

# Unipycation

## A Case Study in Cross-Language Tracing



Edd  
Barrett



Carl Friedrich  
Bolz



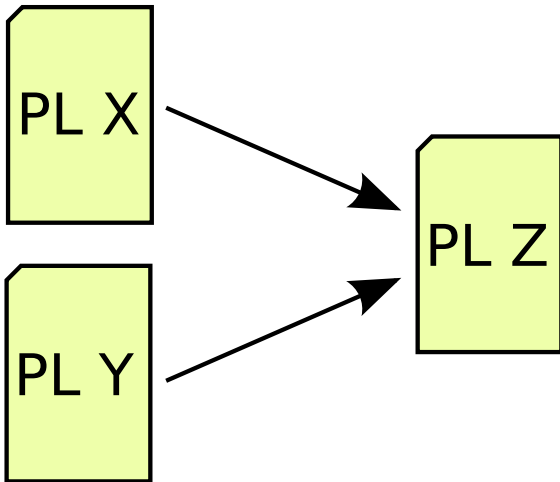
Laurence  
Tratt

**KING'S**  
*College*  
**LONDON**

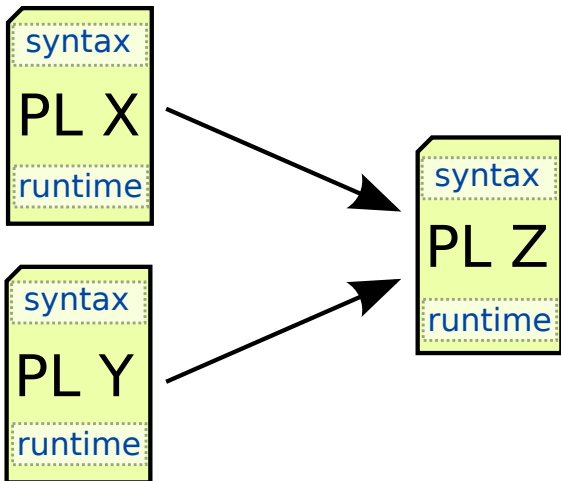
Software Development Team

2013-10-28

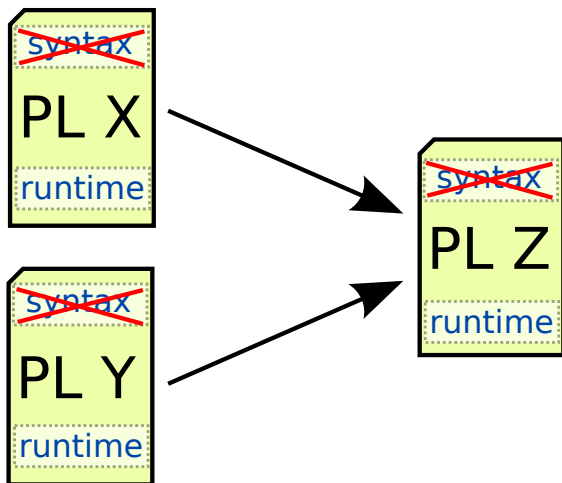
# Language Composition



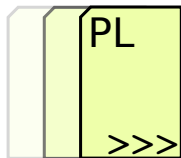
# Language Composition



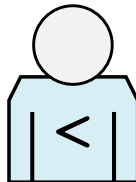
# Language Composition



## Requirements



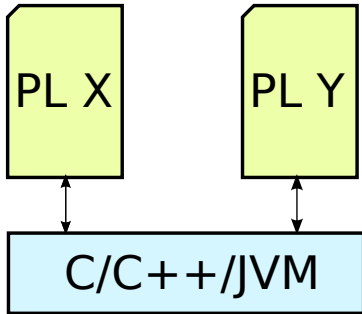
Fast



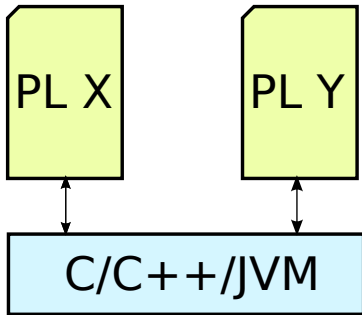
Less effort

Current runtime composition techniques?

# Traditional Approaches



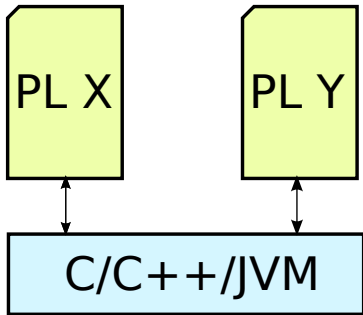
# Traditional Approaches



## C/C++

- Can build fast VMs.
- > Development effort.

# Traditional Approaches

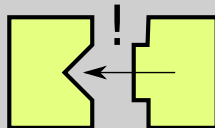


## C/C++

- Can build fast VMs.
- > Development effort.

## JVM

- < Development effort.
- “Semantic mismatch”.





# Hypothesis:

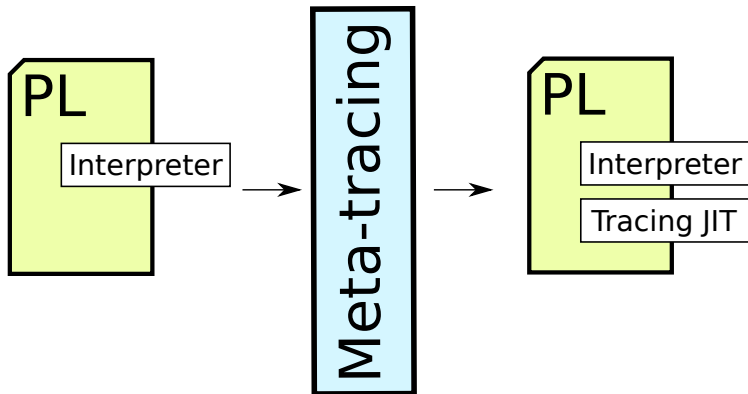
Meta-tracing may overcome these issues.

## Hypothesis:

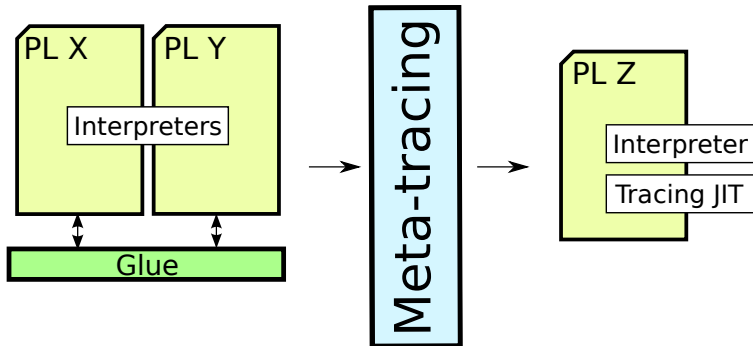
Meta-tracing may overcome these issues.

- Tracer for free.
  - Cross-language tracing.
- Little engineering effort.

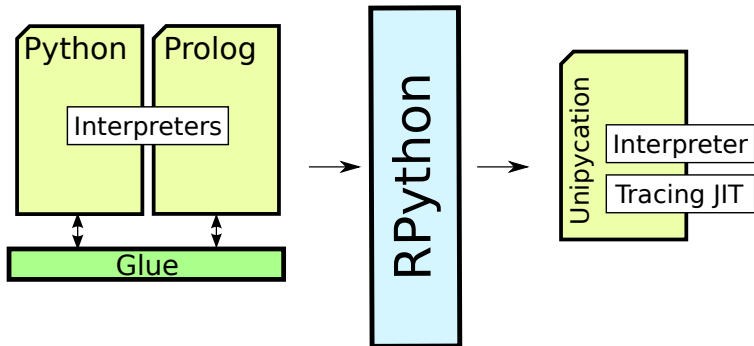
# Meta-tracing



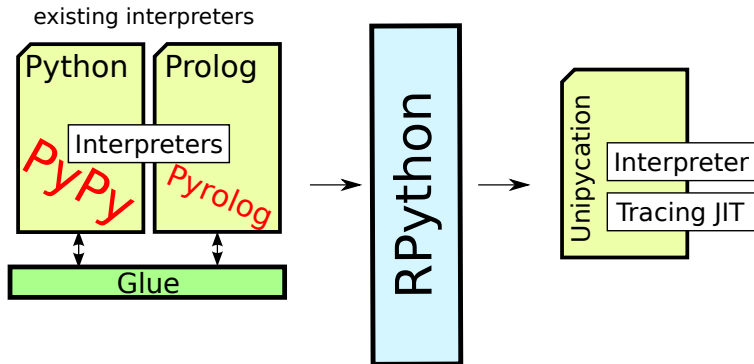
# Meta-tracing



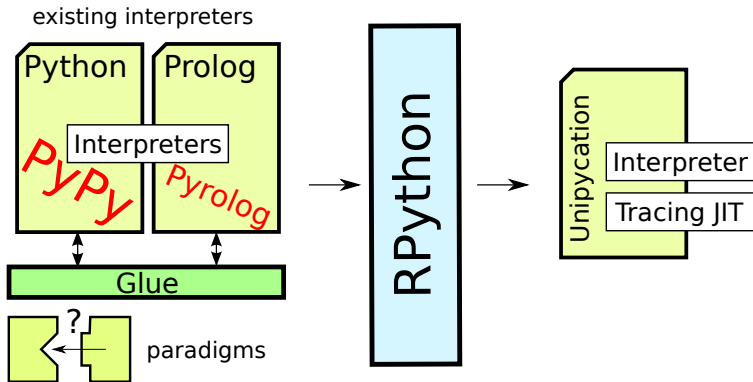
# Unipycation



# Unipycation



# Unipycation



# Design Considerations

type conversion

database/theory

partial lists

cross language calls

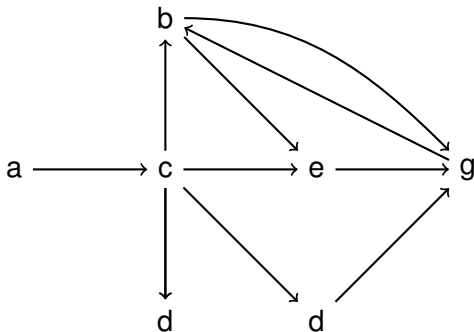
user expectations

unbound variables

non-determinism

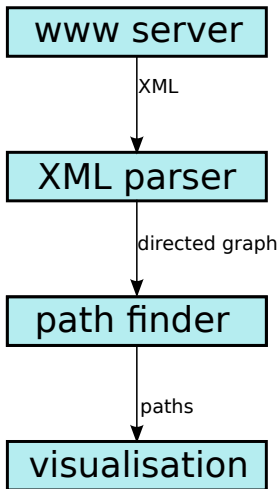


## Example: Underground/Metro System

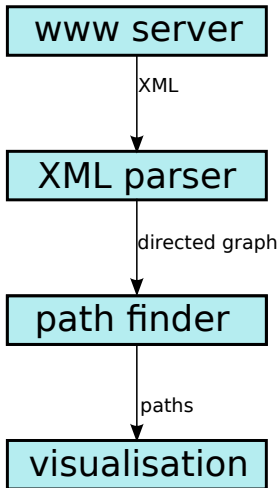


Where can I get to from 'b' via at most 4 nodes and how?

# Example: Underground/Metro System

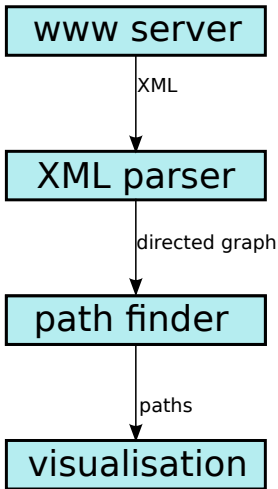


# Example: Underground/Metro System

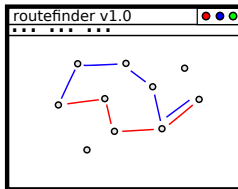


```
<edge src="a" dest="b" />  
<edge src="a" dest="c" />  
<edge src="b" dest="e" />
```

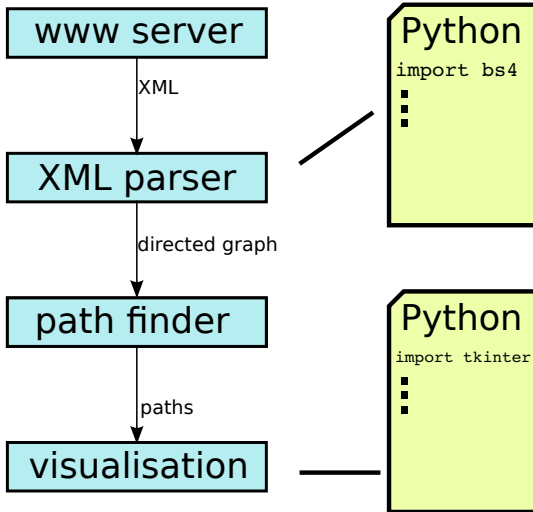
# Example: Underground/Metro System



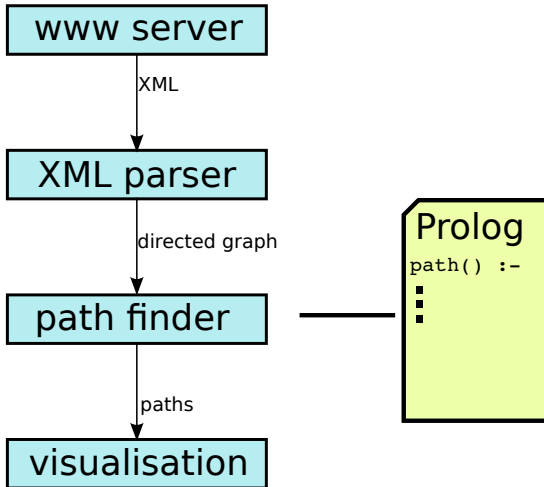
```
<edge src="a" dest="b" />  
<edge src="a" dest="c" />  
<edge src="b" dest="e" />
```



# Example: Underground/Metro System



# Example: Underground/Metro System



# Example: Underground/Metro System

```
import bs4, uni, tkinter

# ...code for parsing would be here

edges = { node : sucessors(node) for node in ... }
# e.g. edges = {
#     "a" : ["b", "c"],
#     "b" : ["e", "h"],
#     ...
# }

# prolog helper
def get_edges(src_node):
    return iter(edges[src_node])
```

# Example: Underground/Metro System

```
import bs4, uni, tkinter
```

```
# ...code for parsing would be here
```

```
edges = { node : sucessors(node) for node in ... }
```

```
# e.g. edges = {
```

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

```
#     "b" : ["e", "h"],
```

```
#     ...
```

```
# }
```

```
# prolog helper
```

```
def get_edges(src_node):
```

```
    return iter(edges[src_node])
```



# Example: Underground/Metro System

```
import bs4, uni, tkinter
```

```
# ...code for parsing would be here
```

```
edges = { node : sucessors(node) for node in ... }
```

```
# e.g. edges = {
```

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

```
#     "b" : ["e", "h"],
```

```
#     ...
```

```
# }
```

```
# prolog helper
```

```
def get_edges(src_node):
```

```
    return iter(edges[src_node])
```

# Example: Underground/Metro System

```
import bs4, uni, tkinter

# ...code for parsing would be here

edges = { node : sucessors(node) for node in ... }
# e.g. edges = {
#     "a" : ["b", "c"],
#     "b" : ["e", "h"],
#     ...
# }

# prolog helper
def get_edges(src_node):
    return iter(edges[src_node])
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")

paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")
```

```
paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")

paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")
```

```
paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")
```

```
paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")

paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```



# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")

paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")

paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")

paths = e.db.path.iter
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

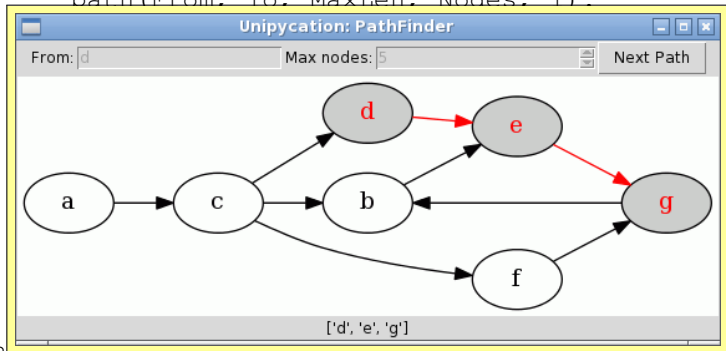
    path(Node, Node, _, [Node], _).
    path(From, To, MaxLen, [From | Ahead ], Len) :-
        Len < MaxLen, edge(From, Next),
        Len1 is Len + 1,
        path(Next, To, MaxLen, Ahead, Len1).

    edge(From, To) :- python:get_edges(From, To).
""")

paths = e.db.path
def get_edges(src_node):
    return iter(edges[src_node])
for (to, nodes) in paths("b", None, 4, None):
    # show path in GUI...
```

# Example: Underground/Metro System

```
e = uni.Engine("""  
path(From, To, MaxLen, Nodes) :-  
    path(From, To, MaxLen, Nodes, 1).
```



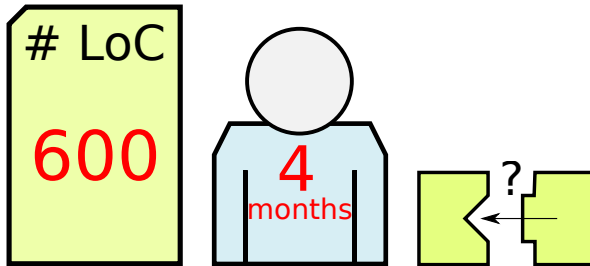
```
""")
```

```
path(From, To, MaxLen, Nodes, 1).
```

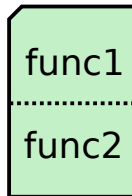
```
for (to, nodes) in paths("b", None, 4, None):  
    # show path in GUI...
```

How much effort was our composition?

# Development Effort



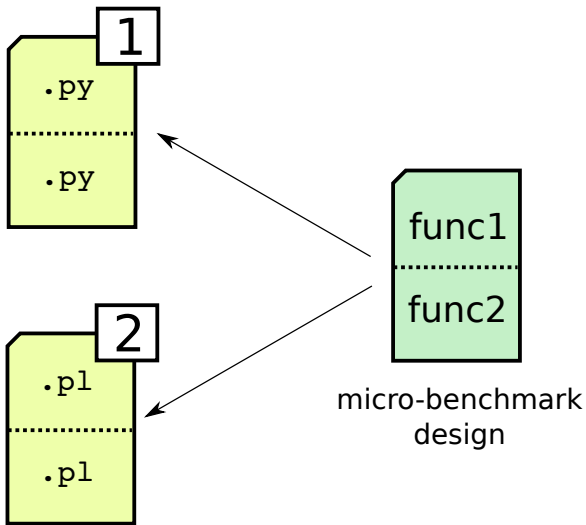
# Experimental Methodology



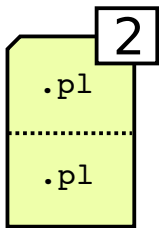
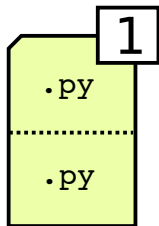
micro-benchmark  
design



# Experimental Methodology

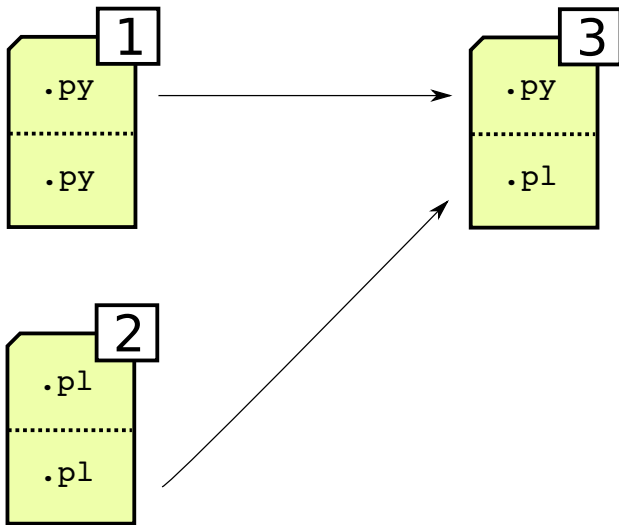


# Experimental Methodology

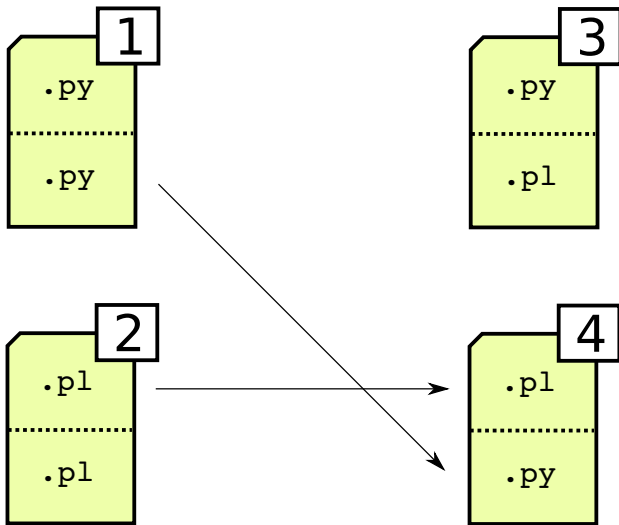


Baseline  
measurements

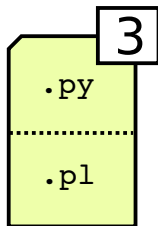
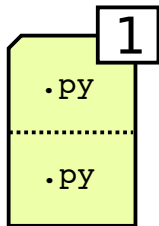
# Experimental Methodology



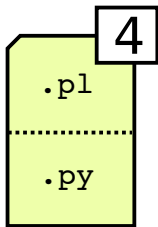
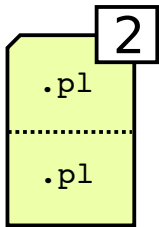
# Experimental Methodology



# Experimental Methodology

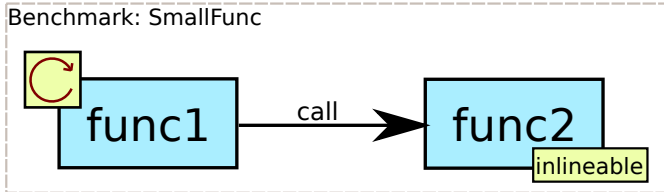


Performance  
comparison



7 micro-benchmarks, e.g.:

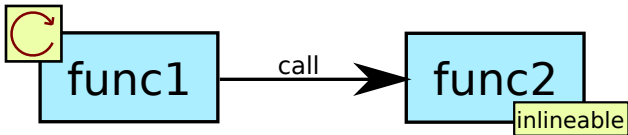
7 micro-benchmarks, e.g.:



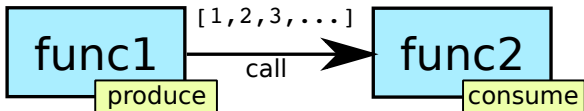
# Experimental Methodology

7 micro-benchmarks, e.g.:

Benchmark: SmallFunc

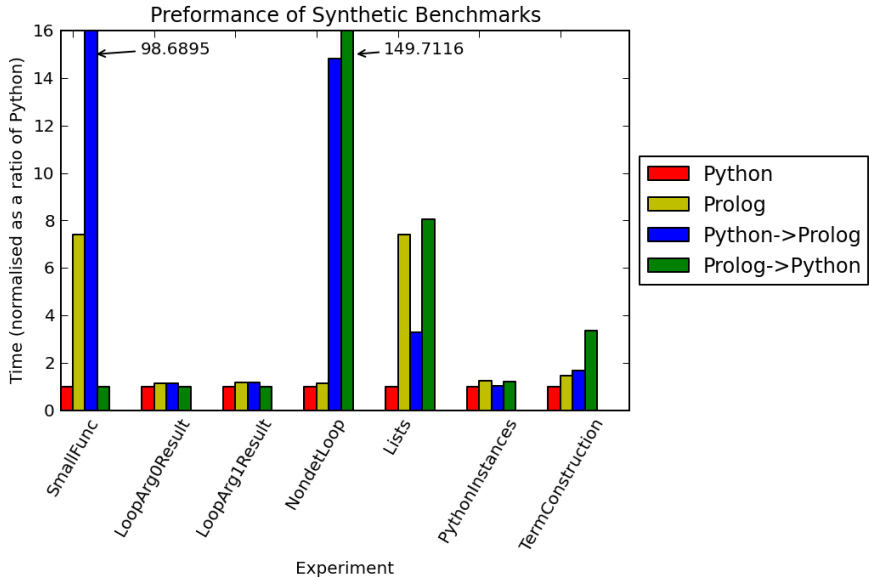


Benchmark: Lists



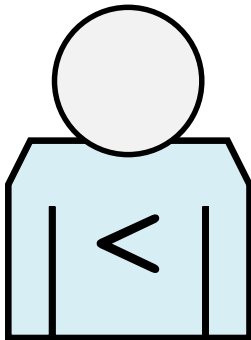


# Experimental Evaluation



# In Summary

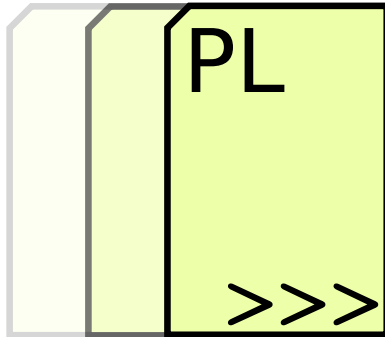
# In Summary



# In Summary



# In Summary

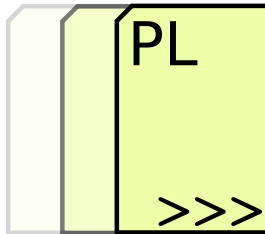


# In Summary



What is next?

# Future Work



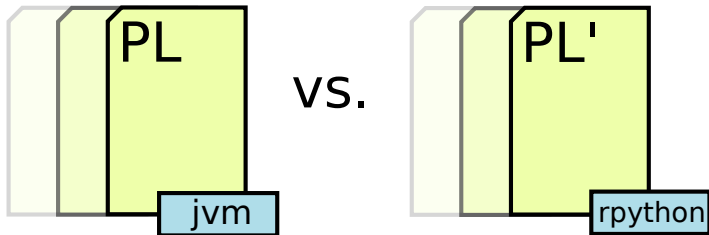
Faster



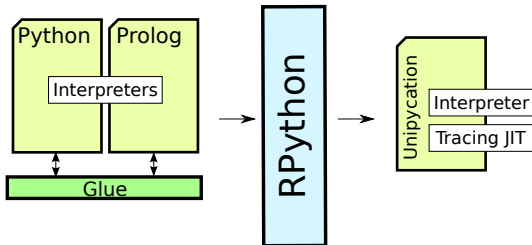
```
for i in path(X, Y, 3, [a, b])
```

## Syntax

# Future Work



# Demo/Question Time



<http://soft-dev.org>