Language integration and migration


Edd Barrett


Carl Friedrich Bolz


Lukas Diekmann


Laurence Tratt


Naveneetha Krishnan Vasudevan

KING'S College LONDON

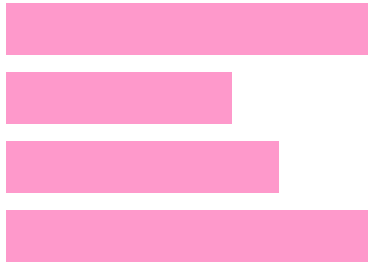
Software Development Team

2014-10-15

What to expect from this talk

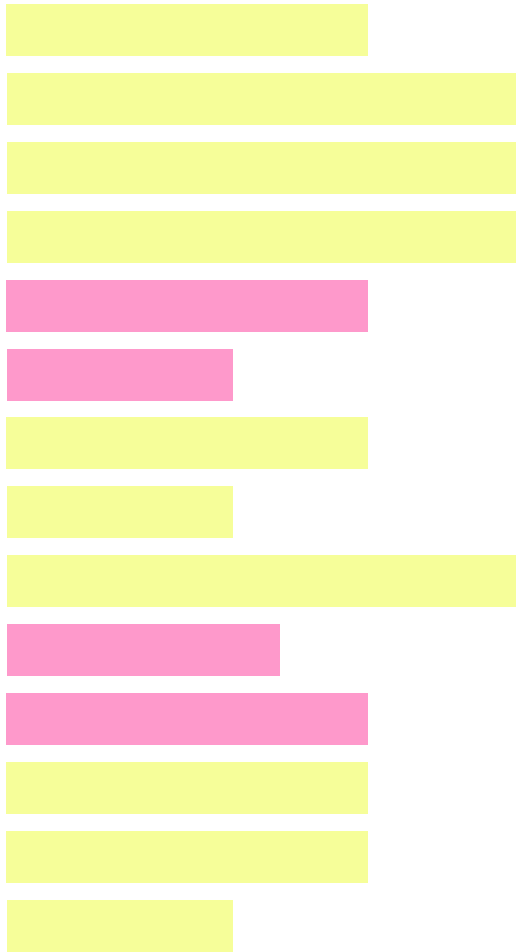
A

B



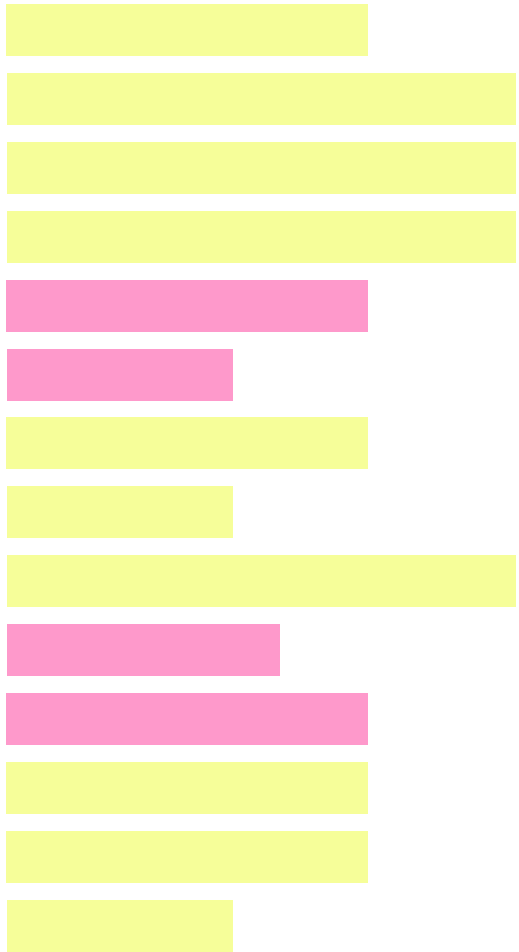
What to expect from this talk

$$A \cup B$$



What to expect from this talk

Python ∪ Prolog



What to expect from this talk

Python ∪ PHP

Our problem

Our problem

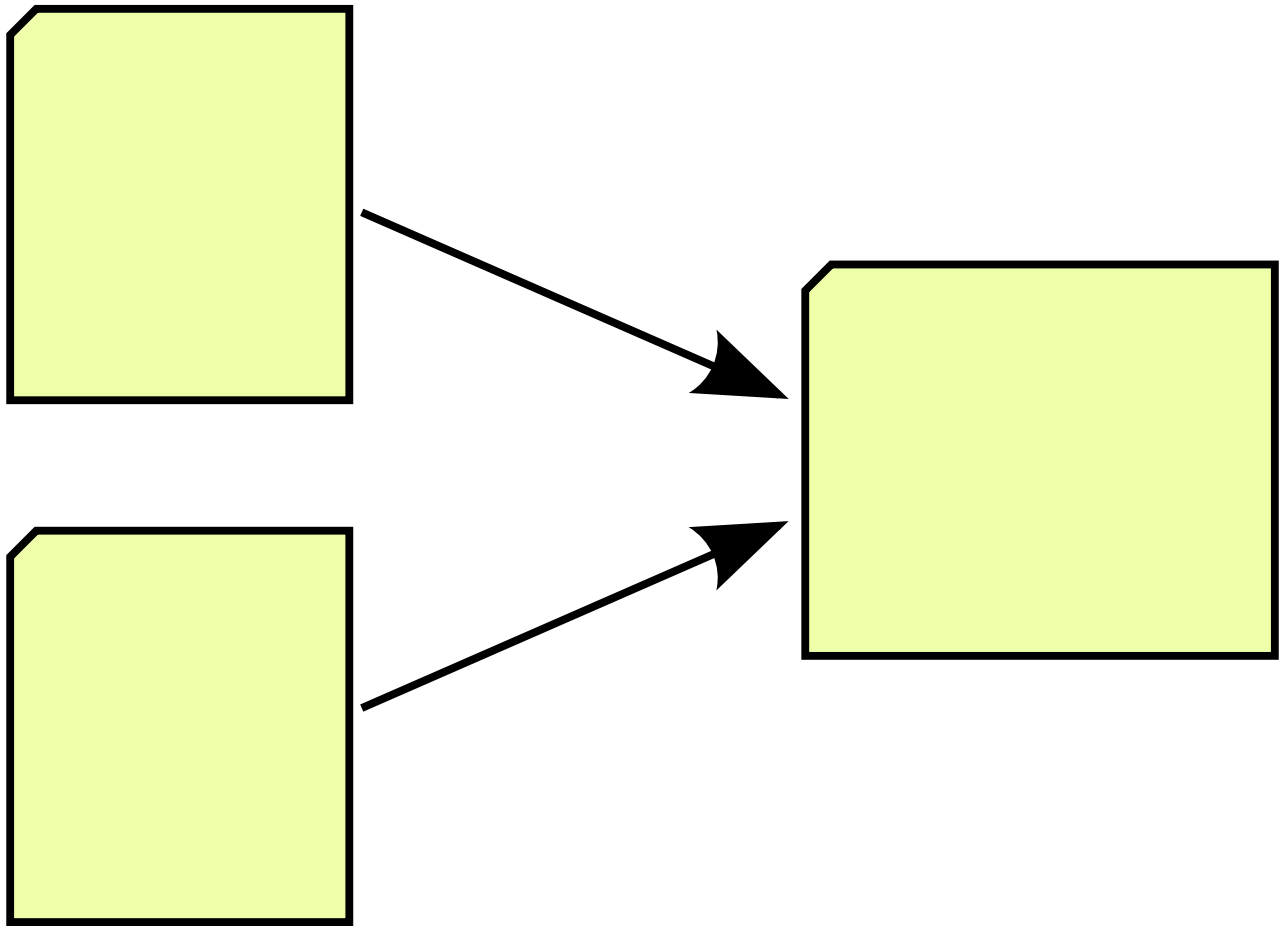
We want **better** programming languages

Our problem

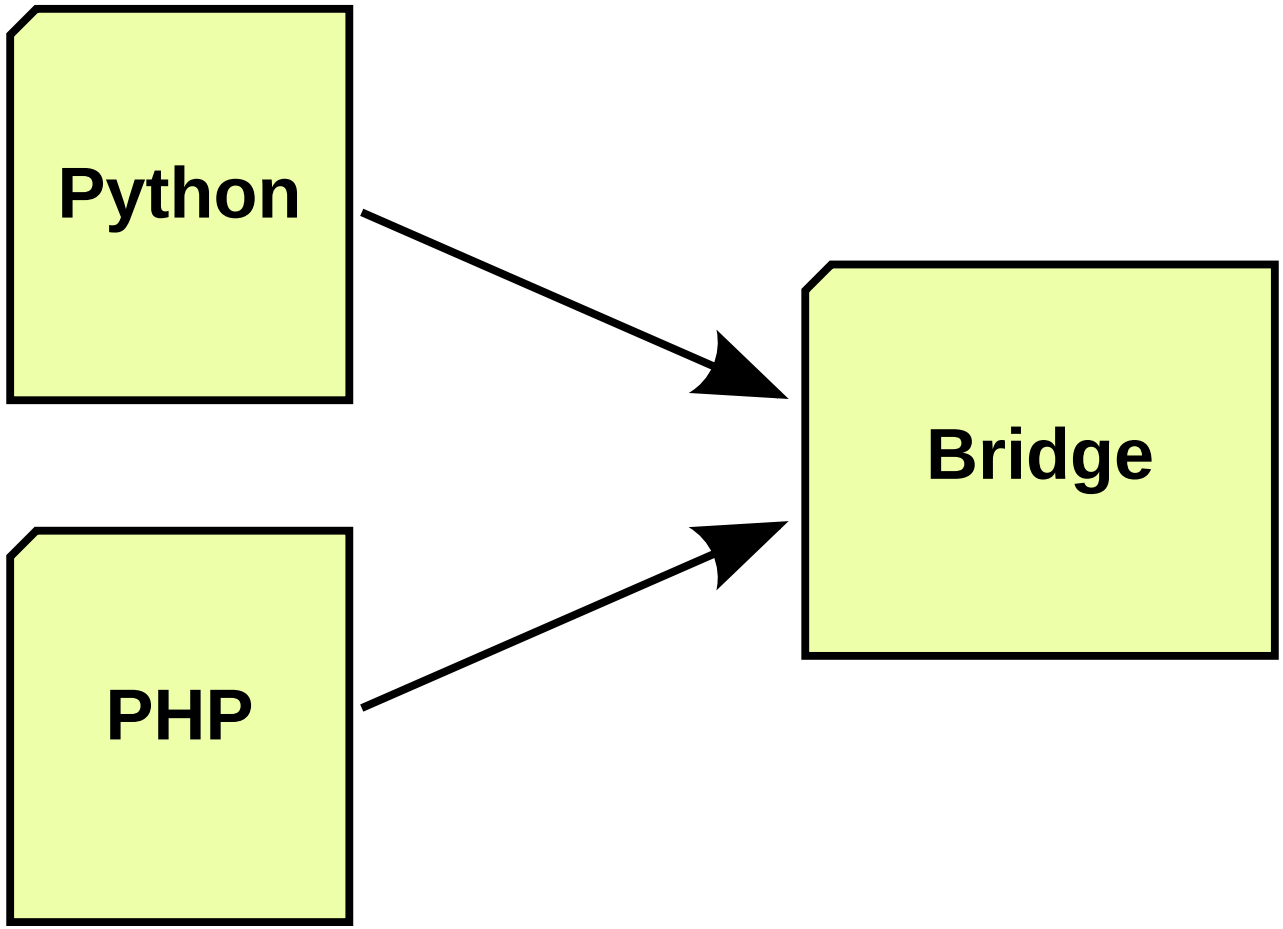
We want **better** programming languages

But better always seems to end up **bigger**

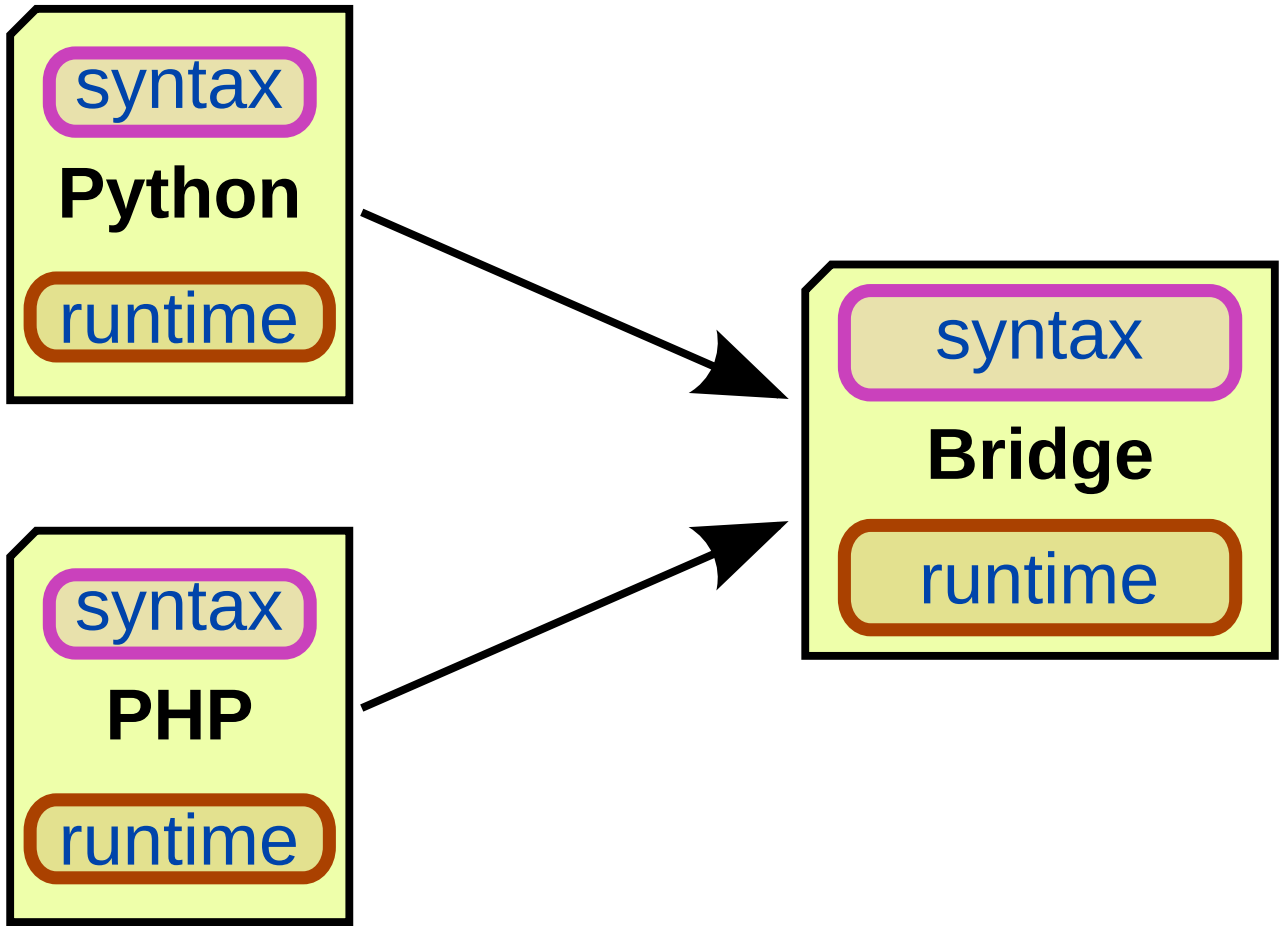
Underlying language composition challenges



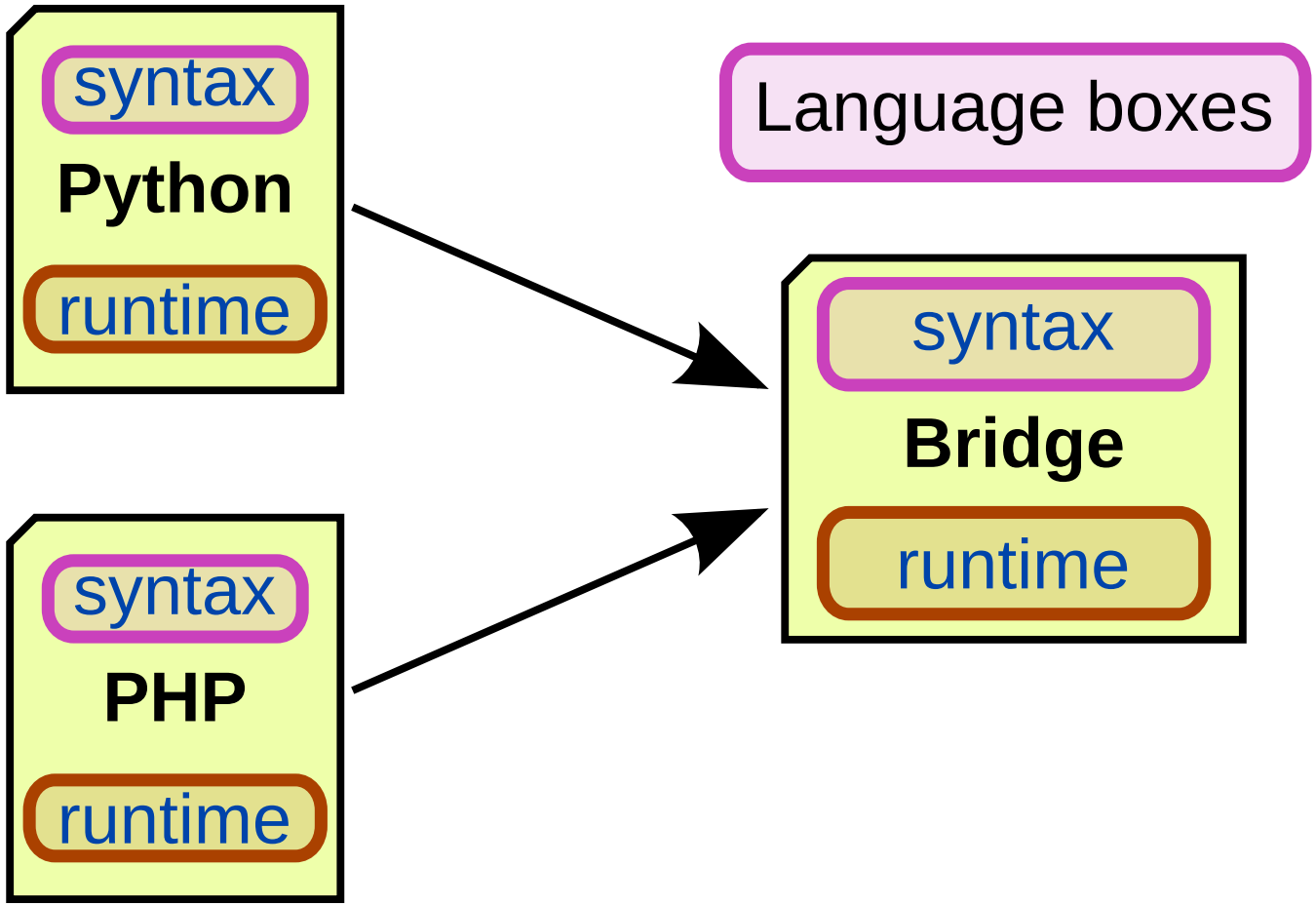
Underlying language composition challenges



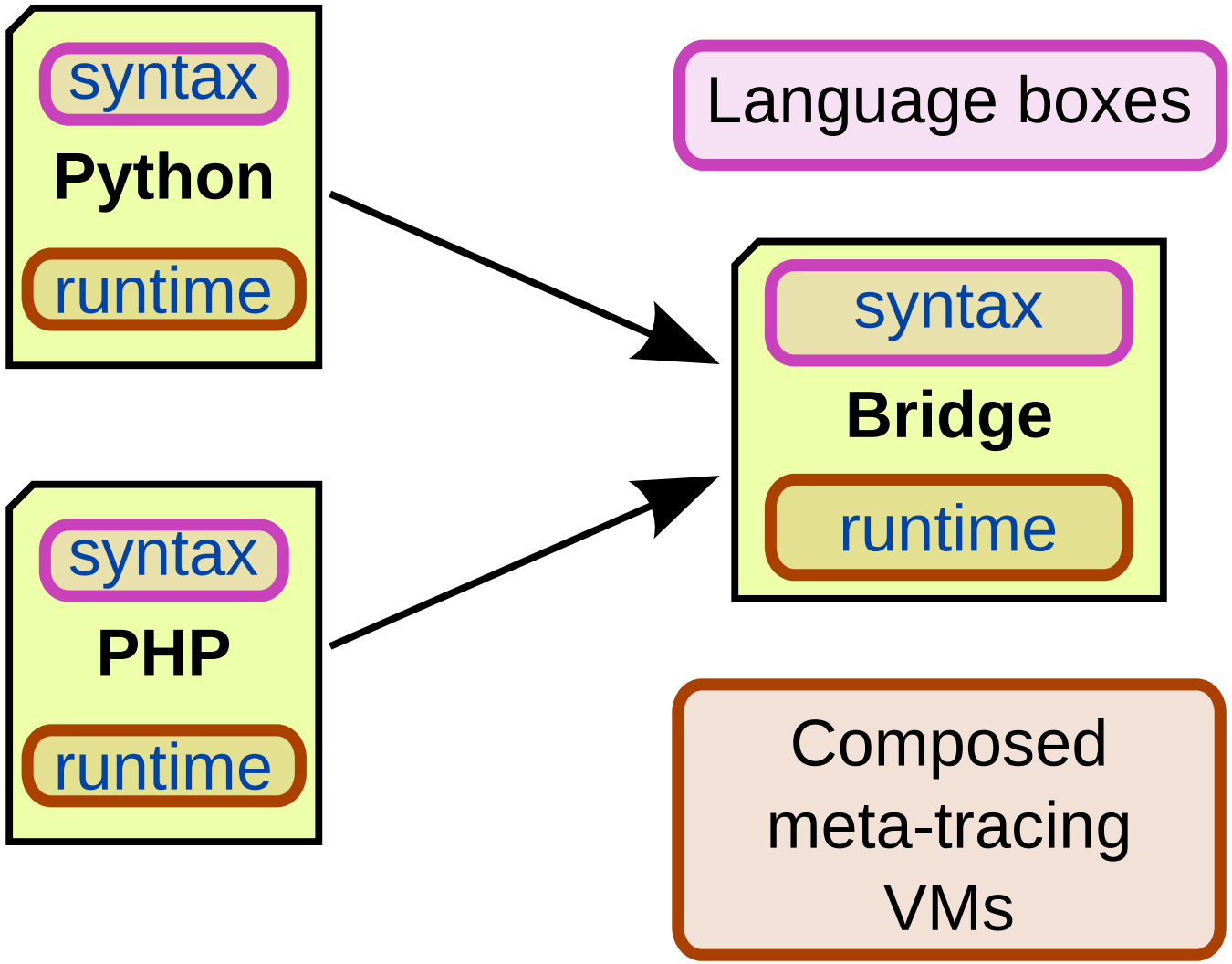
Underlying language composition challenges



Underlying language composition challenges



Underlying language composition challenges



Syntax composition

PL X

<grammar>

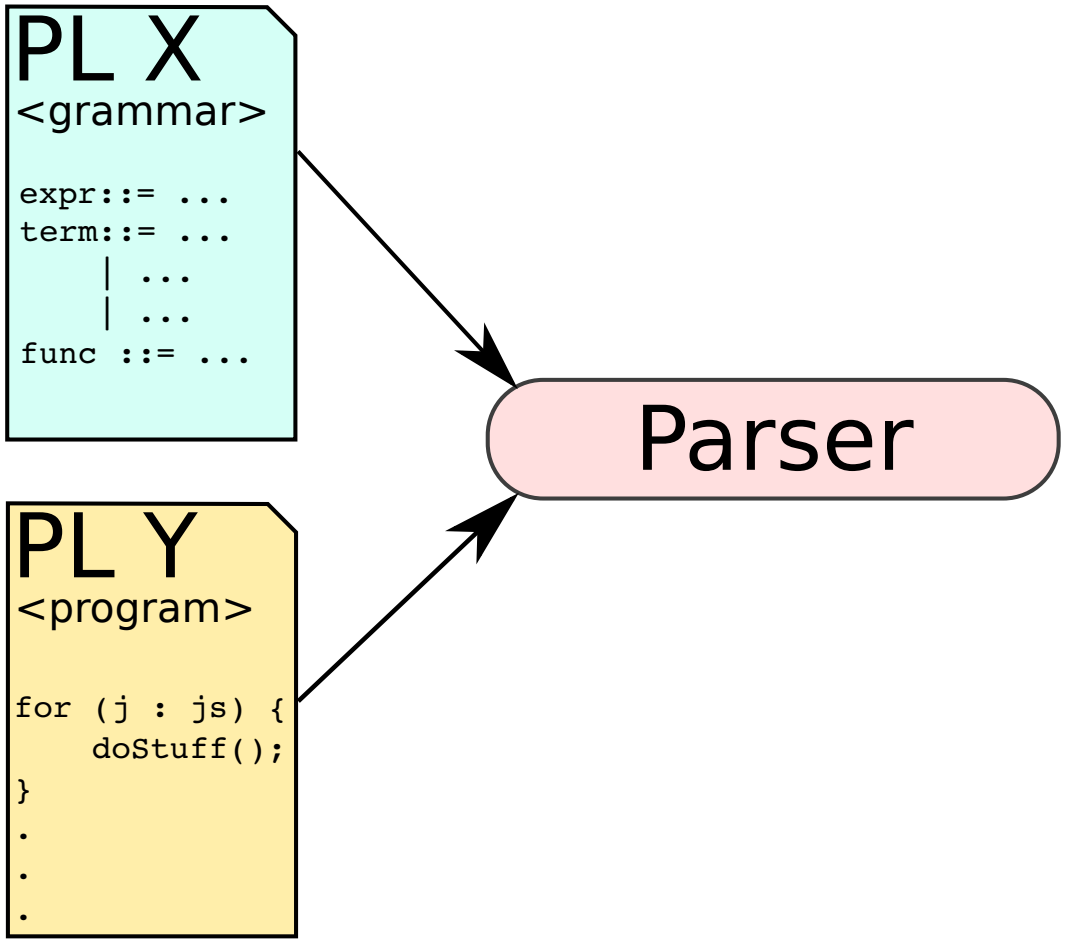
```
expr::= ...
term::= ...
    | ...
    | ...
func ::= ...
```

PL Y

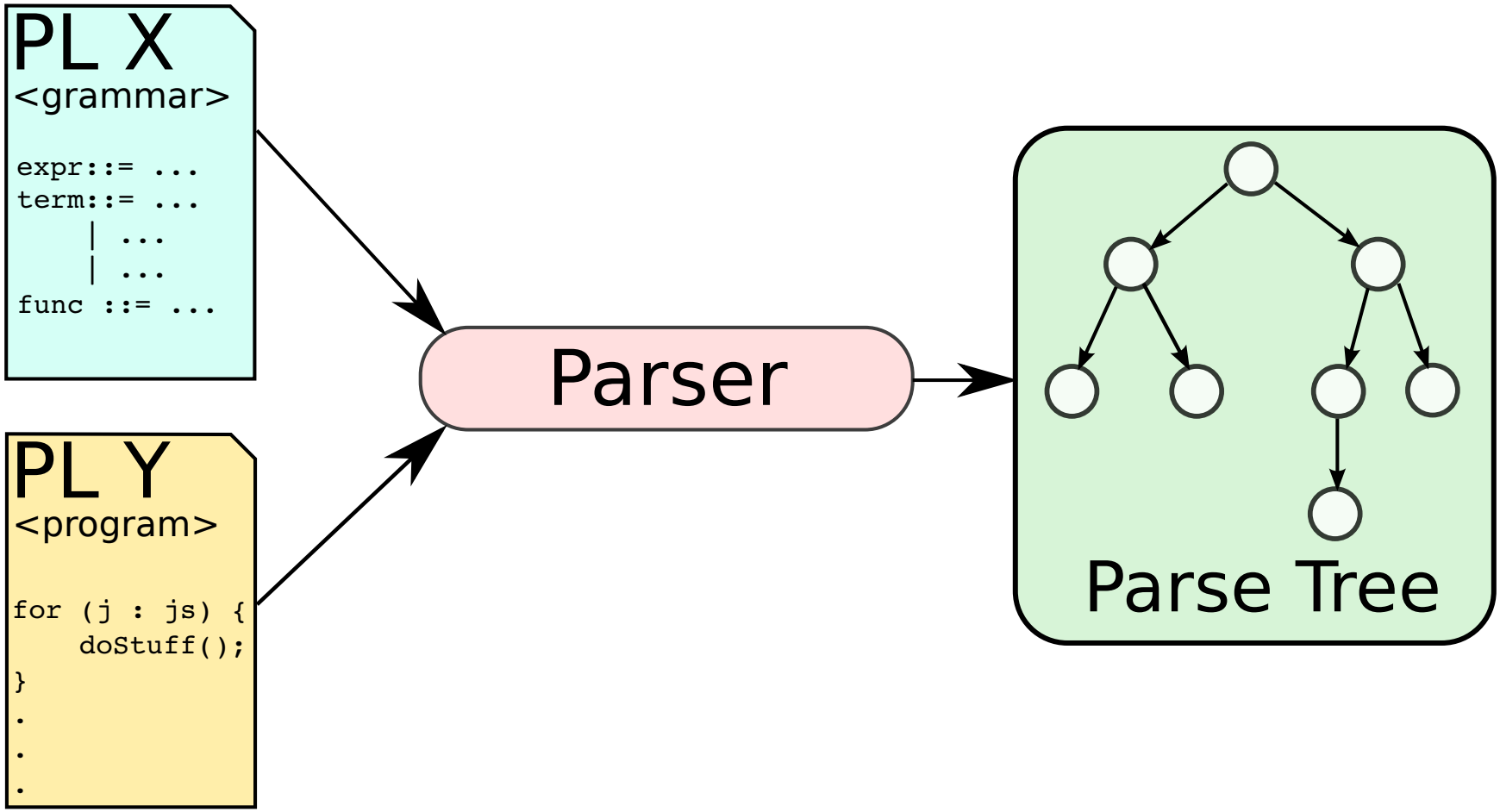
<program>

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

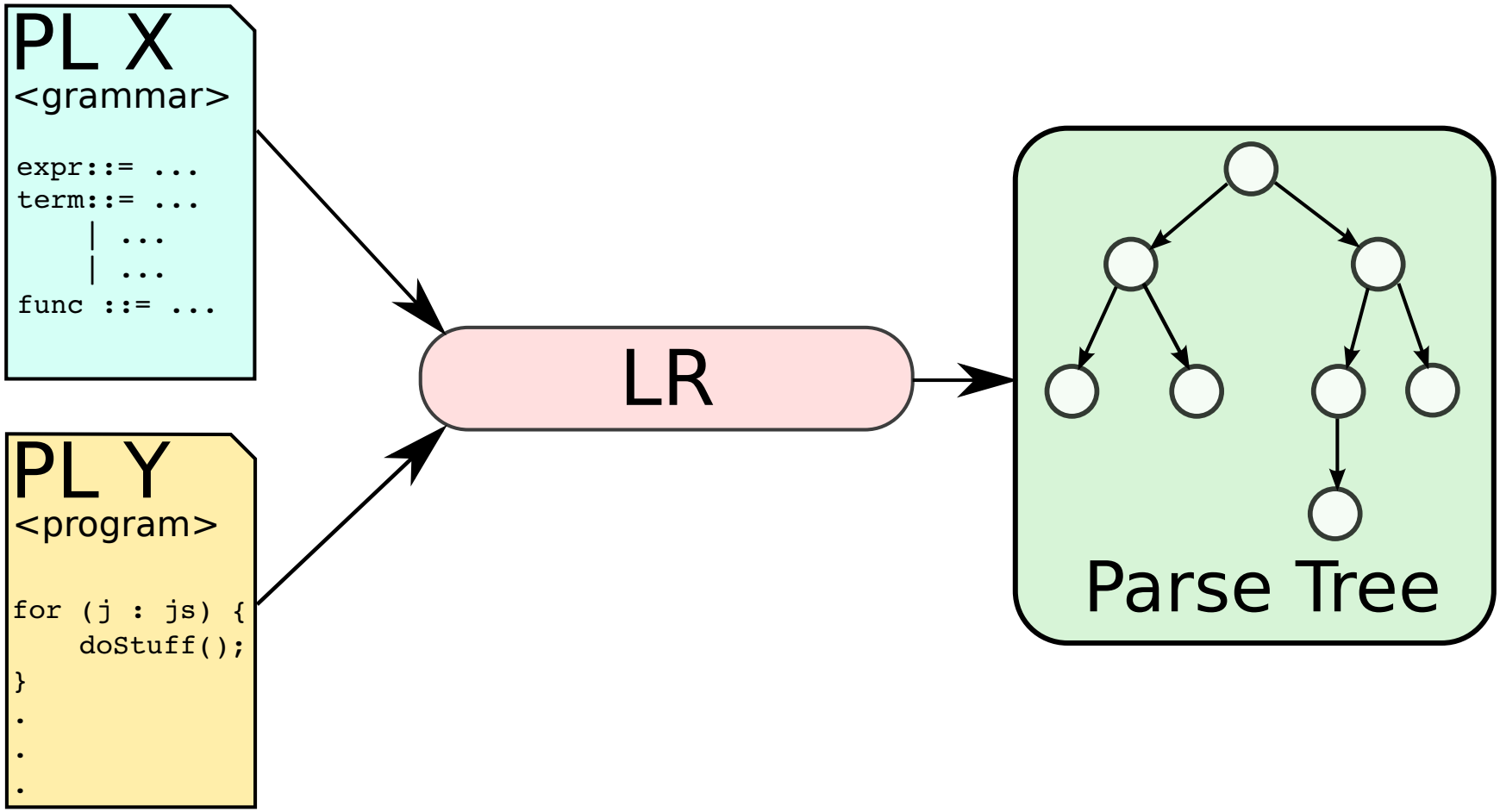
Syntax composition



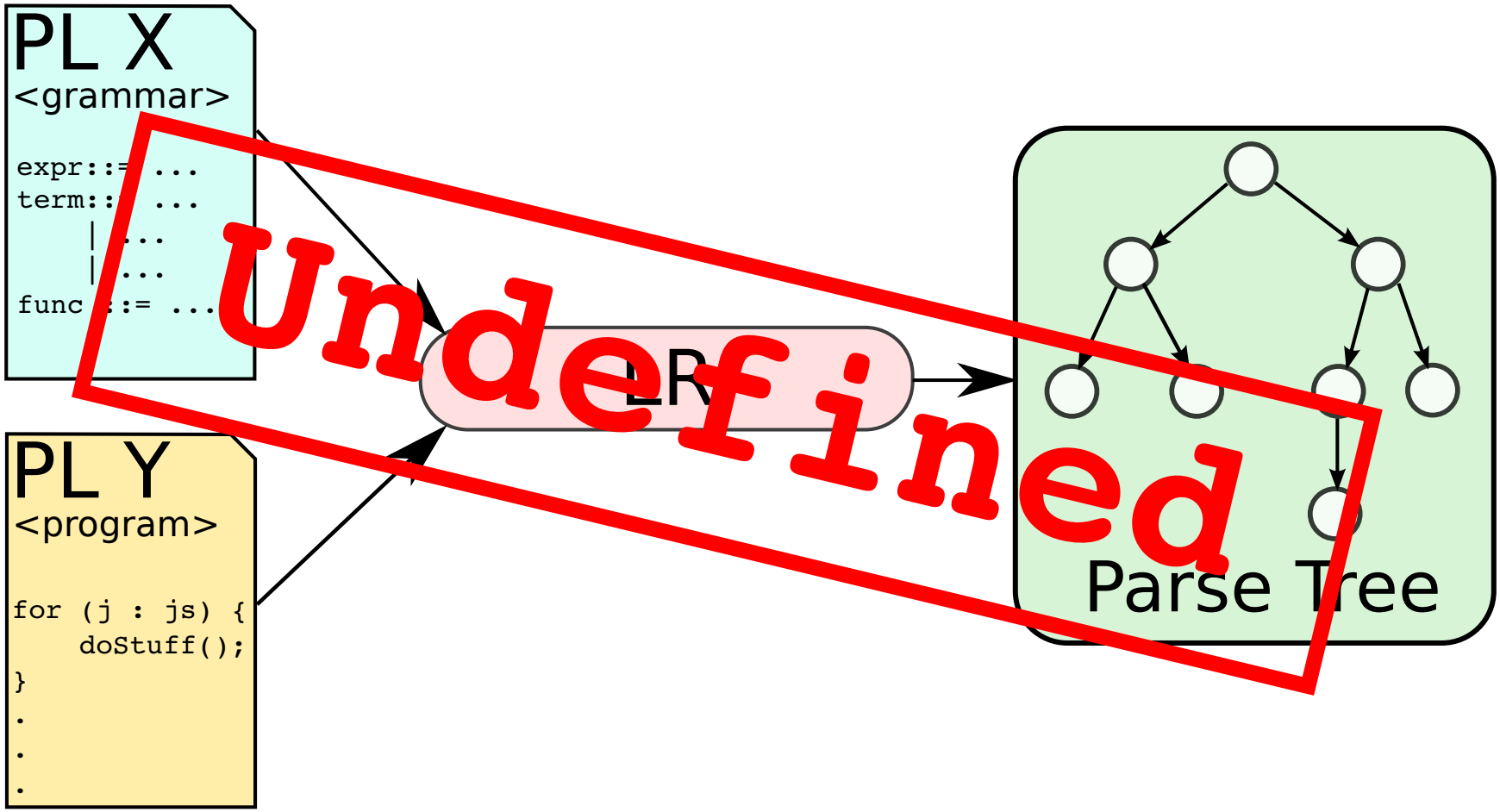
Syntax composition



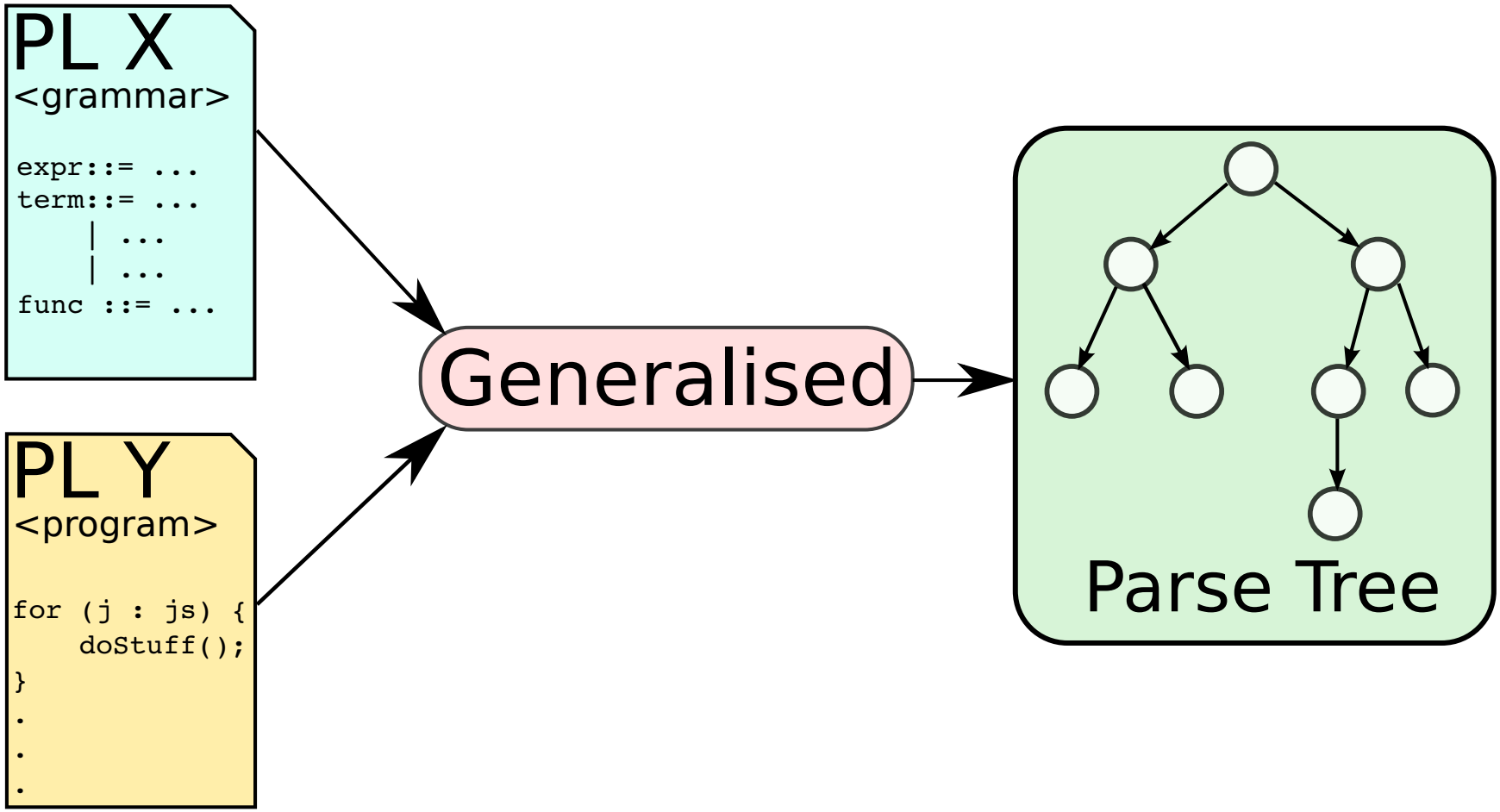
Syntax composition



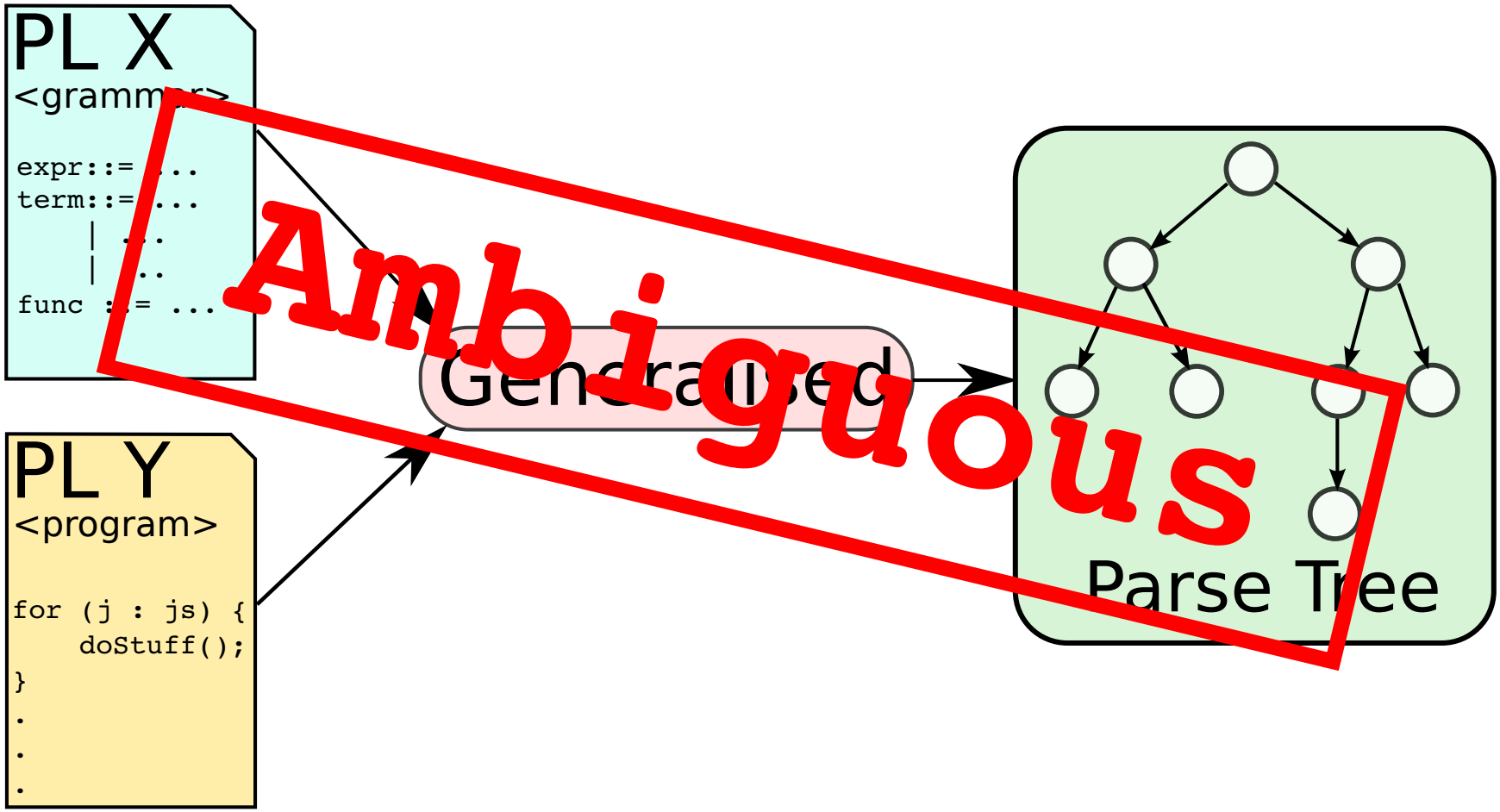
Syntax composition



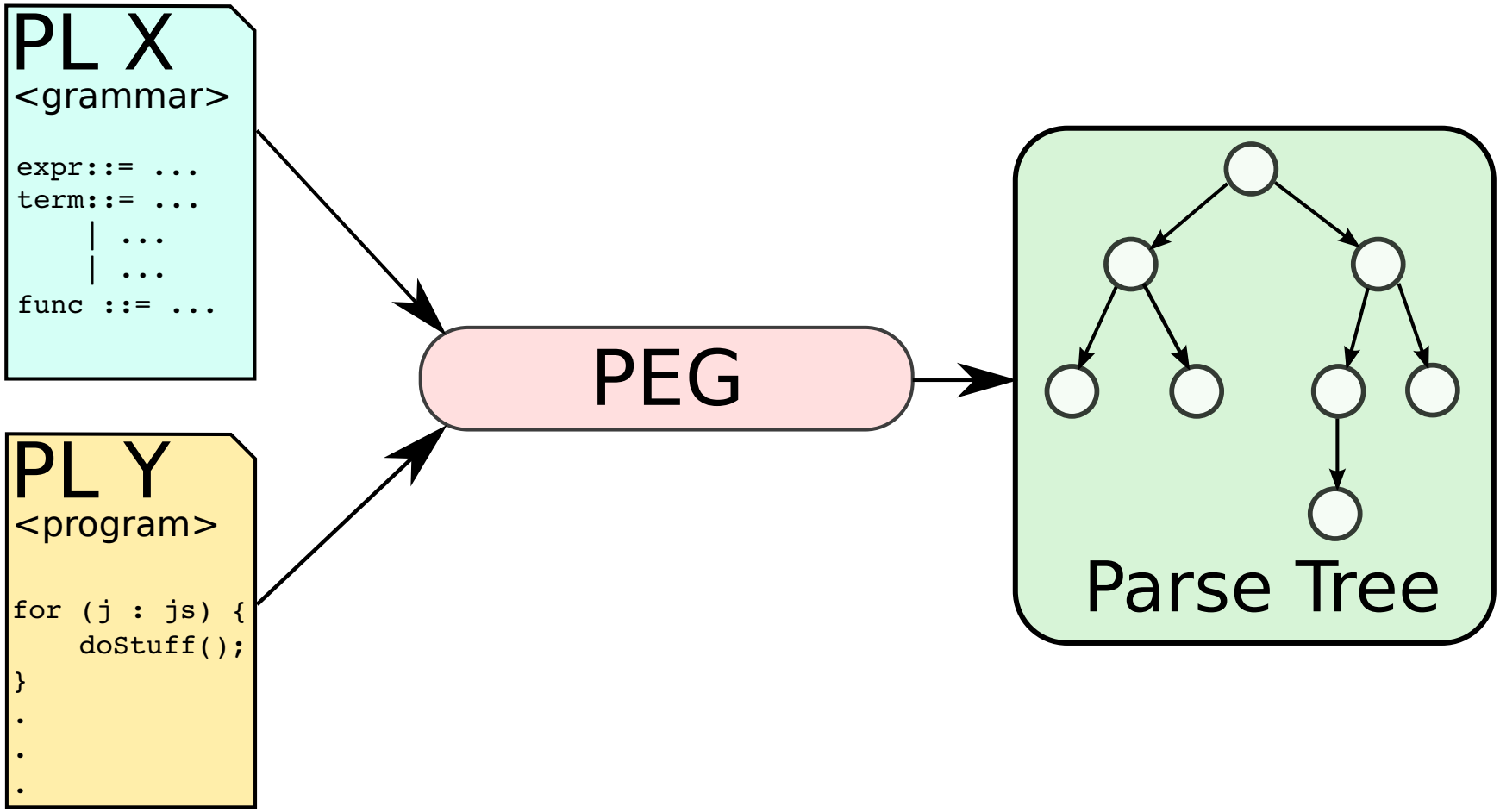
Syntax composition



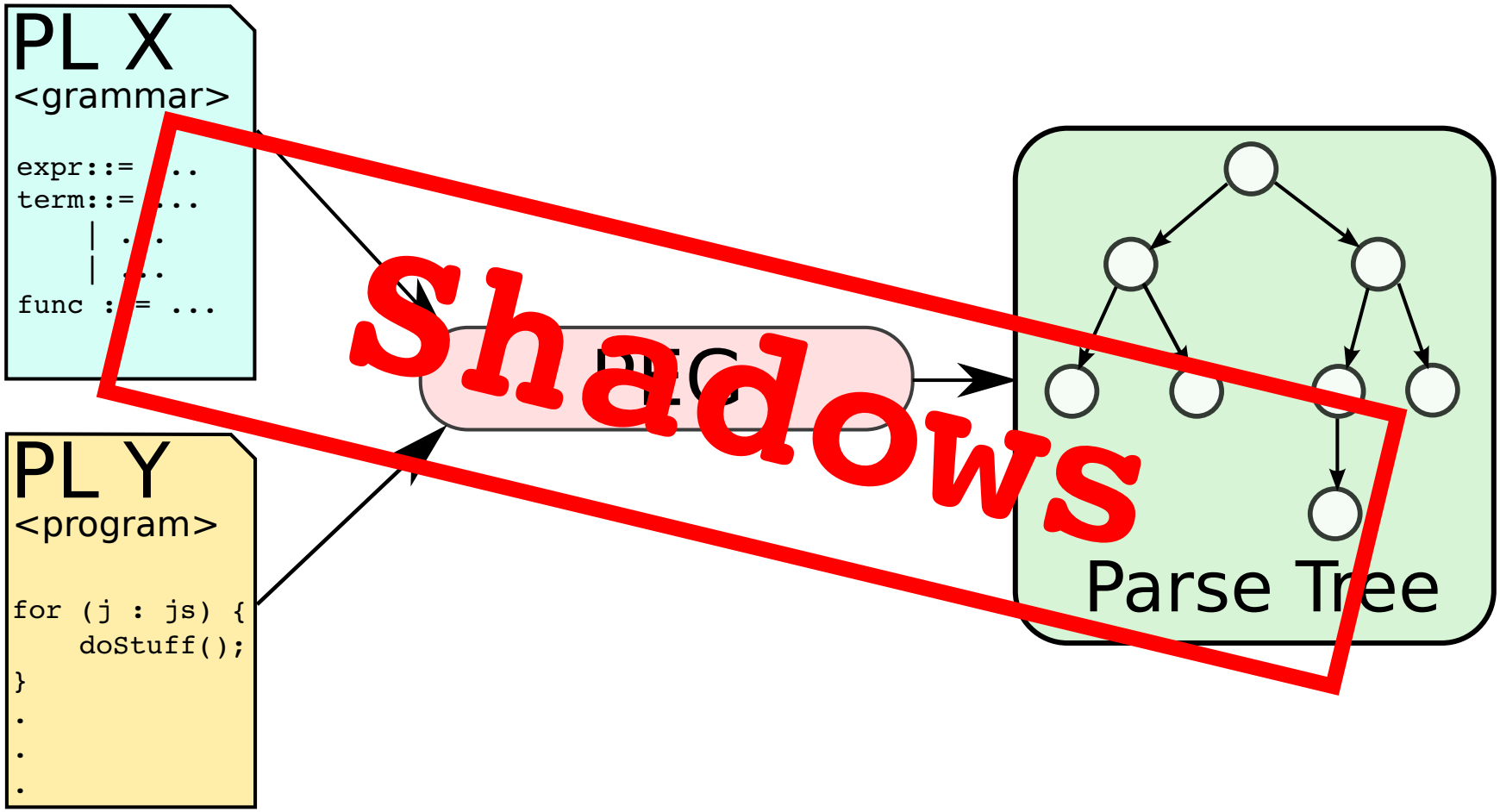
Syntax composition



Syntax composition



Syntax composition



The only choice?

The only choice?

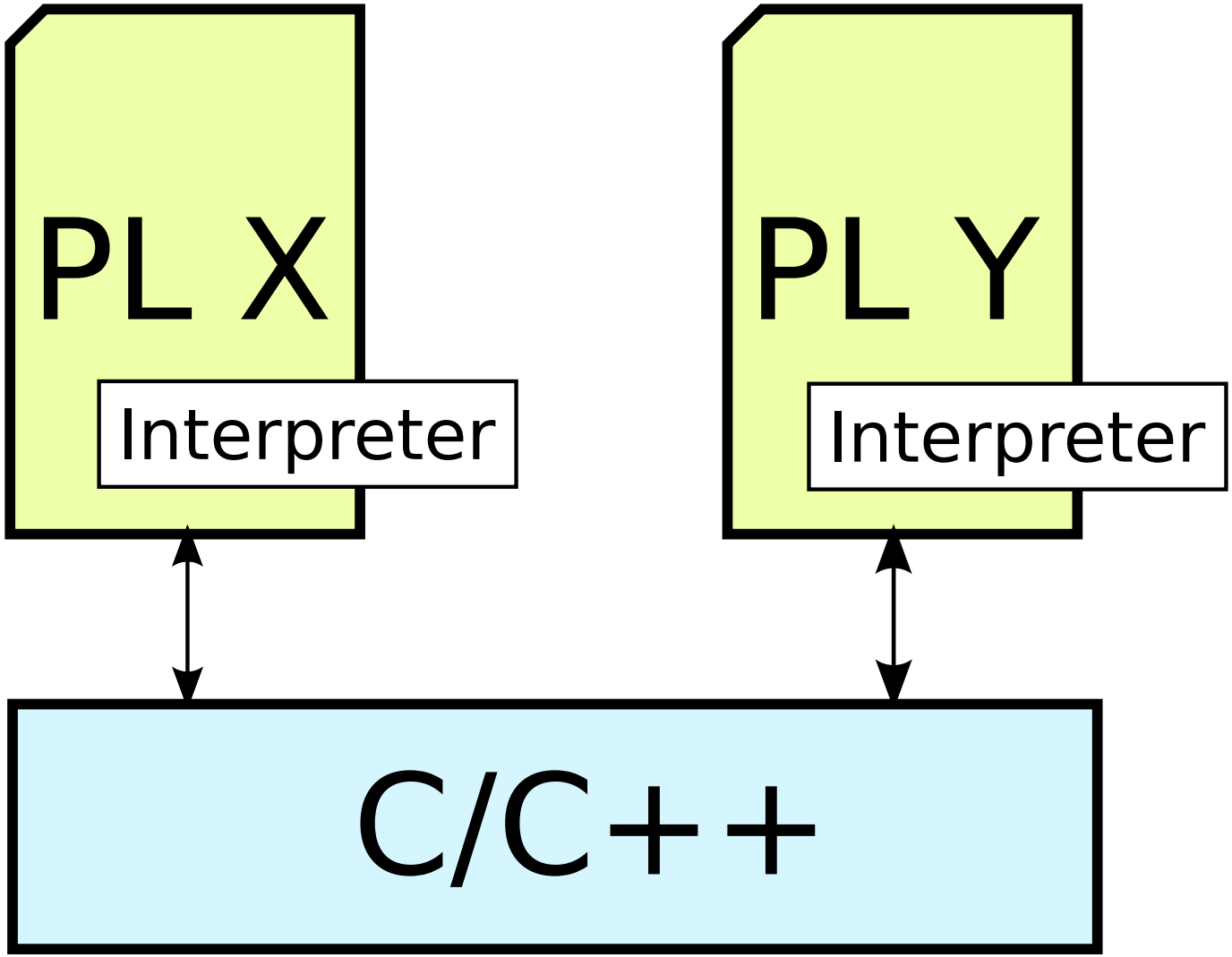
SDE

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

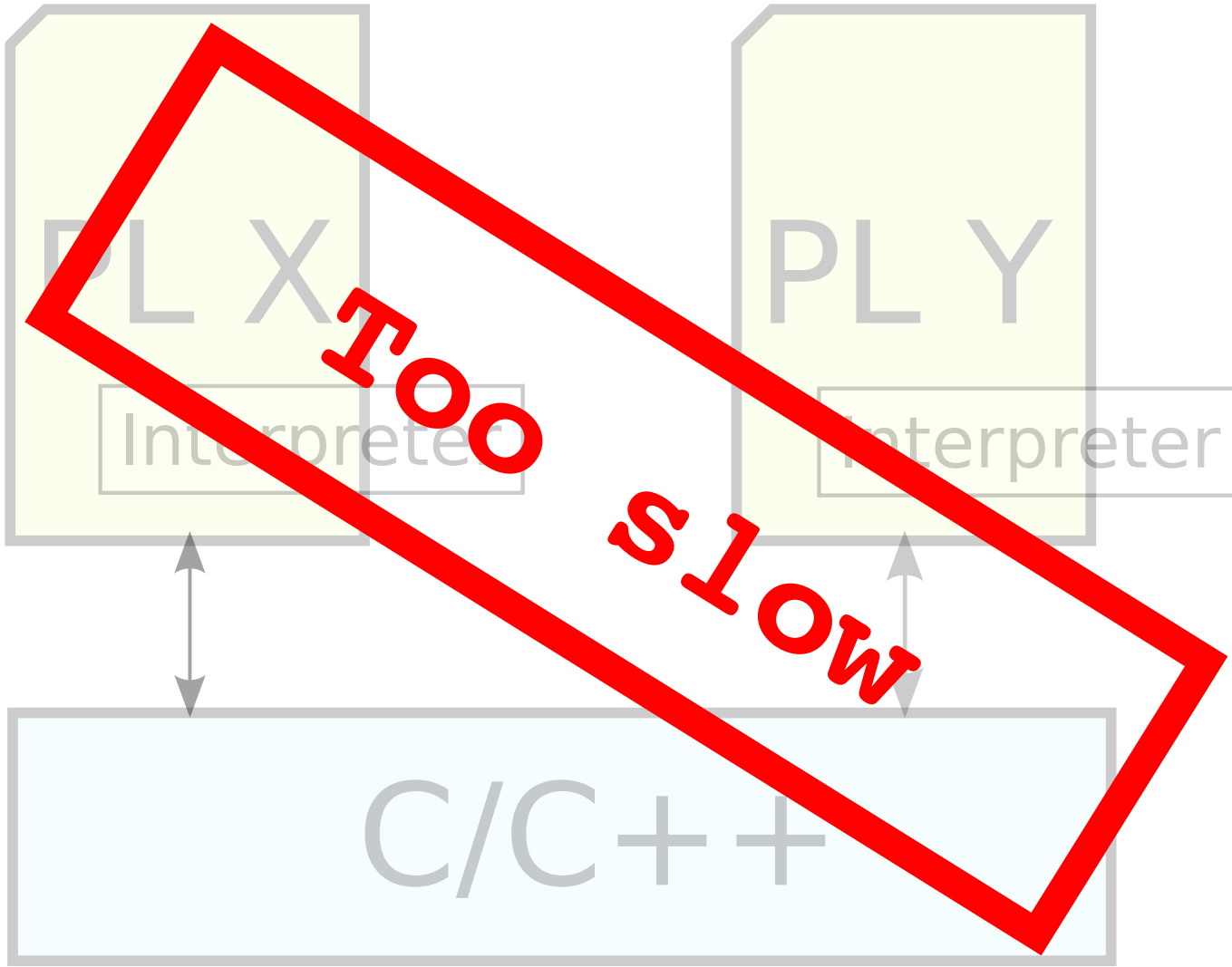
Eco demo

Runtime composition

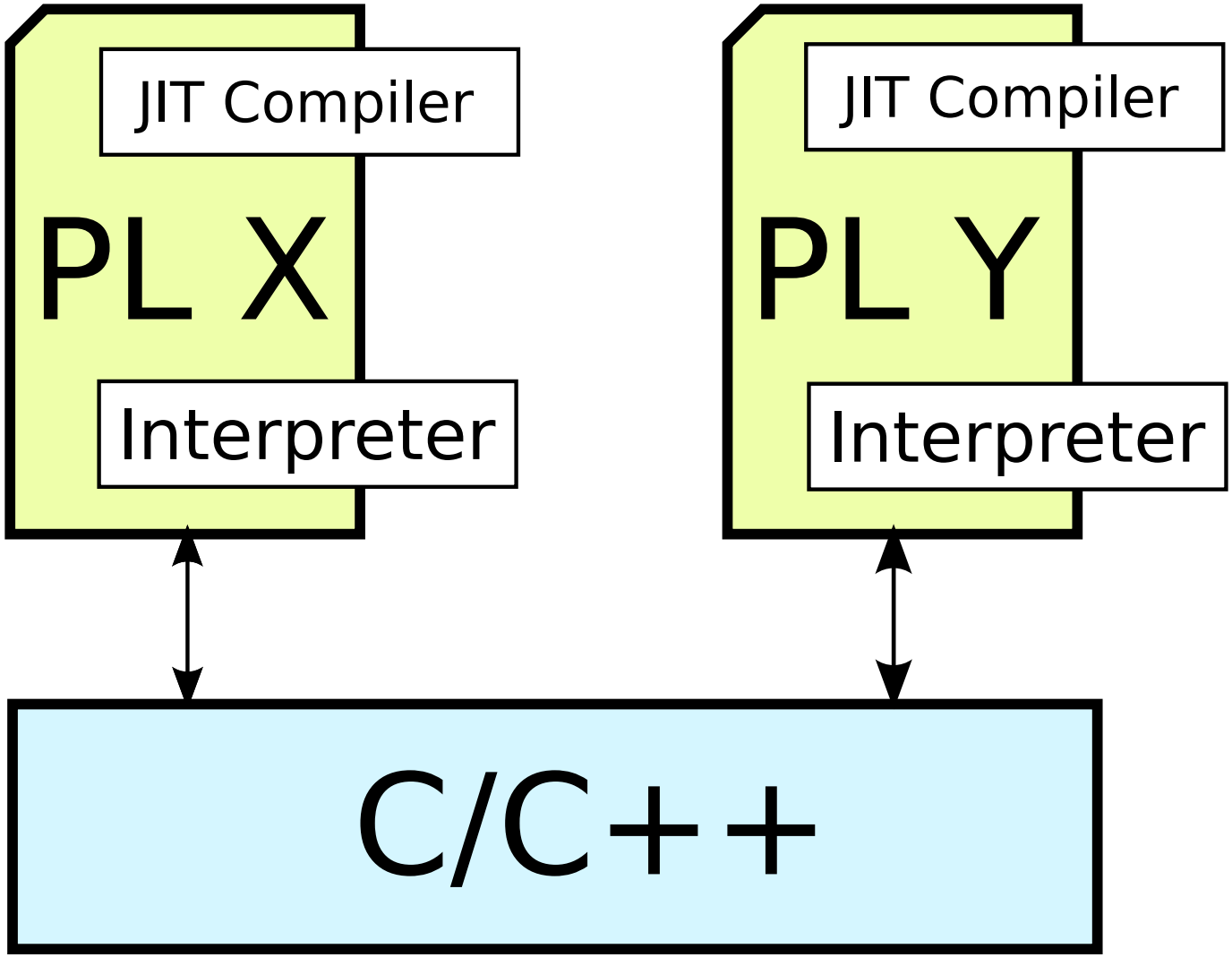
Runtime composition



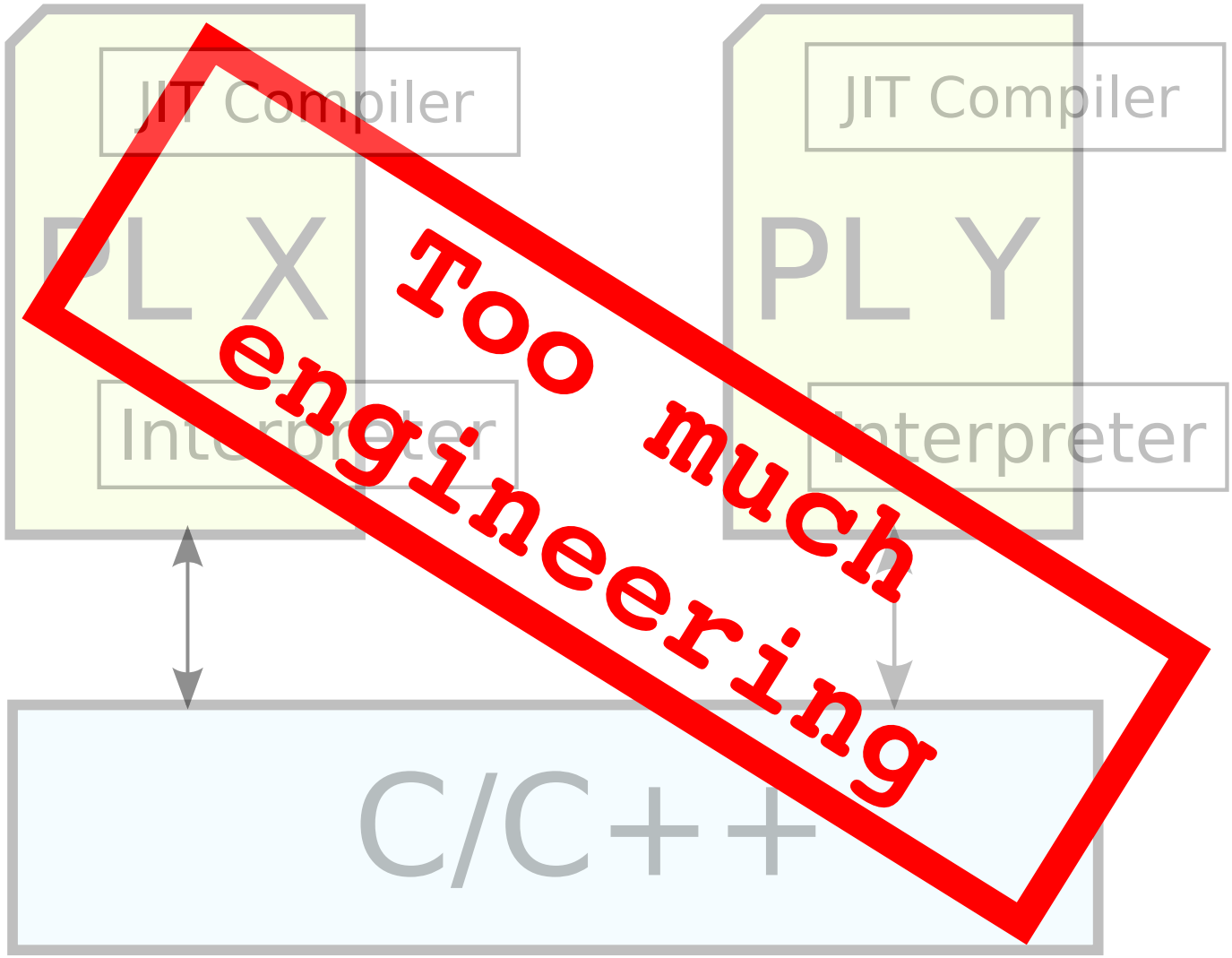
Runtime composition



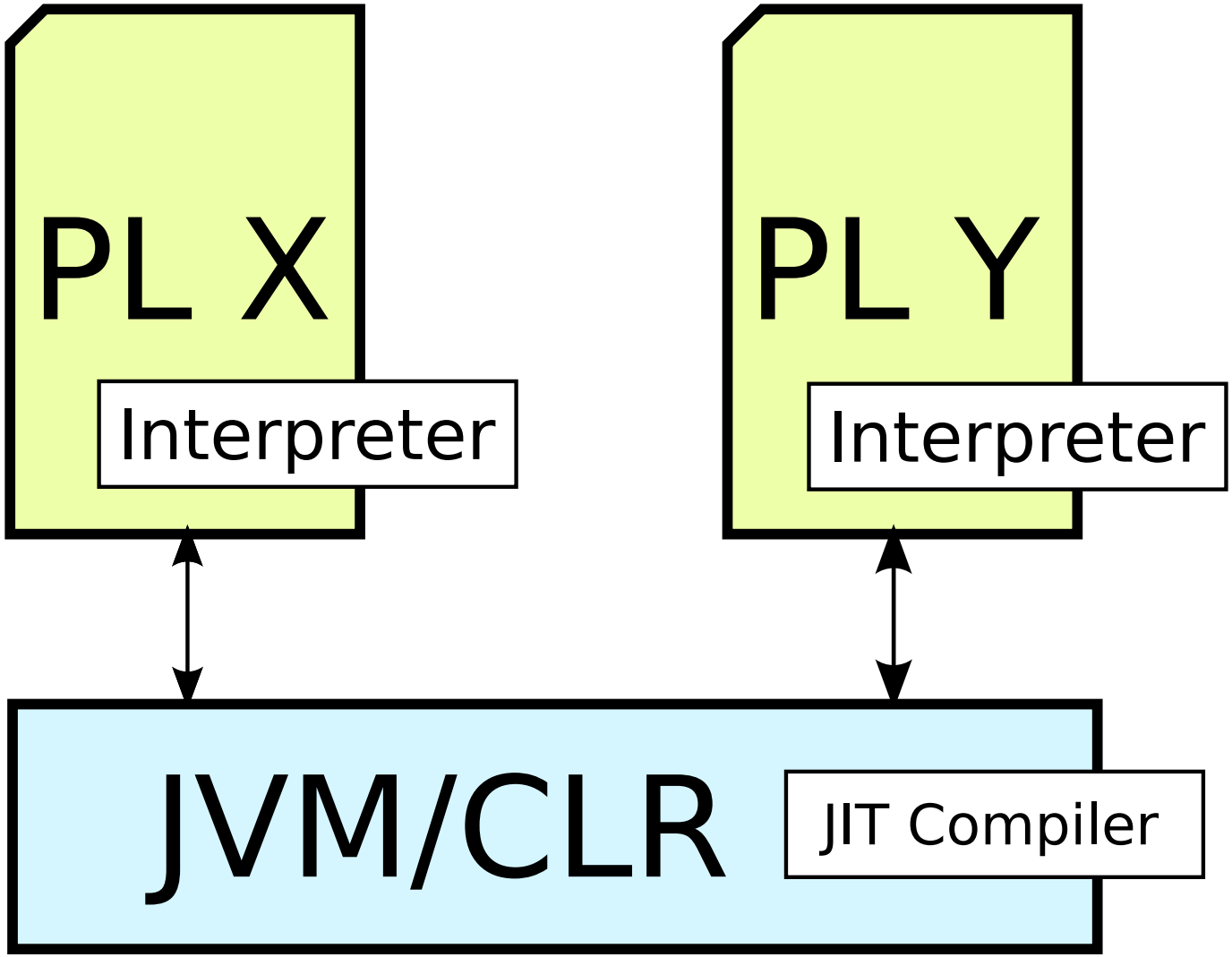
Runtime composition



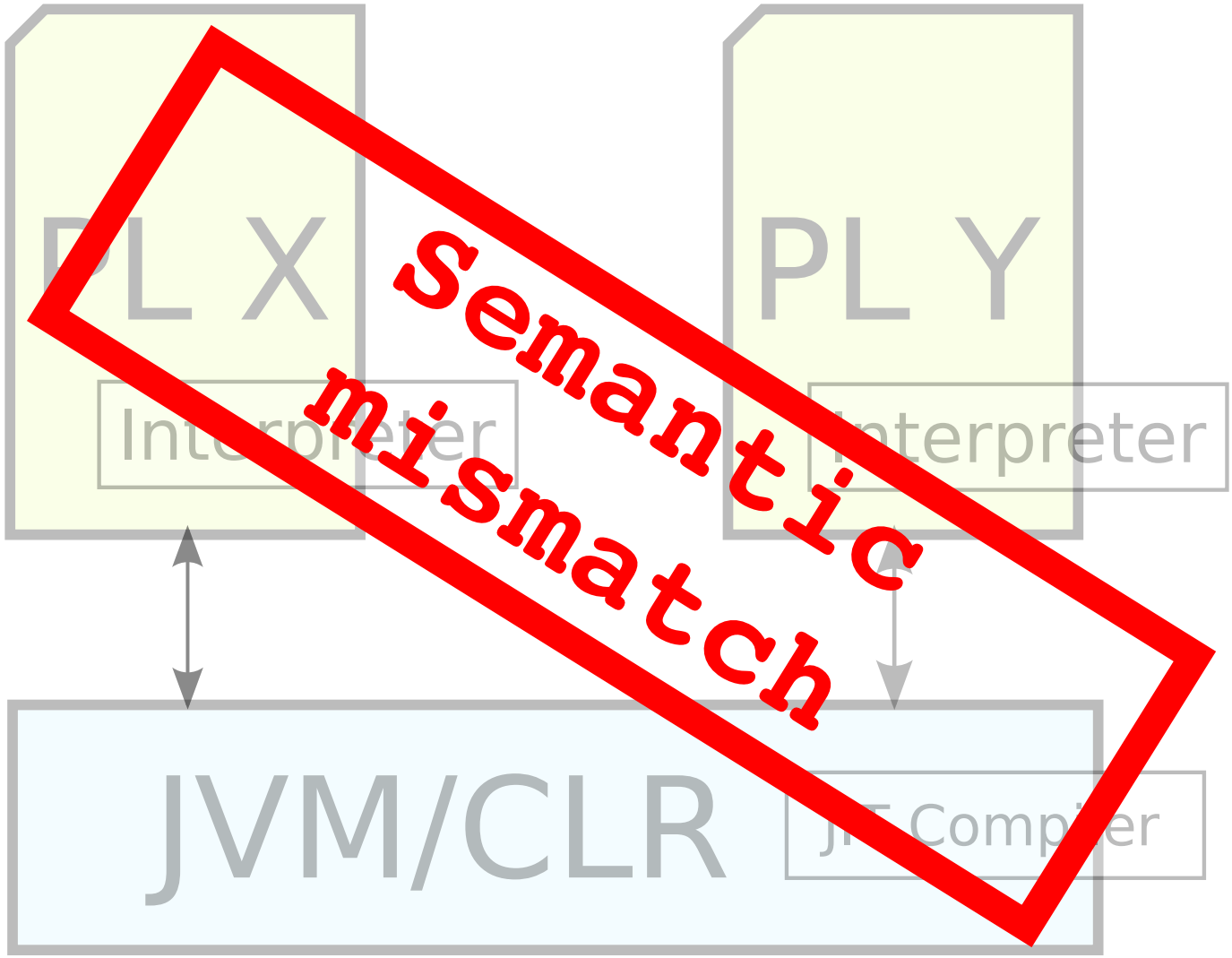
Runtime composition



Runtime composition

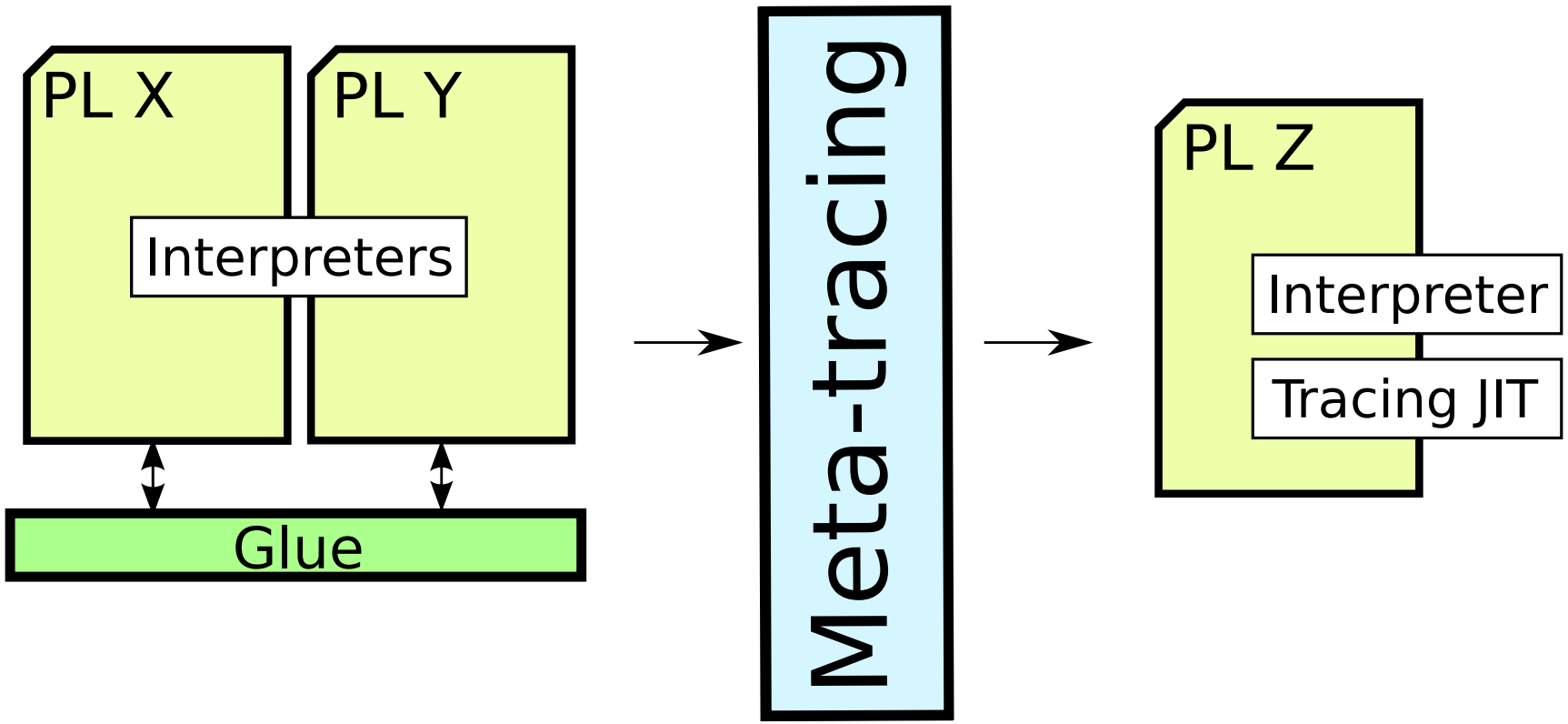


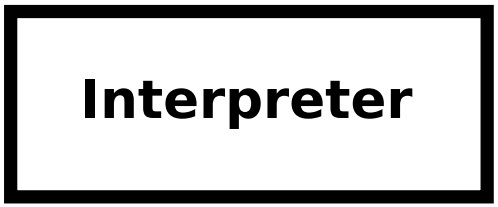
Runtime composition



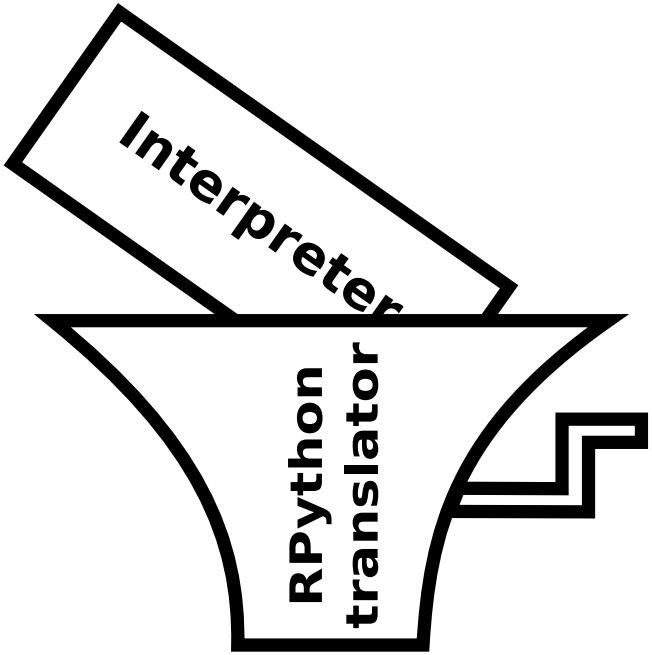
Runtime composition

Runtime composition

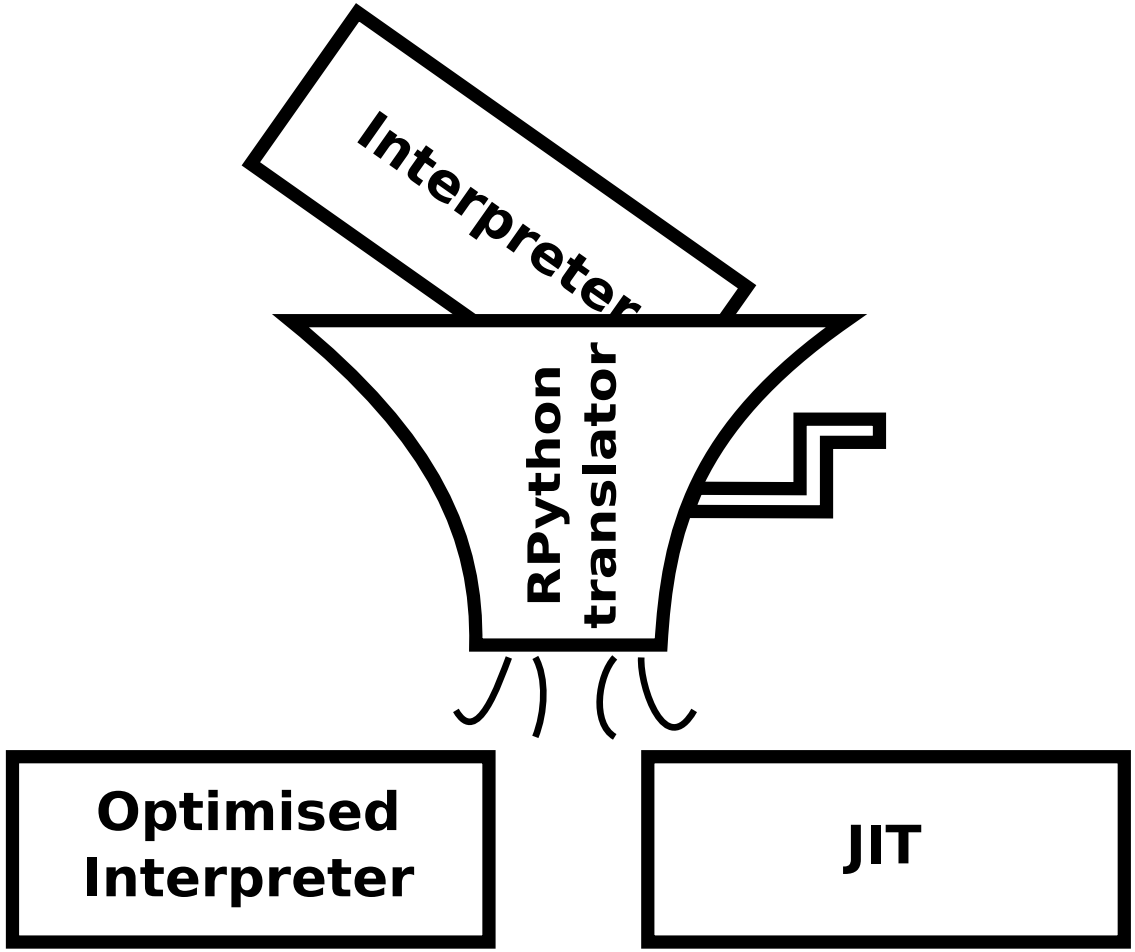



Interpreter

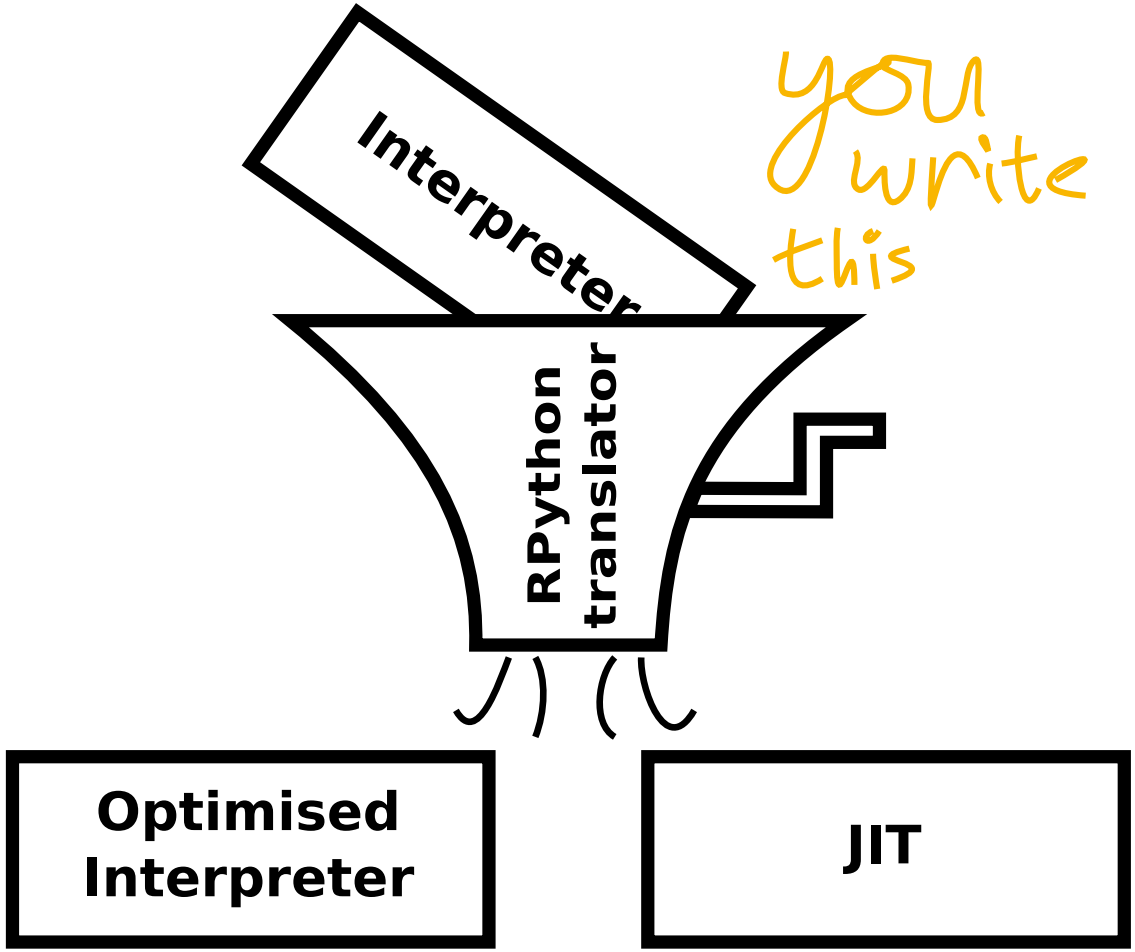
Meta-tracing translation with RPython



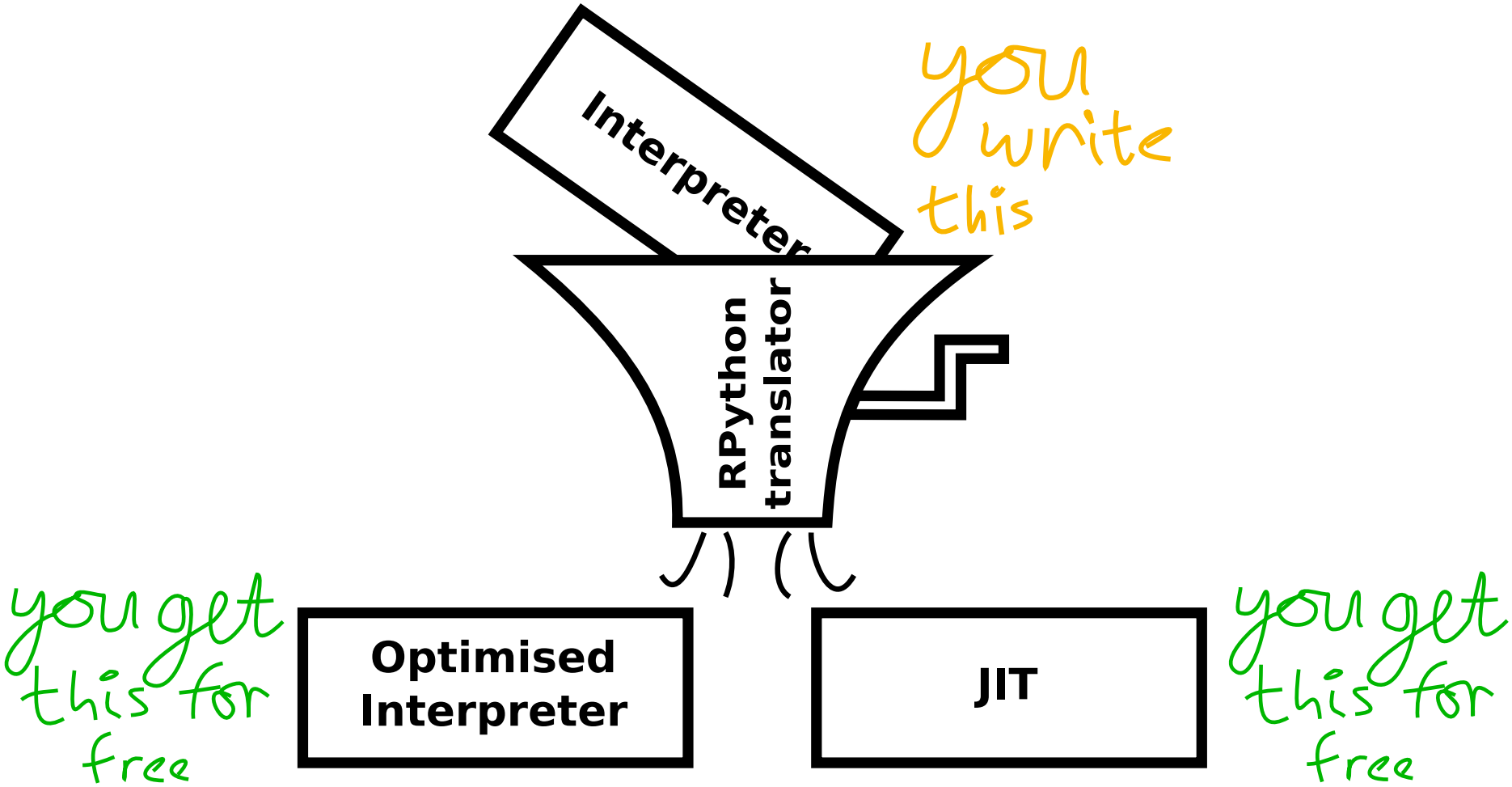
Meta-tracing translation with RPython



Meta-tracing translation with RPython



Meta-tracing translation with RPython



Unipycation demo

Benchmarking VM composition

Warning: draft numbers ahead

Absolute timing comparison

VM	Benchmark	*Python*		*Prolog*		*Python → Prolog*	
CPython-SWI	SmallFunc	0.125s	±0.006	0.257s	±0.001	28.893s	±0.175
	Loop1Arg0Result	2.924s	±0.215	7.352s	±0.037	9.310s	±0.065
	Loop1Arg1Result	4.184s	±0.028	18.890s	±0.082	20.865s	±0.050
	NondetLoop1Arg1Result	7.531s	±0.065	18.643s	±0.159	667.682s	±5.594
	TermConstruction	264.415s	±1.815	48.819s	±0.208	2185.150s	±14.251
	Lists	9.374s	±0.046	25.148s	±0.182	2207.304s	±12.344
Unipycation	SmallFunc	0.001s	±0.000	0.006s	±0.001	0.001s	±0.000
	Loop1Arg0Result	0.085s	±0.000	0.086s	±0.000	0.087s	±0.000
	Loop1Arg1Result	0.112s	±0.000	0.114s	±0.000	0.115s	±0.000
	NondetLoop1Arg1Result	0.500s	±0.002	0.548s	±0.064	2.674s	±0.010
	TermConstruction	6.053s	±0.218	2.444s	±0.002	36.069s	±0.171
	Lists	0.845s	±0.002	1.416s	±0.003	5.056s	±0.026
Jython-tuProlog	SmallFunc	0.088s	±0.002	3.050s	±0.036	52.294s	±0.371
	Loop1Arg0Result	1.078s	±0.007	206.590s	±2.884	199.963s	±1.784
	Loop1Arg1Result	2.145s	±0.175	293.311s	±4.270	294.781s	±4.746
	NondetLoop1Arg1Result	7.939s	±0.341	timeout		timeout	
	TermConstruction	timeout		timeout		timeout	
	Lists	timeout		timeout		timeout	

Relative timing comparison

VM	Benchmark	$\frac{Python \to Prolog}{Python}$		$\frac{Python \to Prolog}{Prolog}$		$\frac{Python \to Prolog}{\text{Unipycation}}$	
CPython-SWI	SmallFunc	231.770×	±10.154	112.567×	±0.934	27821.079×	±1896.725
	Loop1Arg0Result	3.184×	±0.232	1.266×	±0.011	107.591×	±0.779
	Loop1Arg1Result	4.987×	±0.039	1.105×	±0.006	181.899×	±0.444
	NondetLoop1Arg1Result	88.654×	±1.026	35.814×	±0.389	249.737×	±2.244
	TermConstruction	8.264×	±0.081	44.760×	±0.348	60.583×	±0.487
	Lists	235.459×	±1.742	87.772×	±0.789	436.609×	±3.494
Unipycation	SmallFunc	1.295×	±0.086	0.182×	±0.036	1.000×	
	Loop1Arg0Result	1.020×	±0.001	1.012×	±0.002	1.000×	
	Loop1Arg1Result	1.025×	±0.002	1.002×	±0.002	1.000×	
	NondetLoop1Arg1Result	5.349×	±0.035	4.879×	±0.631	1.000×	
	TermConstruction	5.959×	±0.224	14.756×	±0.069	1.000×	
	Lists	5.982×	±0.034	3.569×	±0.019	1.000×	
Jython-tuProlog	SmallFunc	592.904×	±14.602	17.143×	±0.259	50354.204×	±3330.993
	Loop1Arg0Result	185.460×	±2.182	0.968×	±0.017	2310.844×	±21.996
	Loop1Arg1Result	137.427×	±11.805	1.005×	±0.022	2569.873×	±41.331
	NondetLoop1Arg1Result	timeout		timeout		timeout	
	TermConstruction	timeout		timeout		timeout	
	Lists	timeout		timeout		timeout	

PHP / Python bridge demo

Composed Richards vs. other VMs

Warning: even draftier numbers ahead!

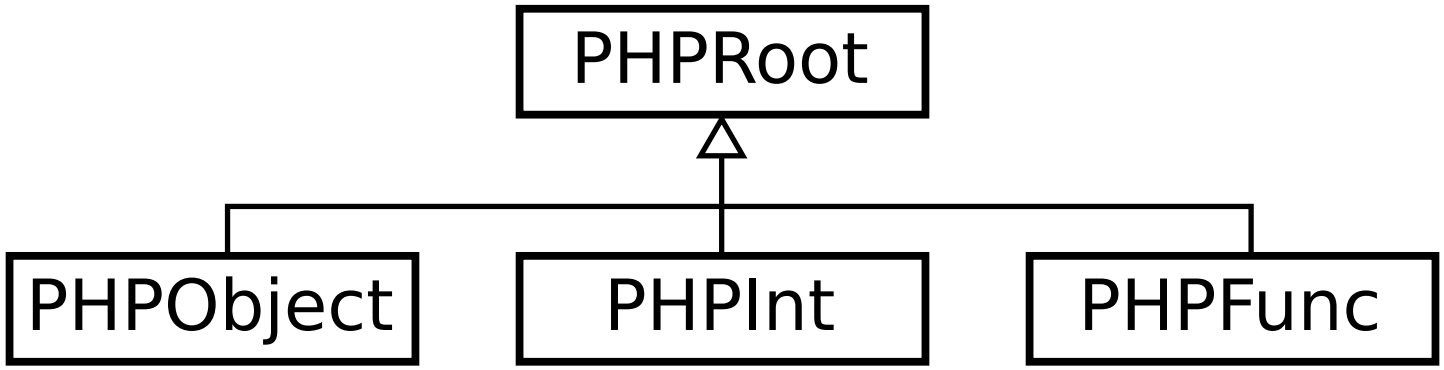
Composed Richards vs. other VMs

Type	VM	
Mono	PyPy 2.4.0	0.370 ± 0.000
	Hippy	0.553 ± 0.008
	Bridge	0.556 ± 0.006
	HHVM 3.2.0	5.353 ± 0.262
	ZEND 5.4.4	10.406 ± 0.106

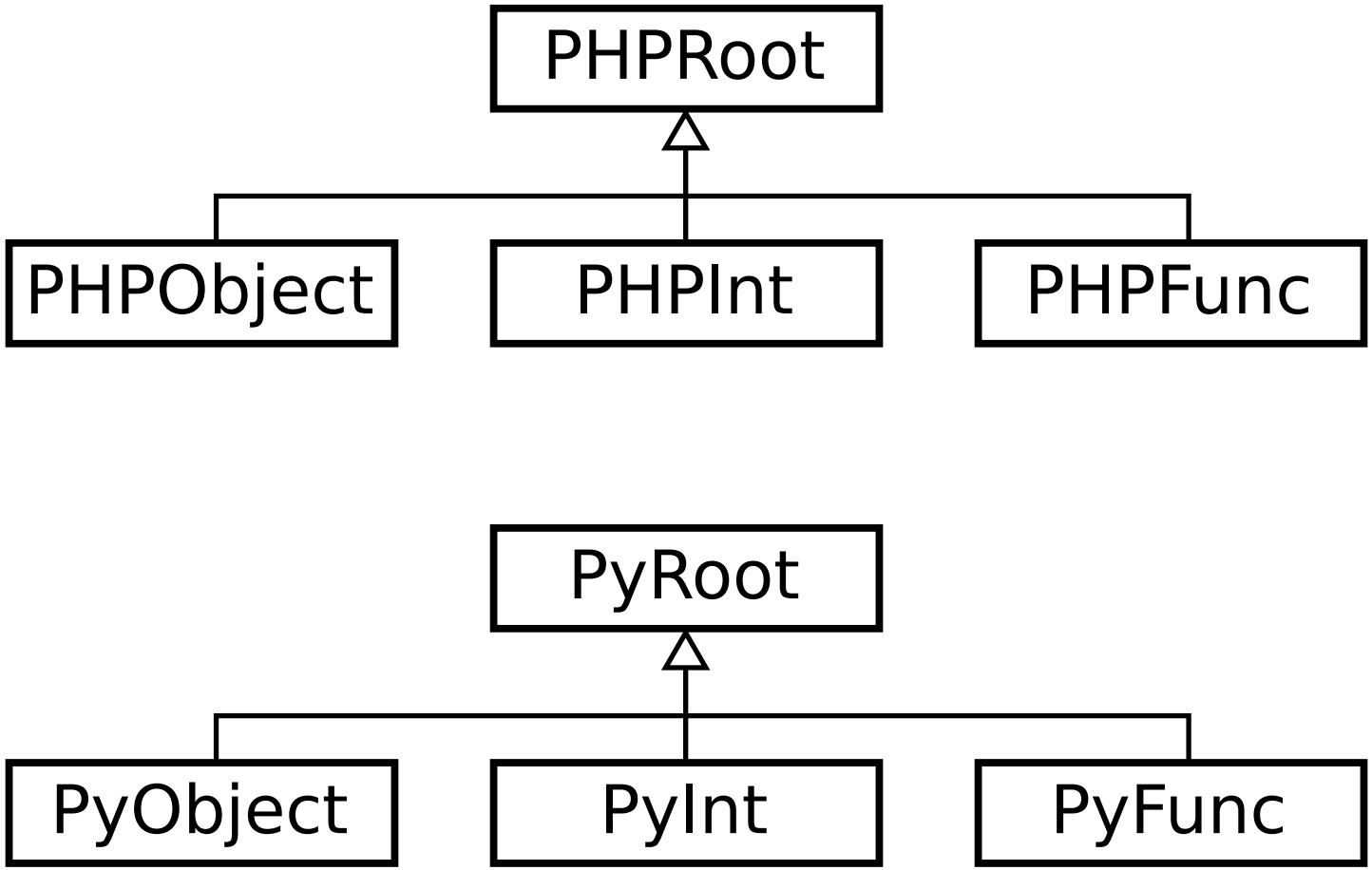
Composed Richards vs. other VMs

Type	VM	
Mono	PyPy 2.4.0	0.370 ± 0.000
	Hippy	0.553 ± 0.008
	Bridge	0.556 ± 0.006
	HHVM 3.2.0	5.353 ± 0.262
	ZEND 5.4.4	10.406 ± 0.105
Composed	Bridge	0.936 ± 0.038

Datatype conversion



Datatype conversion



Datatype conversion: primitive types

PHP

Python

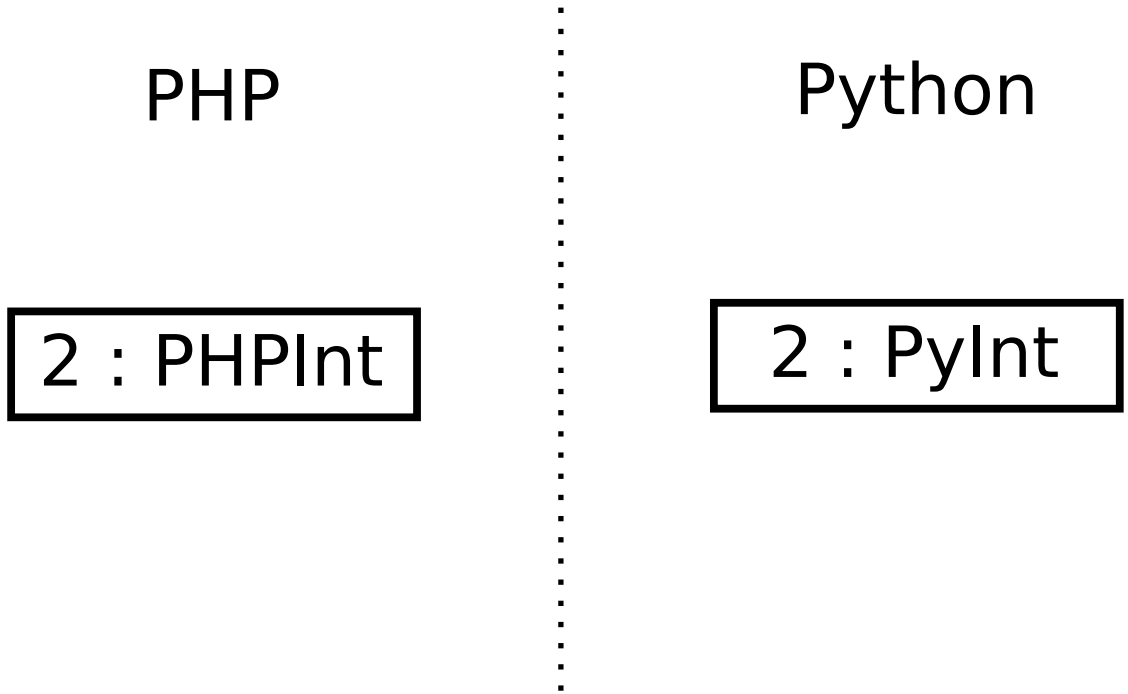
Datatype conversion: primitive types

PHP

Python

2 : PHPInt

Datatype conversion: primitive types

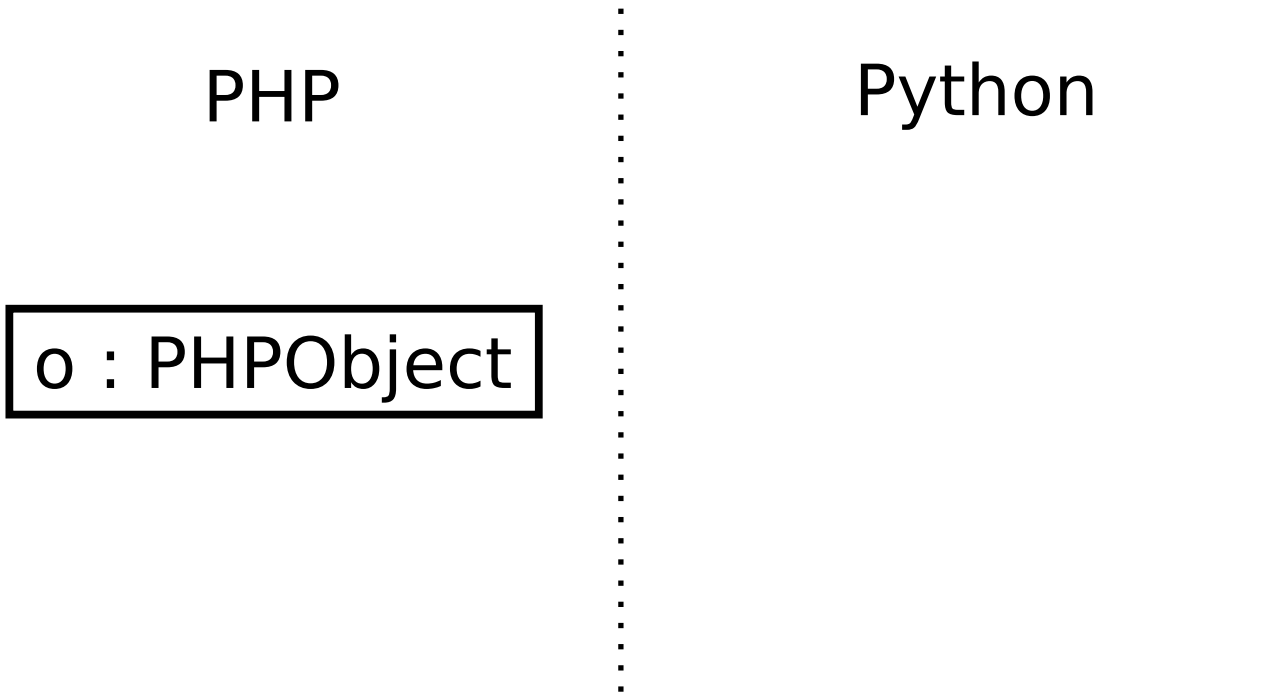


Datatype conversion: user types

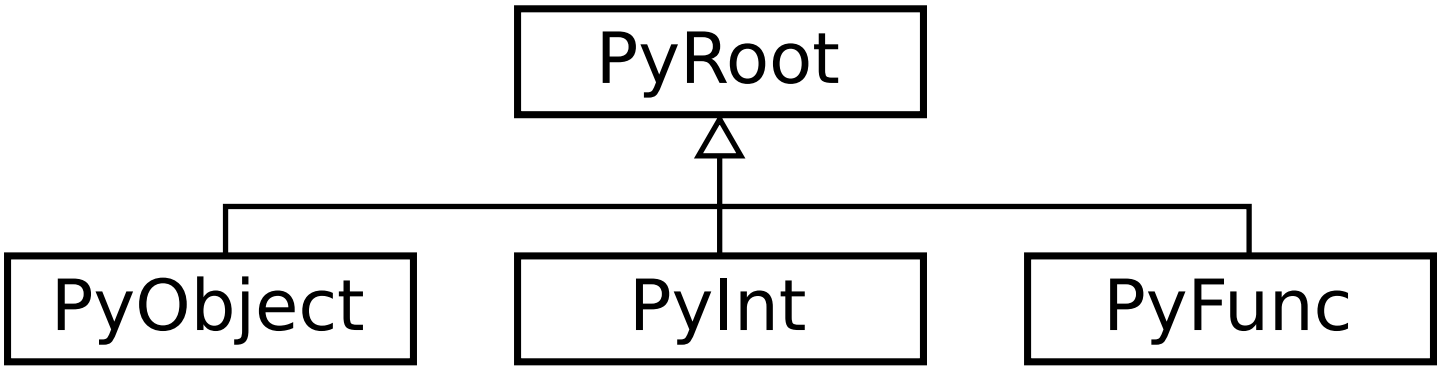
PHP

Python

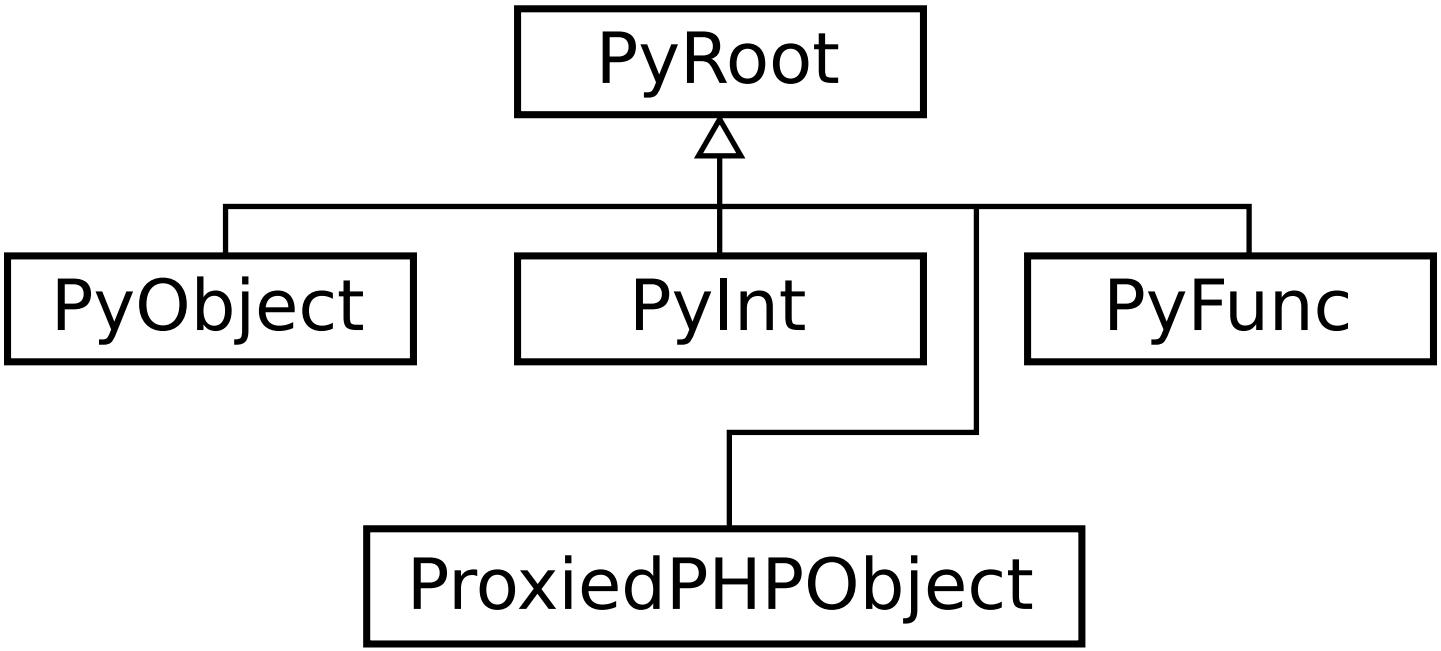
Datatype conversion: user types



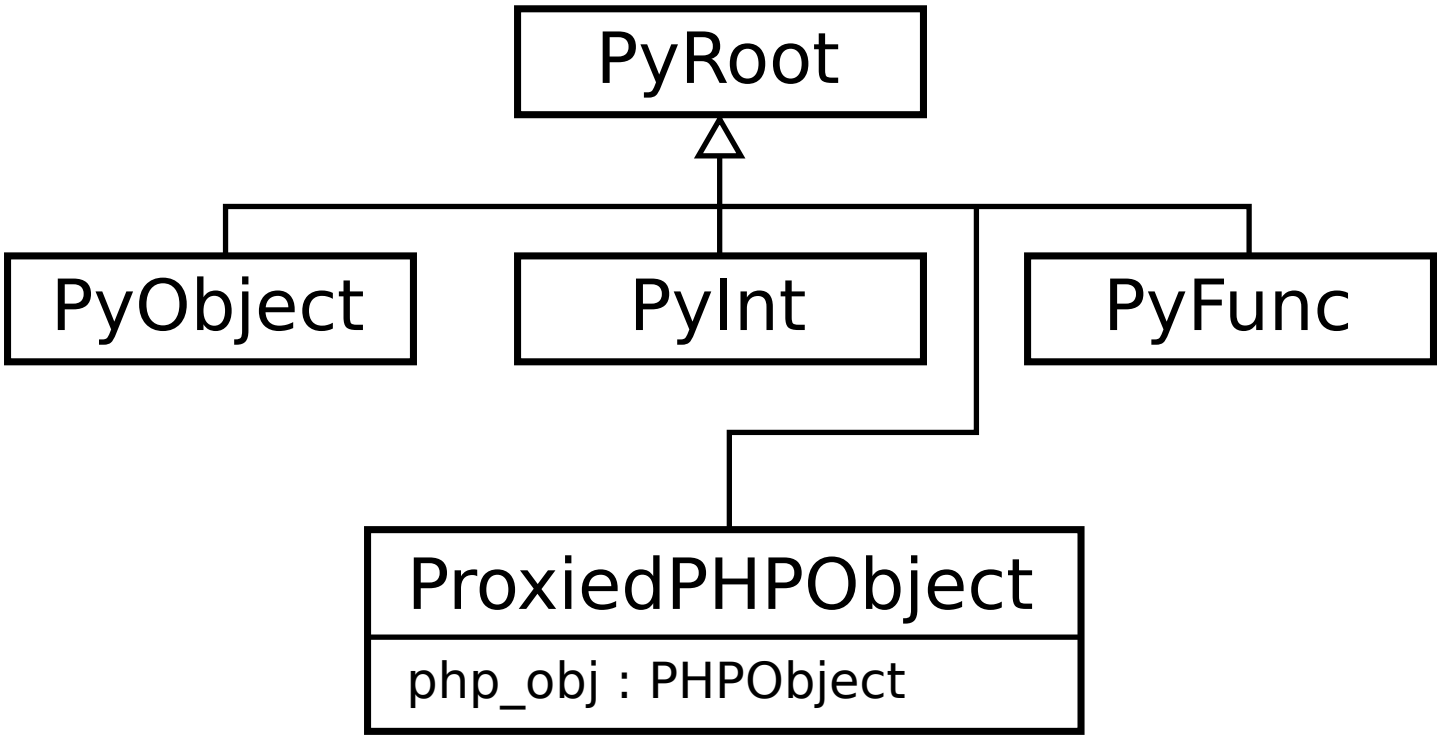
Datatype conversion: user types



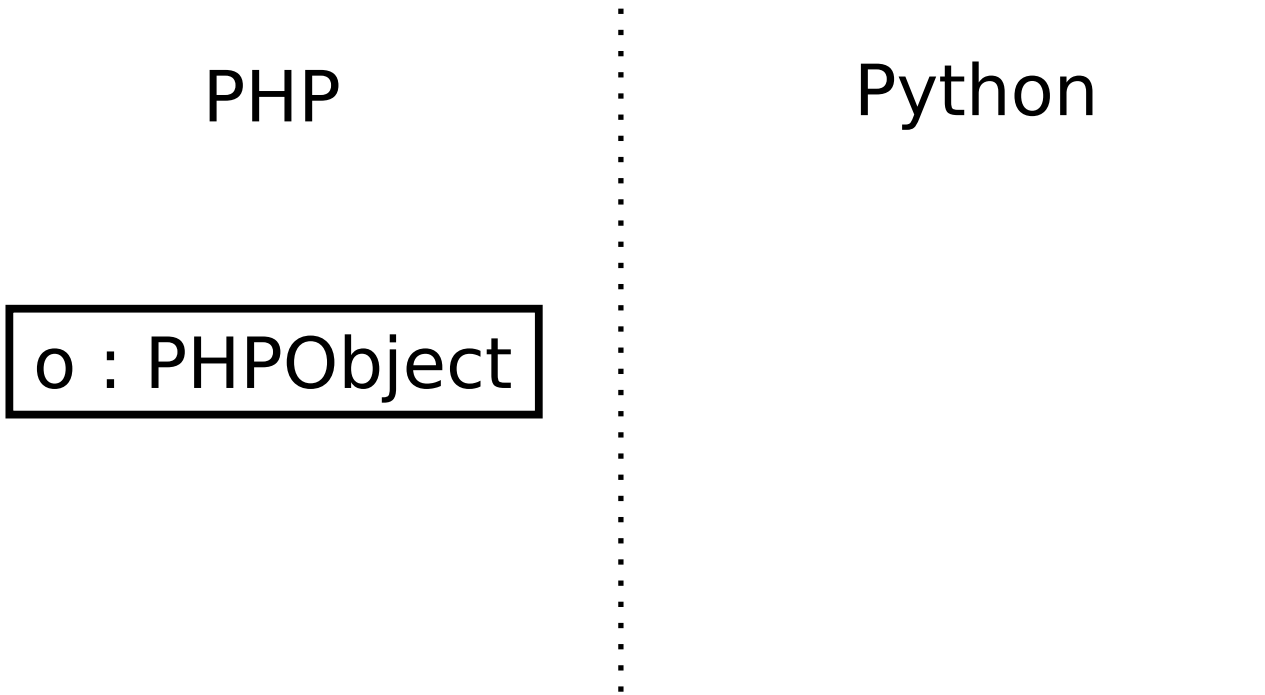
Datatype conversion: user types



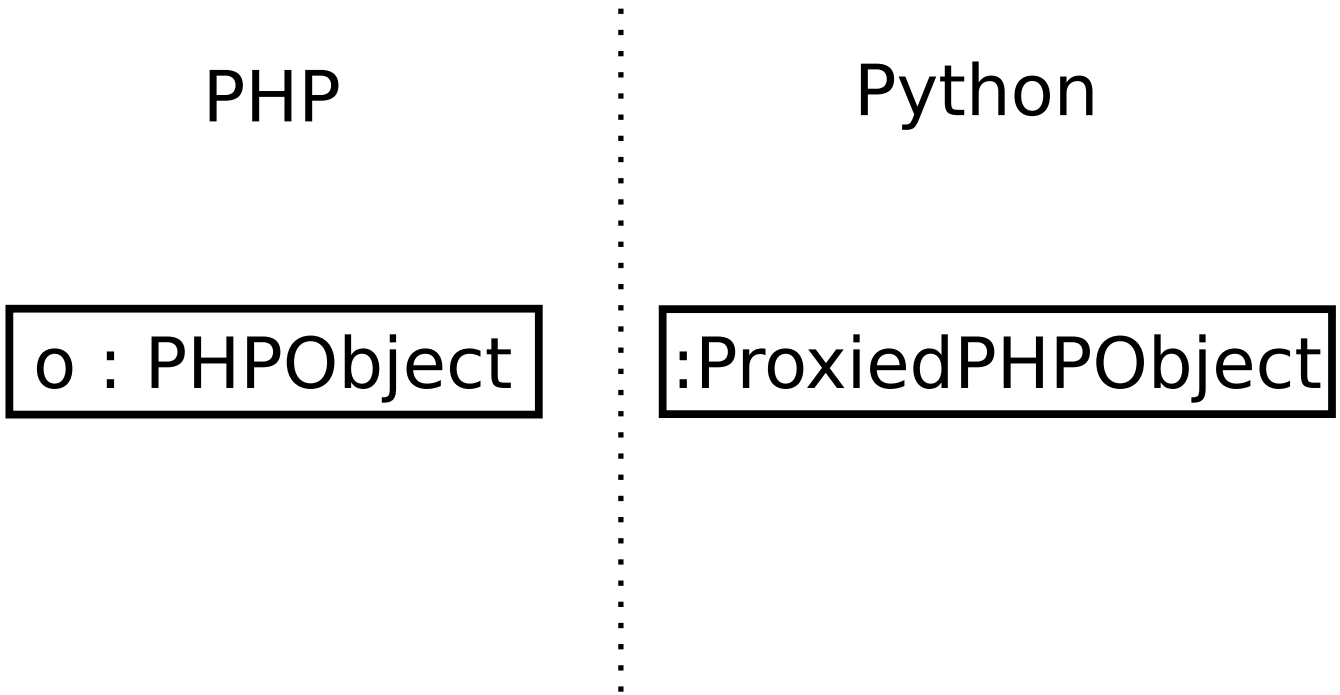
Datatype conversion: user types



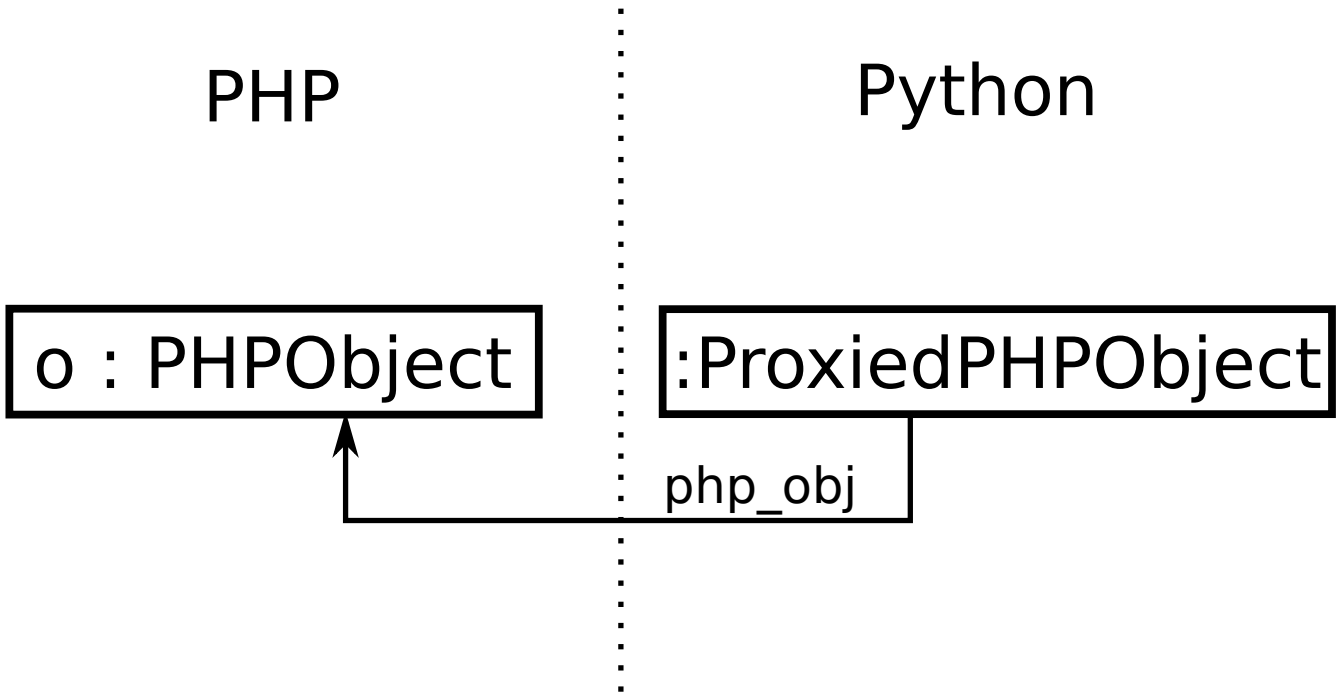
Datatype conversion: user types



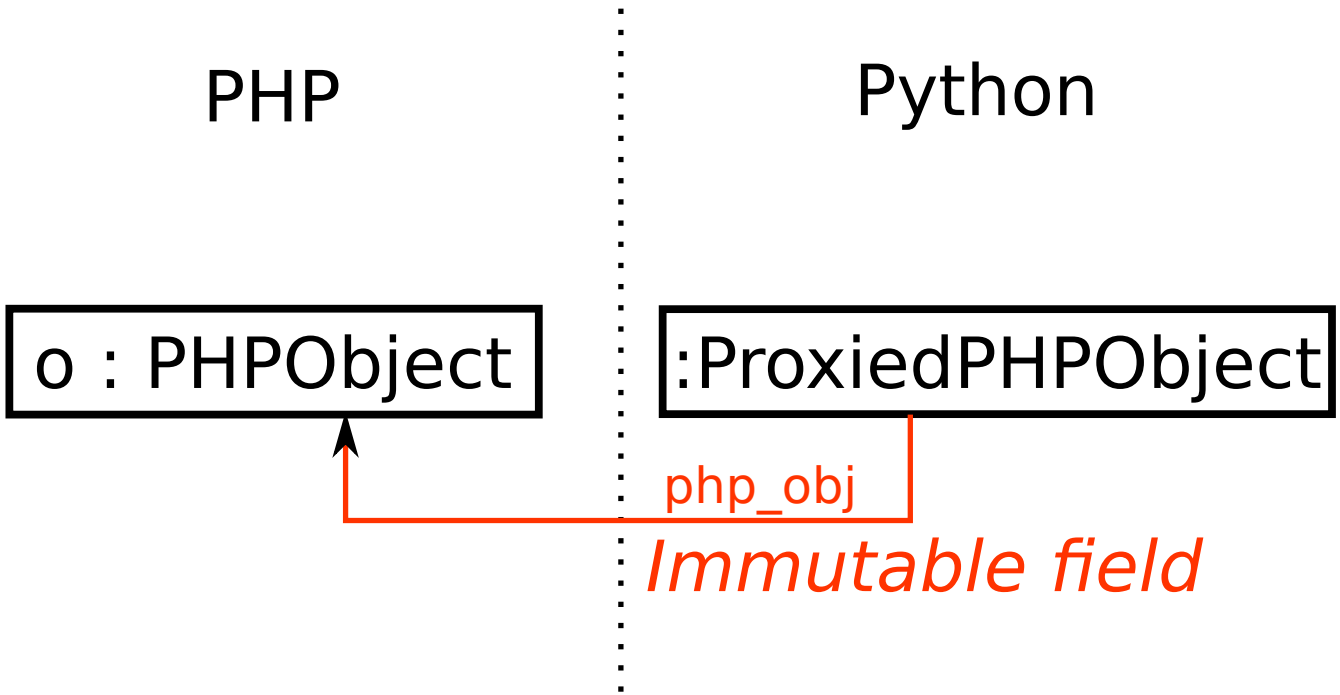
Datatype conversion: user types



Datatype conversion: user types



Datatype conversion: user types



Some thoughts

- Critical: single meta-language (e.g. RPython / Truffle).

Some thoughts

- Critical: single meta-language (e.g. RPython / Truffle).
- Simplicity: good performance, yet understandable.

Some thoughts

- Critical: single meta-language (e.g. RPython / Truffle).
- Simplicity: good performance, yet understandable.
- Immutable wrappers give near-native performance.

Some thoughts

- Critical: single meta-language (e.g. RPython / Truffle).
- Simplicity: good performance, yet understandable.
- Immutable wrappers give near-native performance.
- **Whole new world of challenges for language designers & formalisers.**

What can we use this for?

What can we use this for?

First-class languages

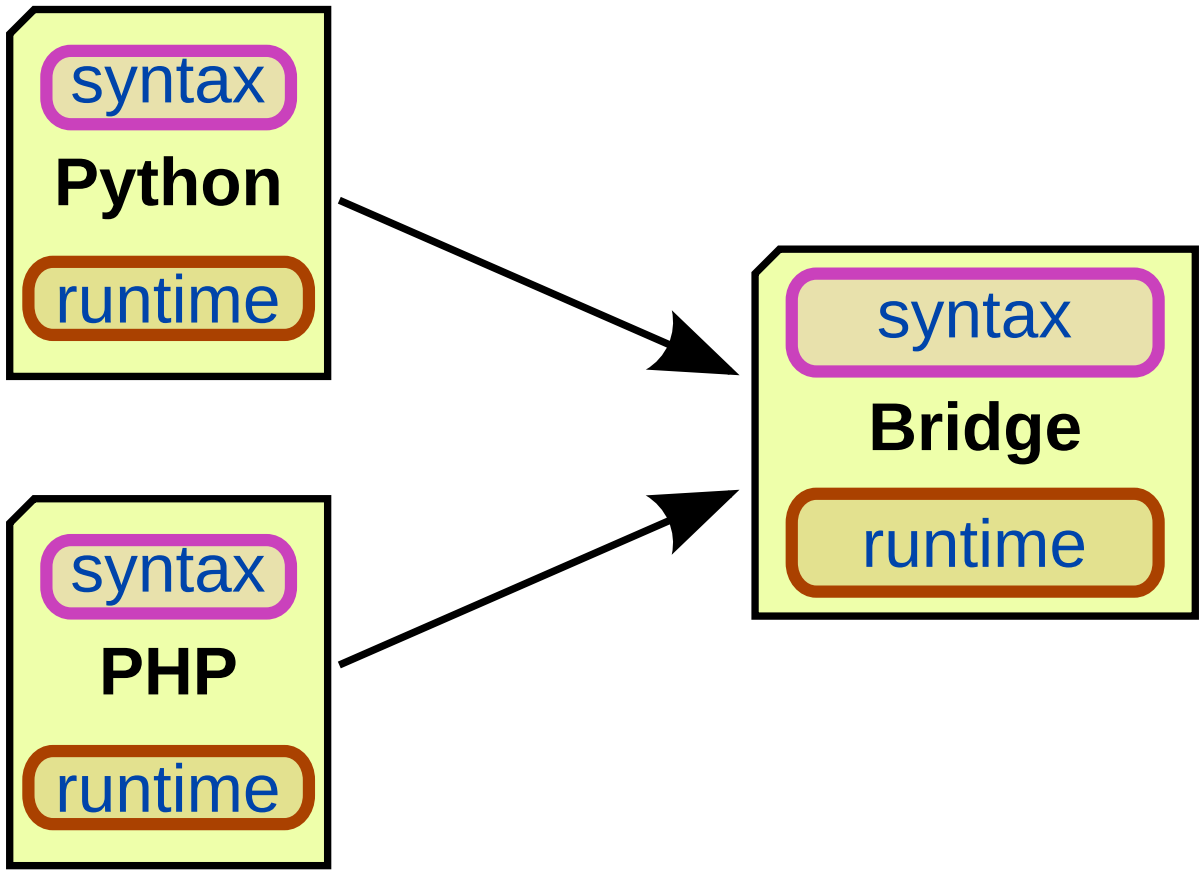
What can we use this for?

First-class languages

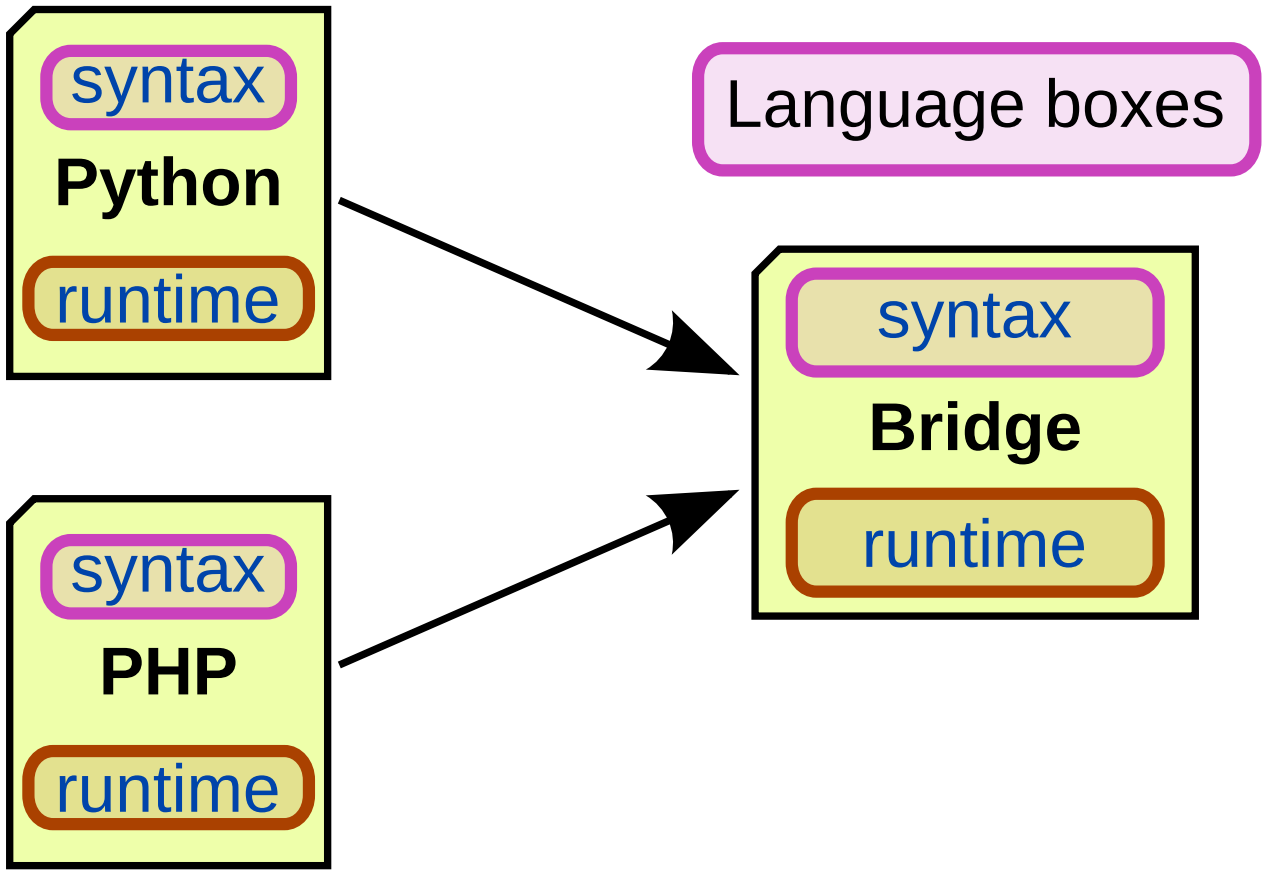
Language migration

Summary

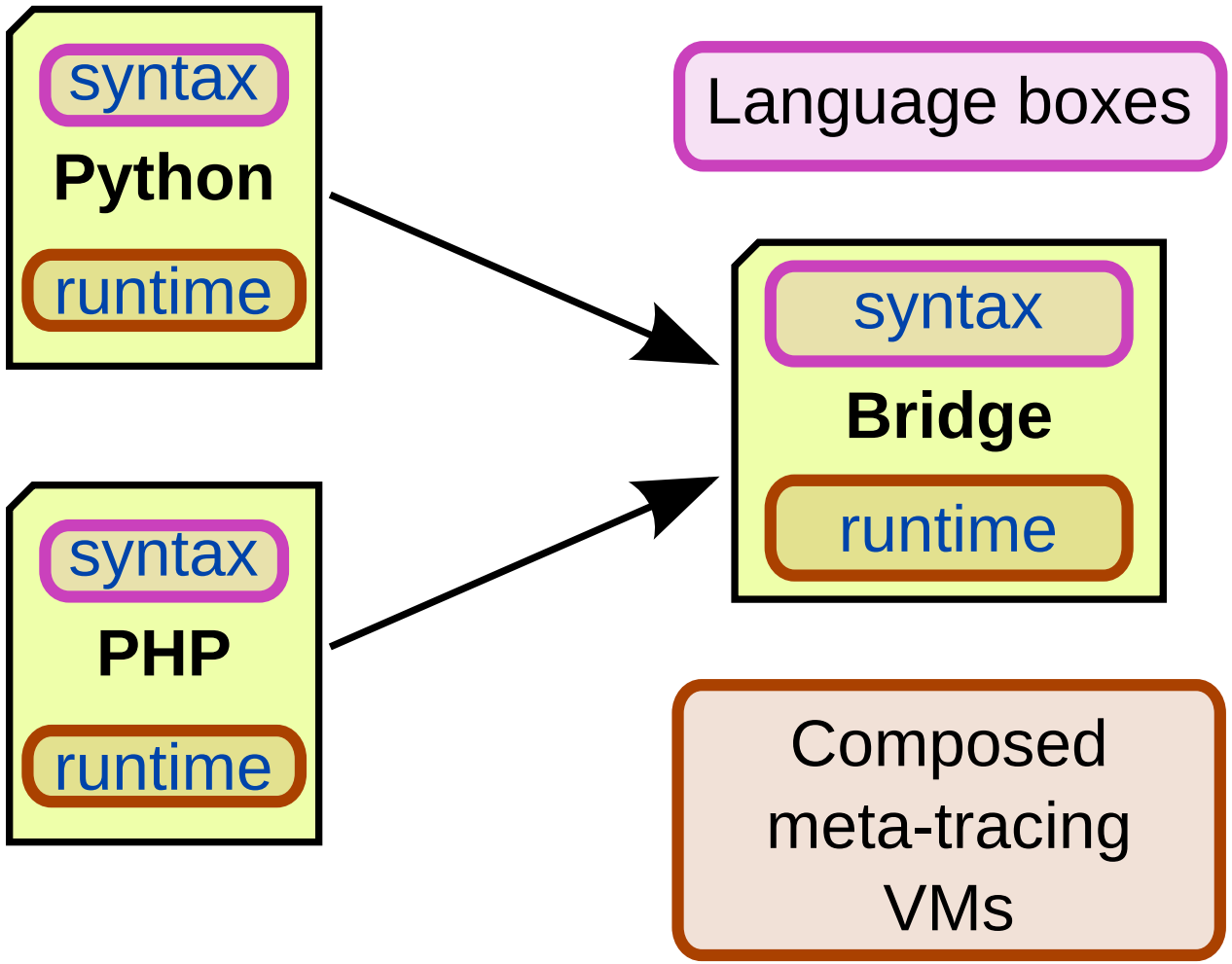
Summary


syntax
Python
runtime
syntax
PHP
runtime
syntax
Bridge
runtime

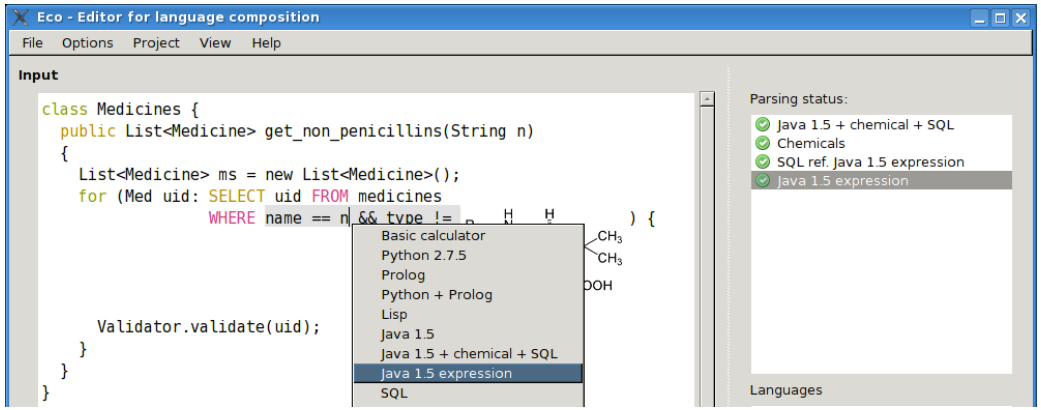
Summary

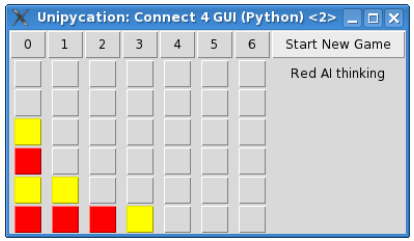

syntax
Python
runtime
Language boxes
syntax
Bridge
runtime
syntax
PHP
runtime

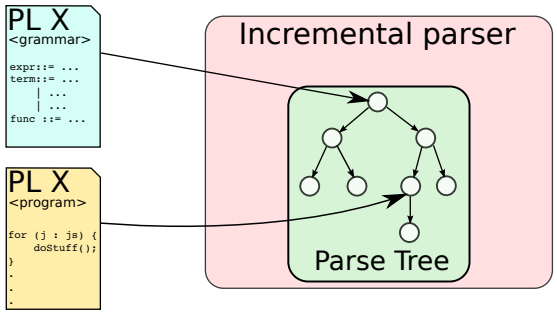
Summary

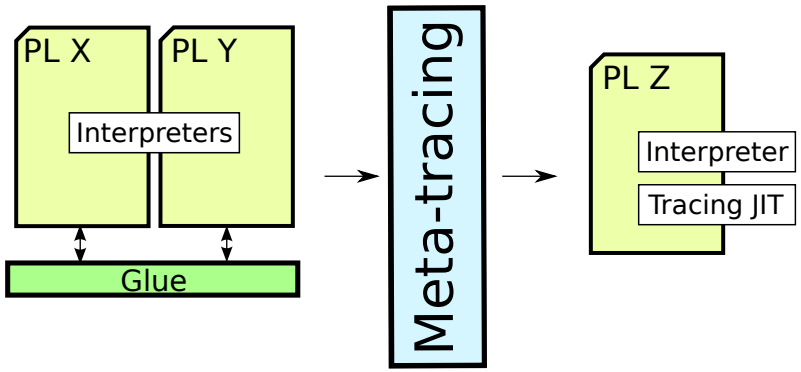

syntax
Python
runtime
syntax
PHP
runtime
Language boxes
syntax
Bridge
runtime
Composed
meta-tracing
VMs

Thanks for listening









http://soft-dev.org/