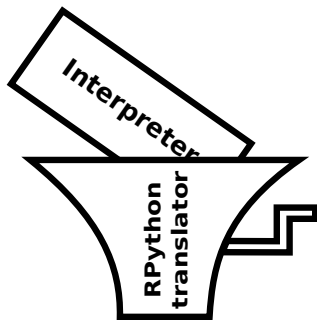


# Meta-tracing translation with RPython



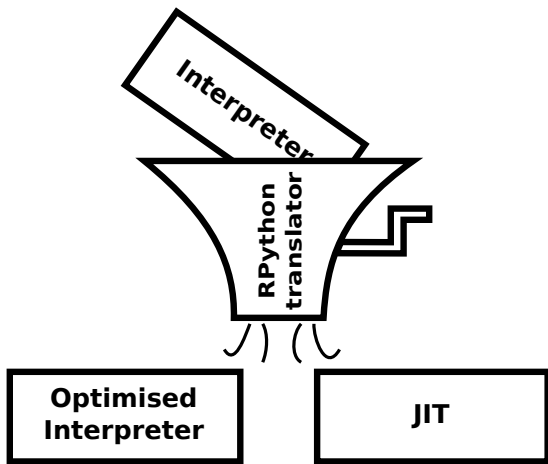
**Interpreter**

# Meta-tracing translation with RPython

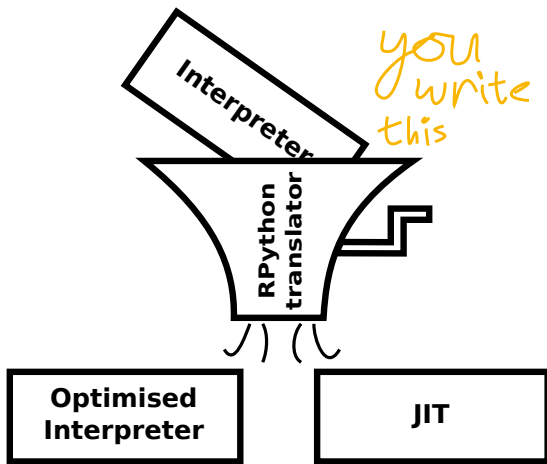




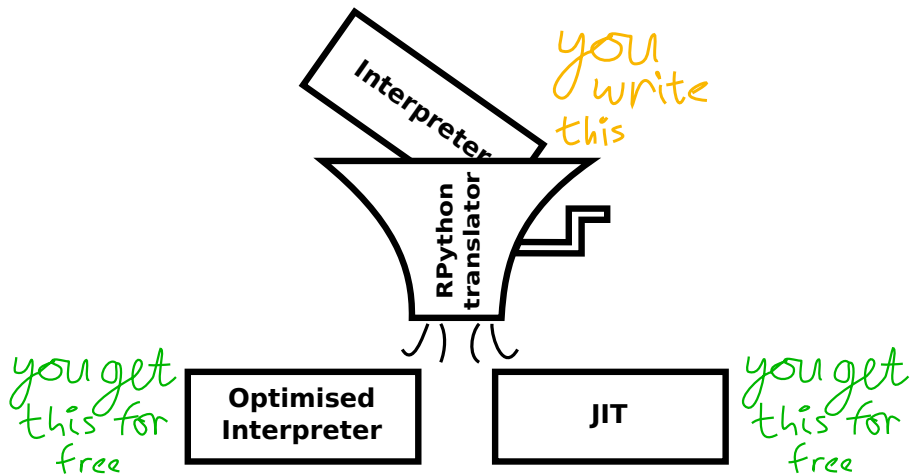
# Meta-tracing translation with RPython



# Meta-tracing translation with RPython

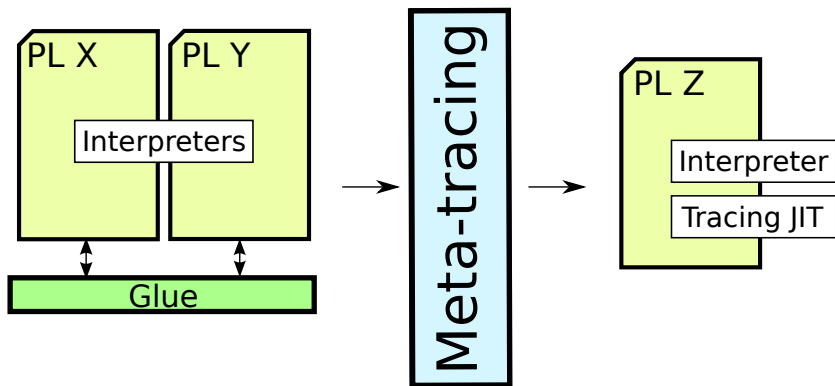


# Meta-tracing translation with RPython

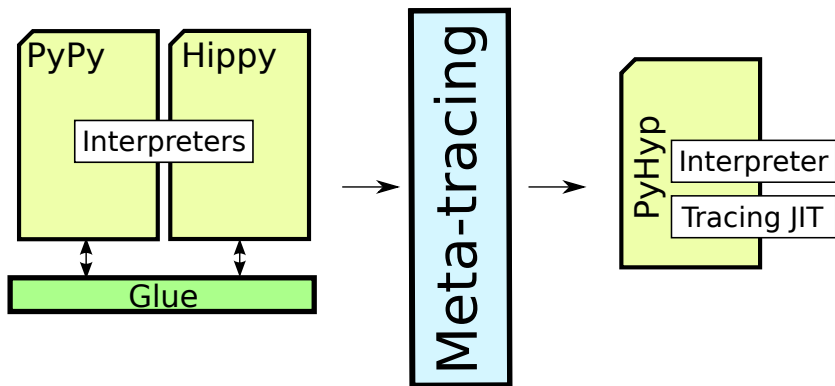


# Runtime composition recap

## Runtime composition recap



## Runtime composition recap



# Composed Richards vs. other VMs

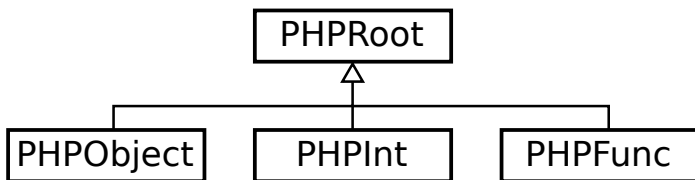
| Type | VM            |                    |
|------|---------------|--------------------|
| Mono | CPython 2.7.7 | $9.475 \pm 0.0127$ |
|      | HHVM 3.4.0    | $4.264 \pm 0.0386$ |
|      | HippyVM       | $0.250 \pm 0.0008$ |
|      | PyPy 2.4.0    | $0.178 \pm 0.0006$ |
|      | Zend 5.5.13   | $9.070 \pm 0.0361$ |

# Composed Richards vs. other VMs

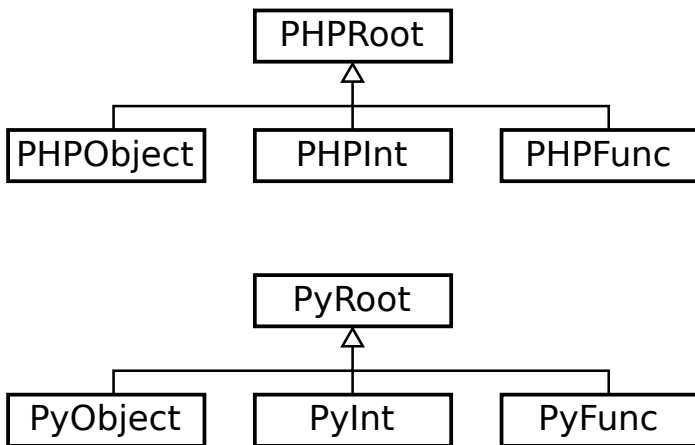
| Type     | VM            |                    |
|----------|---------------|--------------------|
| Mono     | CPython 2.7.7 | $9.475 \pm 0.0127$ |
|          | HHVM 3.4.0    | $4.264 \pm 0.0386$ |
|          | HippyVM       | $0.250 \pm 0.0008$ |
|          | PyPy 2.4.0    | $0.178 \pm 0.0006$ |
|          | Zend 5.5.13   | $9.070 \pm 0.0361$ |
| Composed | PyHyp         | $0.335 \pm 0.0012$ |



# Datatype conversion



# Datatype conversion



# Datatype conversion: primitive types

PHP

Python

# Datatype conversion: primitive types

PHP

2 : PHPInt

Python

# Datatype conversion: primitive types

PHP

2 : PHPInt

Python

2 : PyInt

# Datatype conversion: user types

PHP

Python

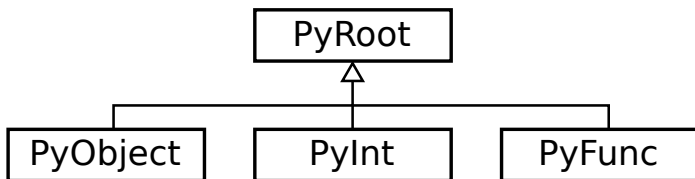
# Datatype conversion: user types

PHP

`o : PHPObject`

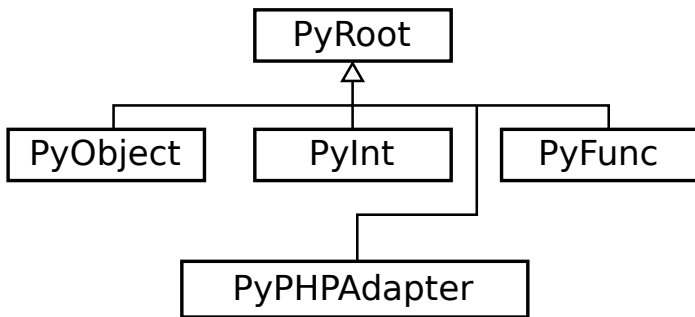
Python

# Datatype conversion: user types

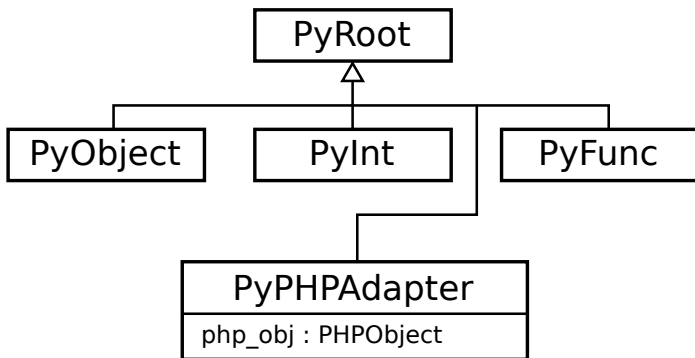




## Datatype conversion: user types



# Datatype conversion: user types



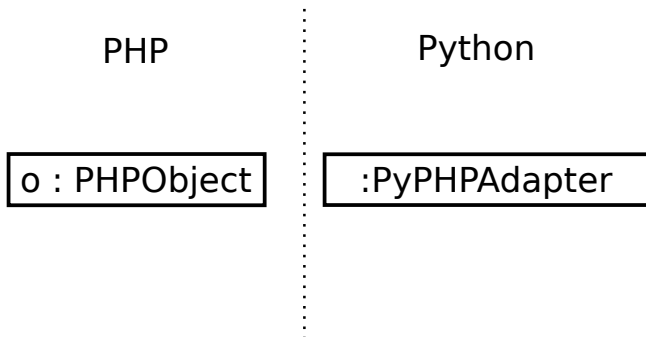
# Datatype conversion: user types

PHP

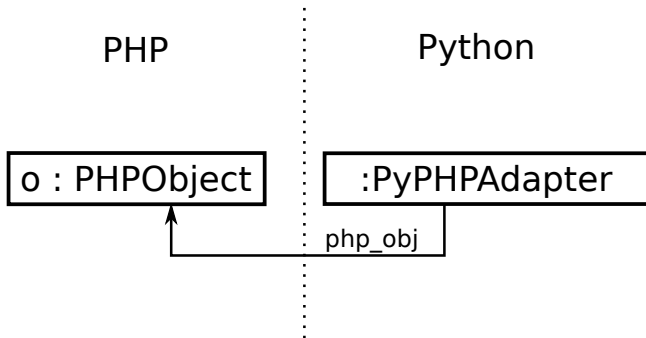
`o : PHPObject`

Python

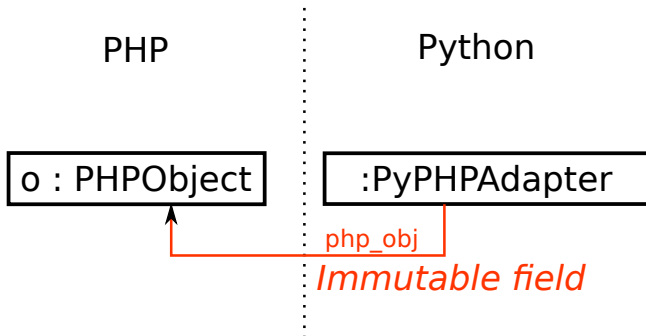
# Datatype conversion: user types



# Datatype conversion: user types



# Datatype conversion: user types



# Friction

A good composition needs to reduce *friction*.

# Friction

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

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



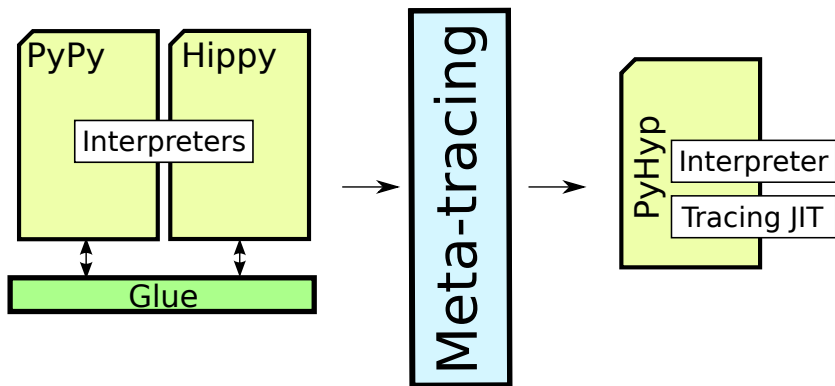
A good composition needs to reduce *friction*. Some examples:

- 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)

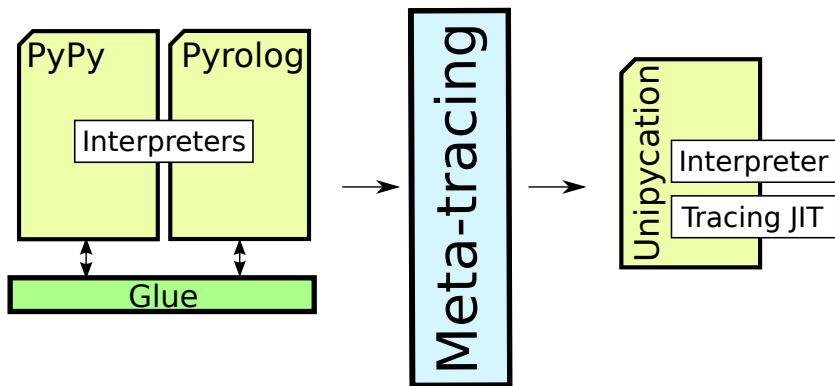
A good composition needs to reduce *friction*. Some examples:

- 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



# Unipycation



# Unipycation demo

# Absolute timing comparison

| VM              | Benchmark | <i>Python</i> |        | <i>Prolog</i> |         | <i>Python</i> → <i>Prolog</i> |         |
|-----------------|-----------|---------------|--------|---------------|---------|-------------------------------|---------|
| CPython-SWI     | SmallFunc | 0.125s        | ±0.007 | 0.257s        | ±0.002  | 28.893s                       | ±0.227  |
|                 | L1A0R     | 2.924s        | ±0.284 | 7.352s        | ±0.048  | 9.310s                        | ±0.084  |
|                 | L1A1R     | 4.184s        | ±0.038 | 18.890s       | ±0.111  | 20.865s                       | ±0.067  |
|                 | NdL1A1R   | 7.531s        | ±0.080 | 18.643s       | ±0.197  | 667.682s                      | ±6.895  |
|                 | TCons     | 264.415s      | ±2.250 | 48.819s       | ±0.252  | 2185.150s                     | ±18.225 |
|                 | Lists     | 9.374s        | ±0.059 | 25.148s       | ±0.221  | 2207.304s                     | ±16.073 |
| Unipycation     | SmallFunc | 0.001s        | ±0.000 | 0.006s        | ±0.001  | 0.001s                        | ±0.000  |
|                 | L1A0R     | 0.085s        | ±0.000 | 0.086s        | ±0.000  | 0.087s                        | ±0.000  |
|                 | L1A1R     | 0.112s        | ±0.000 | 0.114s        | ±0.000  | 0.115s                        | ±0.000  |
|                 | NdL1A1R   | 0.500s        | ±0.003 | 0.548s        | ±0.085  | 2.674s                        | ±0.012  |
|                 | TCons     | 6.053s        | ±0.288 | 2.444s        | ±0.003  | 36.069s                       | ±0.225  |
|                 | Lists     | 0.845s        | ±0.002 | 1.416s        | ±0.003  | 5.056s                        | ±0.035  |
| Jython-tuProlog | SmallFunc | 0.088s        | ±0.003 | 3.050s        | ±0.053  | 52.294s                       | ±0.475  |
|                 | L1A0R     | 1.078s        | ±0.009 | 206.590s      | ±3.846  | 199.963s                      | ±2.476  |
|                 | L1A1R     | 2.145s        | ±0.232 | 293.311s      | ±5.691  | 294.781s                      | ±6.193  |
|                 | NdL1A1R   | 7.939s        | ±0.457 | 1857.687s     | ±5.169  | 1990.985s                     | ±15.071 |
|                 | TCons     | 543.347s      | ±3.289 | 8014.477s     | ±17.710 | 8202.362s                     | ±24.904 |
|                 | Lists     | 5.661s        | ±0.046 | 6981.873s     | ±18.795 | 5577.322s                     | ±15.754 |

# Relative timing comparison

| VM              | Benchmark | $\frac{Python \rightarrow Prolog}{Python}$ |         | $\frac{Python \rightarrow Prolog}{Prolog}$ |        | $\frac{Python \rightarrow Prolog}{Unipycation}$ |           |
|-----------------|-----------|--------------------------------------------|---------|--------------------------------------------|--------|-------------------------------------------------|-----------|
| CPython-SWI     | SmallFunc | 231.770×                                   | ±13.136 | 112.567×                                   | ±1.242 | 27821.079×                                      | ±2331.665 |
|                 | L1A0R     | 3.184×                                     | ±0.300  | 1.266×                                     | ±0.014 | 107.591×                                        | ±0.995    |
|                 | L1A1R     | 4.987×                                     | ±0.049  | 1.105×                                     | ±0.007 | 181.899×                                        | ±0.590    |
|                 | NdL1A1R   | 88.654×                                    | ±1.368  | 35.814×                                    | ±0.554 | 249.737×                                        | ±2.922    |
|                 | TCons     | 8.264×                                     | ±0.101  | 44.760×                                    | ±0.453 | 60.583×                                         | ±0.637    |
|                 | Lists     | 235.459×                                   | ±2.314  | 87.772×                                    | ±1.017 | 436.609×                                        | ±4.415    |
| Unipycation     | SmallFunc | 1.295×                                     | ±0.105  | 0.182×                                     | ±0.054 | 1.000×                                          |           |
|                 | L1A0R     | 1.020×                                     | ±0.002  | 1.012×                                     | ±0.002 | 1.000×                                          |           |
|                 | L1A1R     | 1.025×                                     | ±0.002  | 1.002×                                     | ±0.003 | 1.000×                                          |           |
|                 | NdL1A1R   | 5.349×                                     | ±0.045  | 4.879×                                     | ±0.924 | 1.000×                                          |           |
|                 | TCons     | 5.959×                                     | ±0.282  | 14.756×                                    | ±0.092 | 1.000×                                          |           |
|                 | Lists     | 5.982×                                     | ±0.045  | 3.569×                                     | ±0.026 | 1.000×                                          |           |
| Jython-tuProlog | SmallFunc | 592.904×                                   | ±19.517 | 17.143×                                    | ±0.338 | 50354.204×                                      | ±4341.413 |
|                 | L1A0R     | 185.460×                                   | ±2.818  | 0.968×                                     | ±0.021 | 2310.844×                                       | ±28.093   |
|                 | L1A1R     | 137.427×                                   | ±14.537 | 1.005×                                     | ±0.028 | 2569.873×                                       | ±52.847   |
|                 | NdL1A1R   | 250.776×                                   | ±14.666 | 1.072×                                     | ±0.009 | 744.699×                                        | ±6.726    |
|                 | TCons     | 15.096×                                    | ±0.106  | 1.023×                                     | ±0.004 | 227.409×                                        | ±1.592    |
|                 | Lists     | 985.149×                                   | ±8.674  | 0.799×                                     | ±0.003 | 1103.206×                                       | ±8.338    |

# What can we use this for?



What can we use this for?

First-class languages

# What can we use this for?

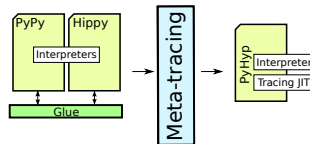
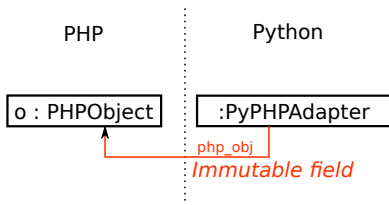
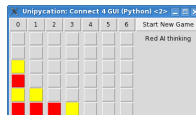
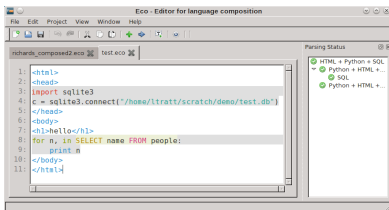
First-class languages

Language migration

# Thanks to our funders

- EPSRC: *COOLER* and *Lecture*.
- Oracle: various.

# Thanks for listening



<http://soft-dev.org/>