Unipycation

A Case Study in Cross-Language Tracing

Edd Barrett

Carl Friedrich Bolz

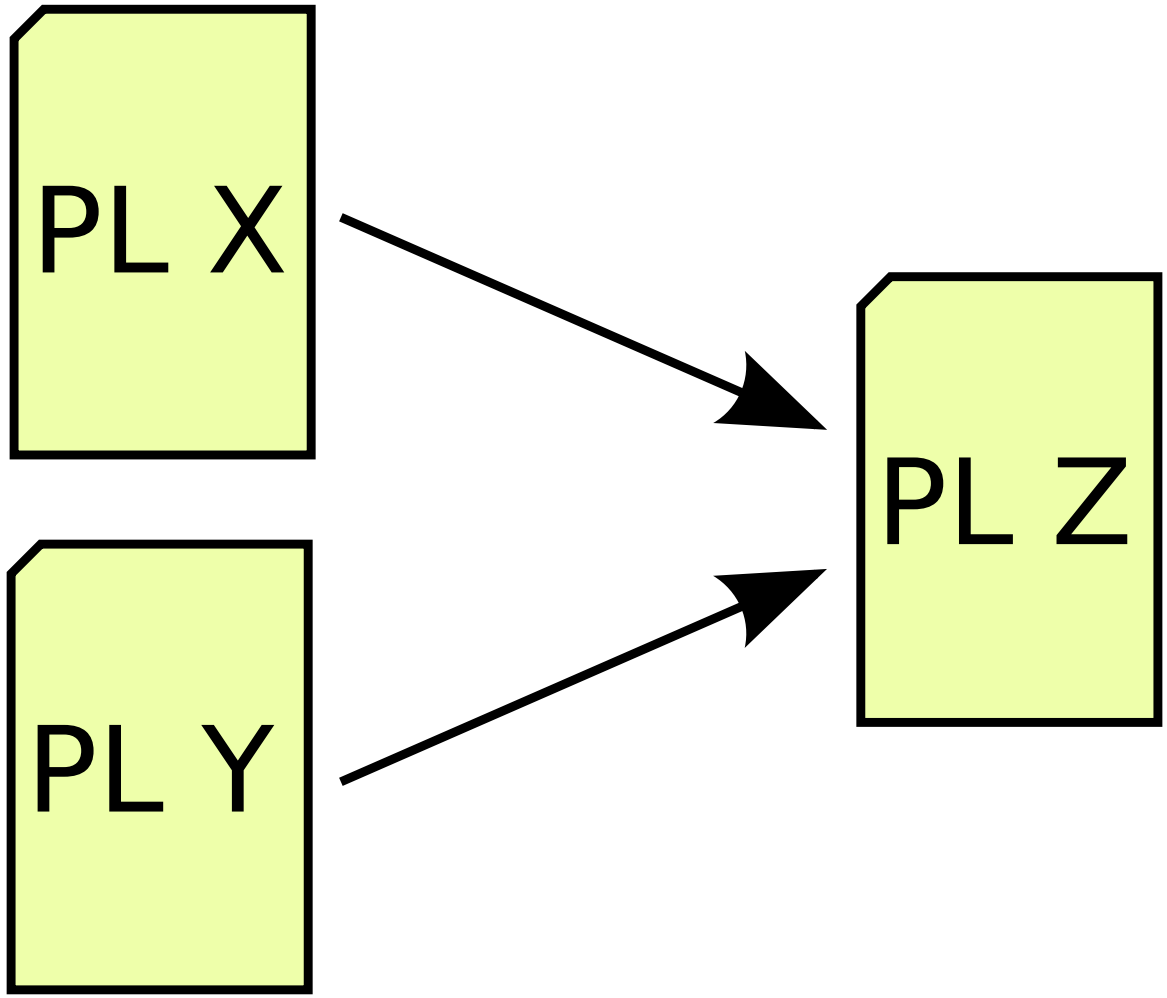
Laurence Tratt

King's College London

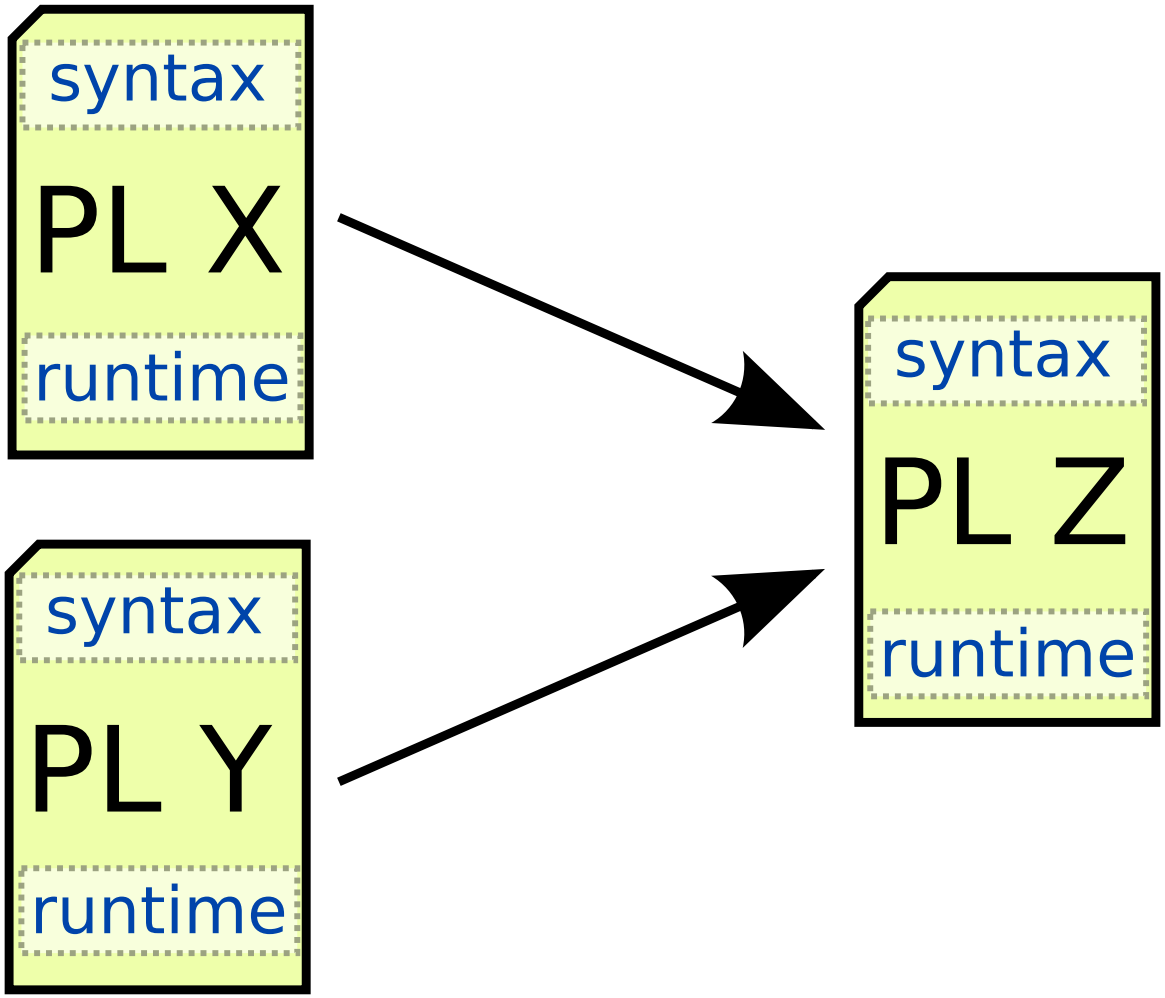
Software Development Team

2013-10-28

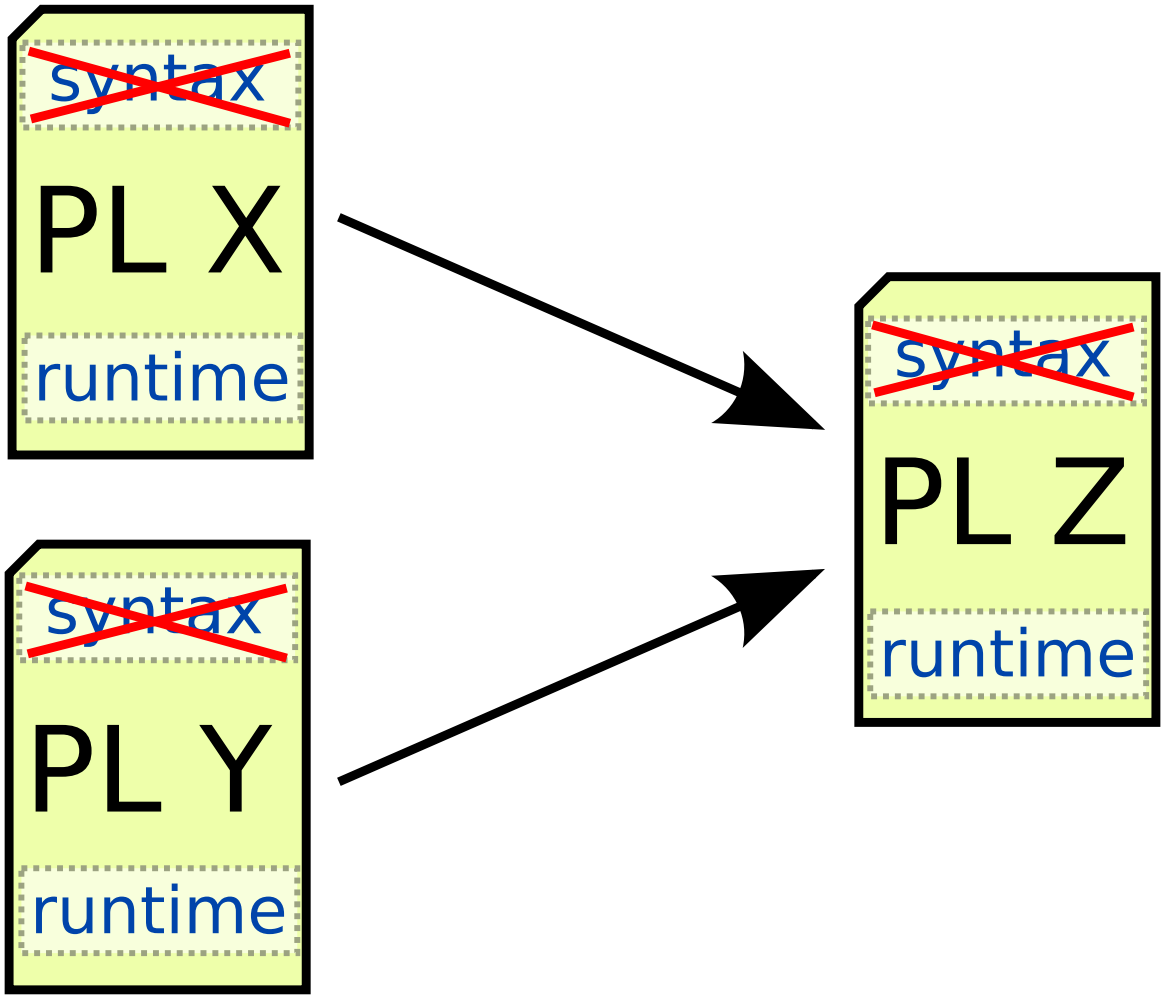
Language Composition



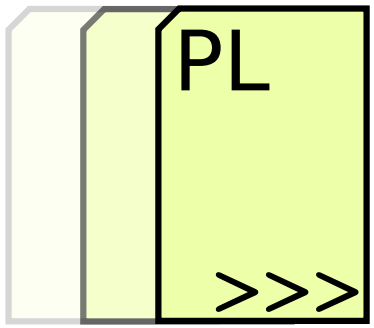
Language Composition

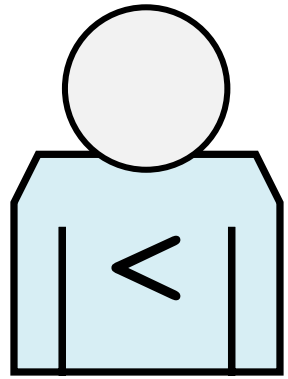


Language Composition



Requirements



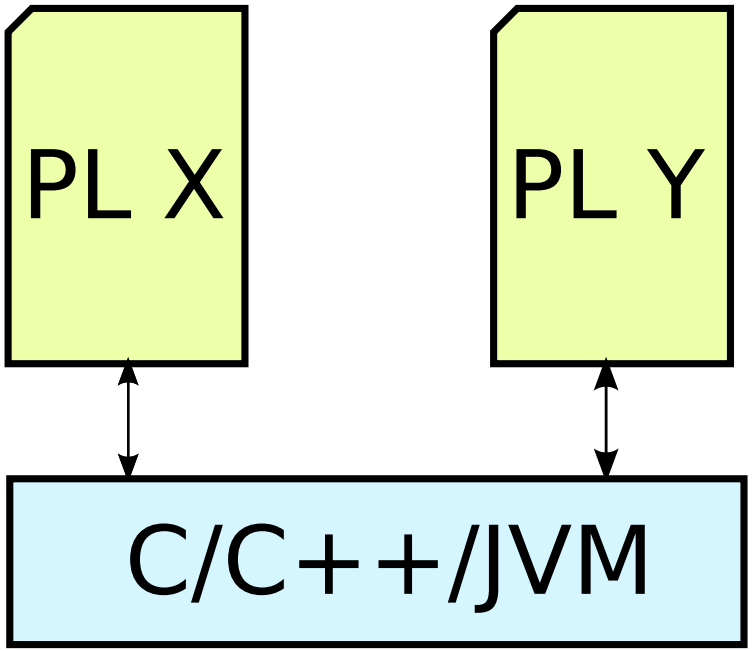


Fast

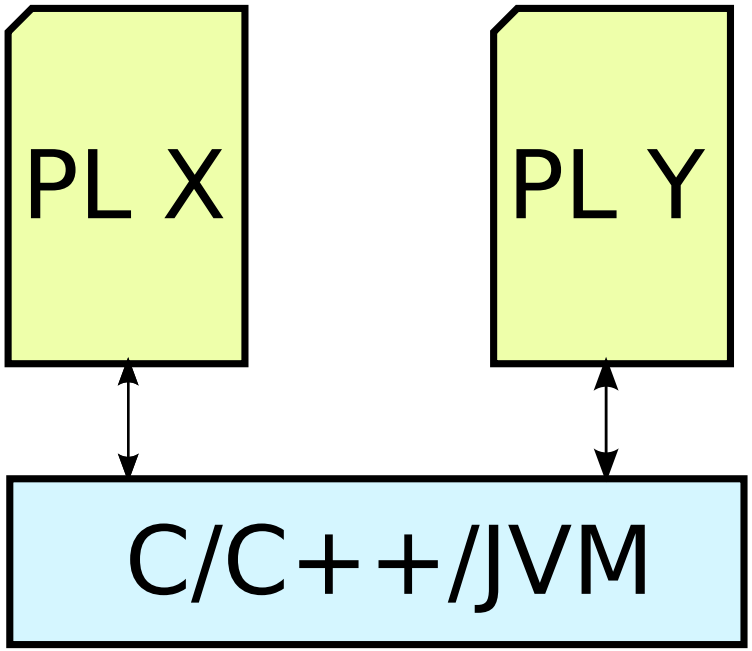
Less effort

Current runtime composition techniques?

Traditional Approaches



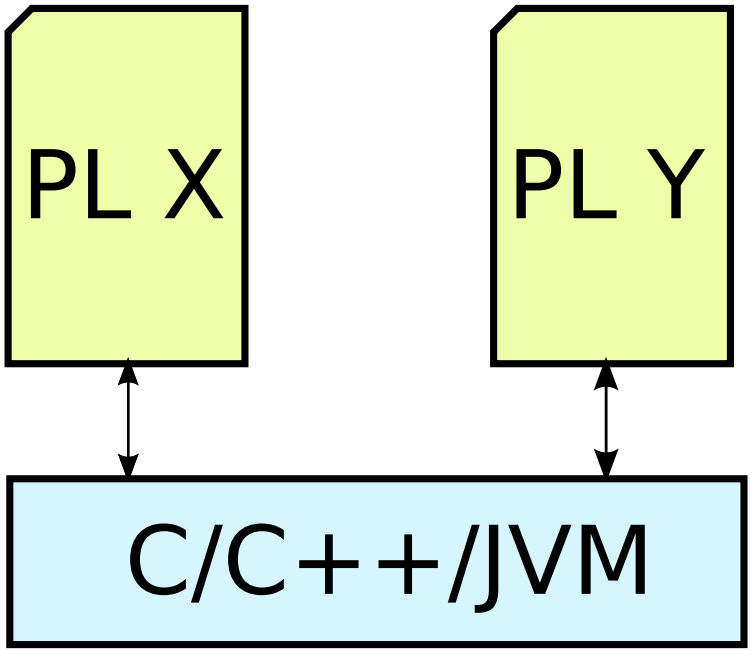
Traditional Approaches



C/C++

- Can build fast VMs.
- $>$ Development effort.

Traditional Approaches

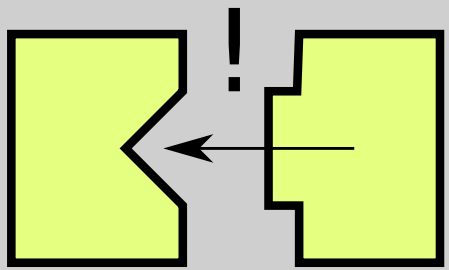


C/C++

- Can build fast VMs.
- $>$ Development effort.

JVM

- $<$ Development effort.
- “Semantic mismatch”.



Hypothesis:

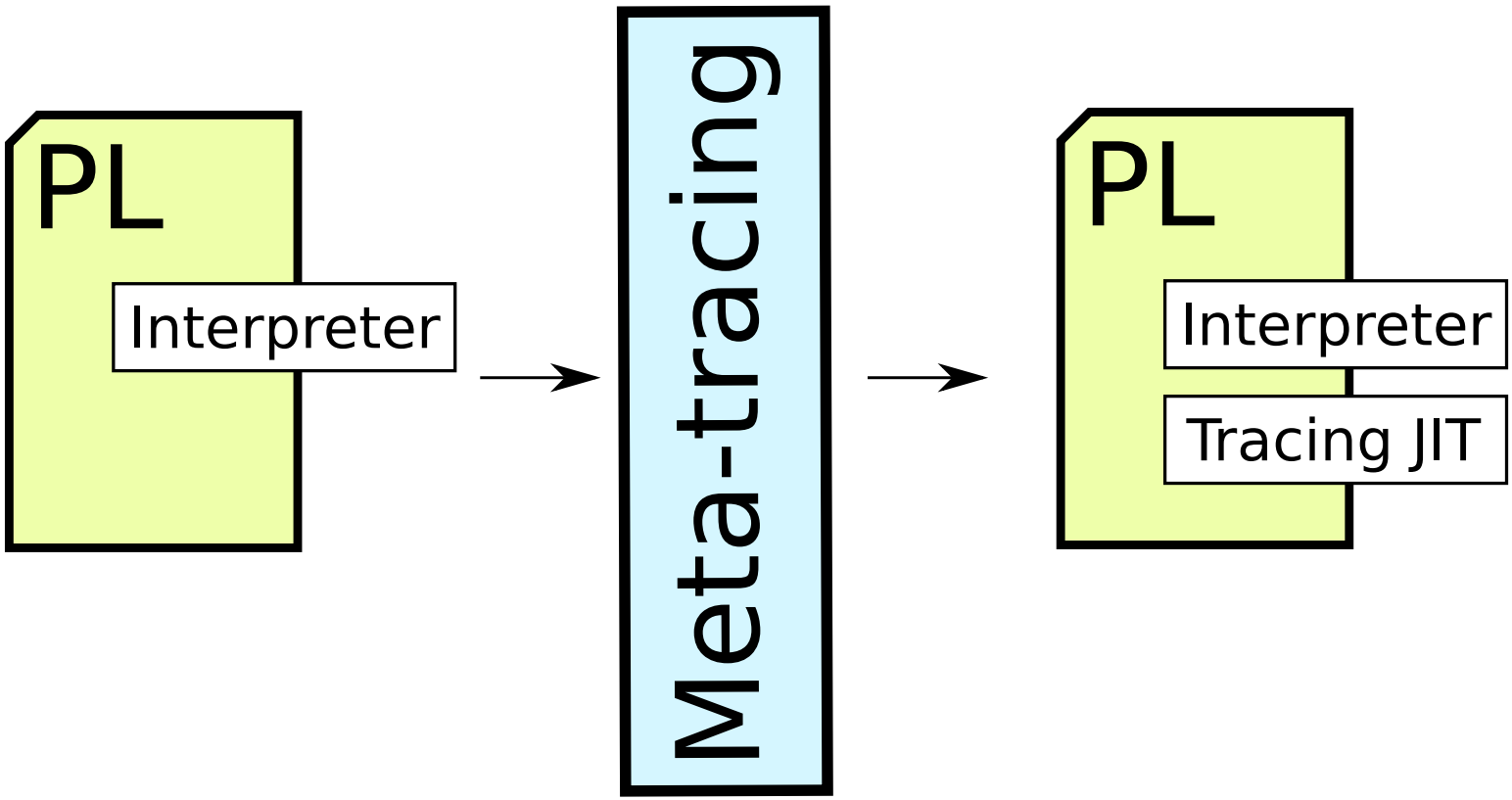
Meta-tracing may overcome these issues.

Hypothesis:

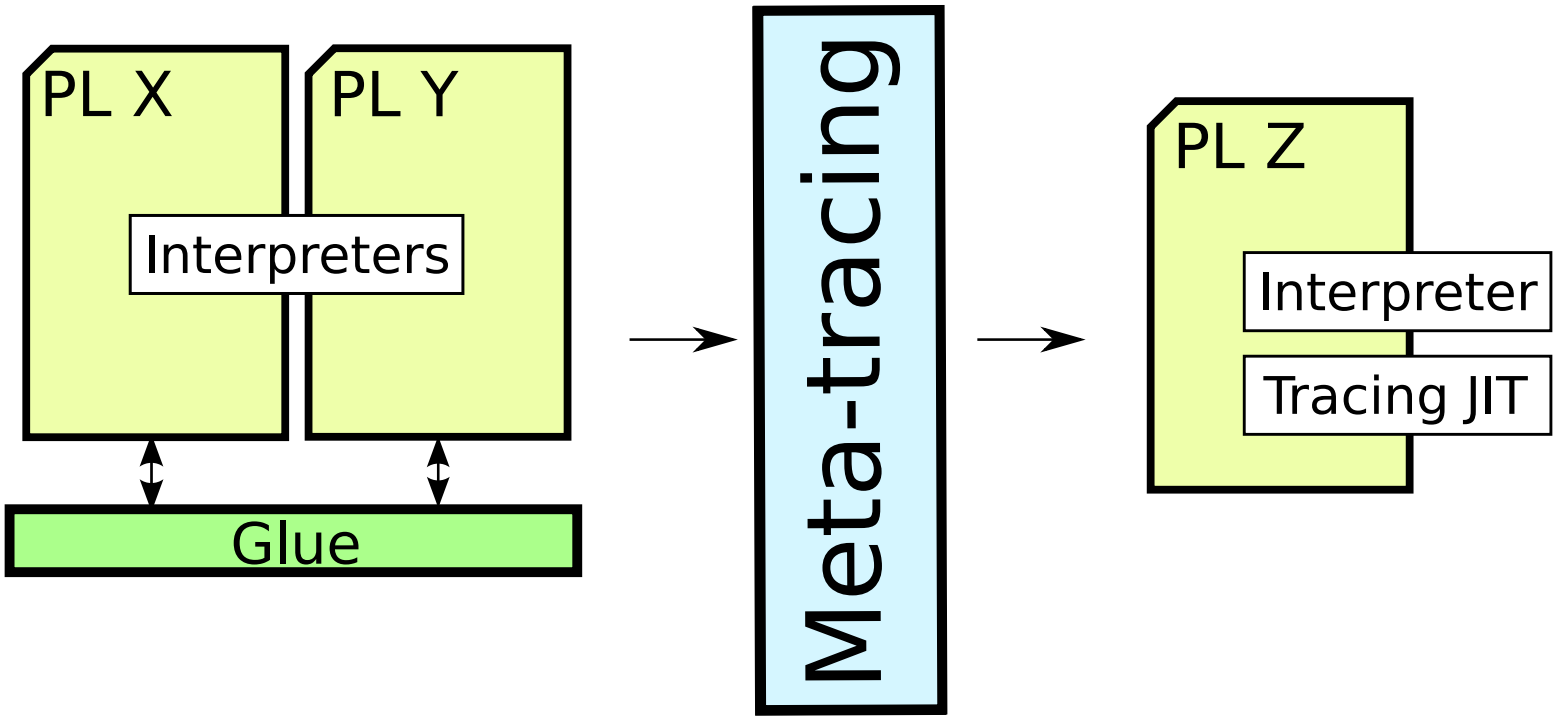
Meta-tracing may overcome these issues.

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

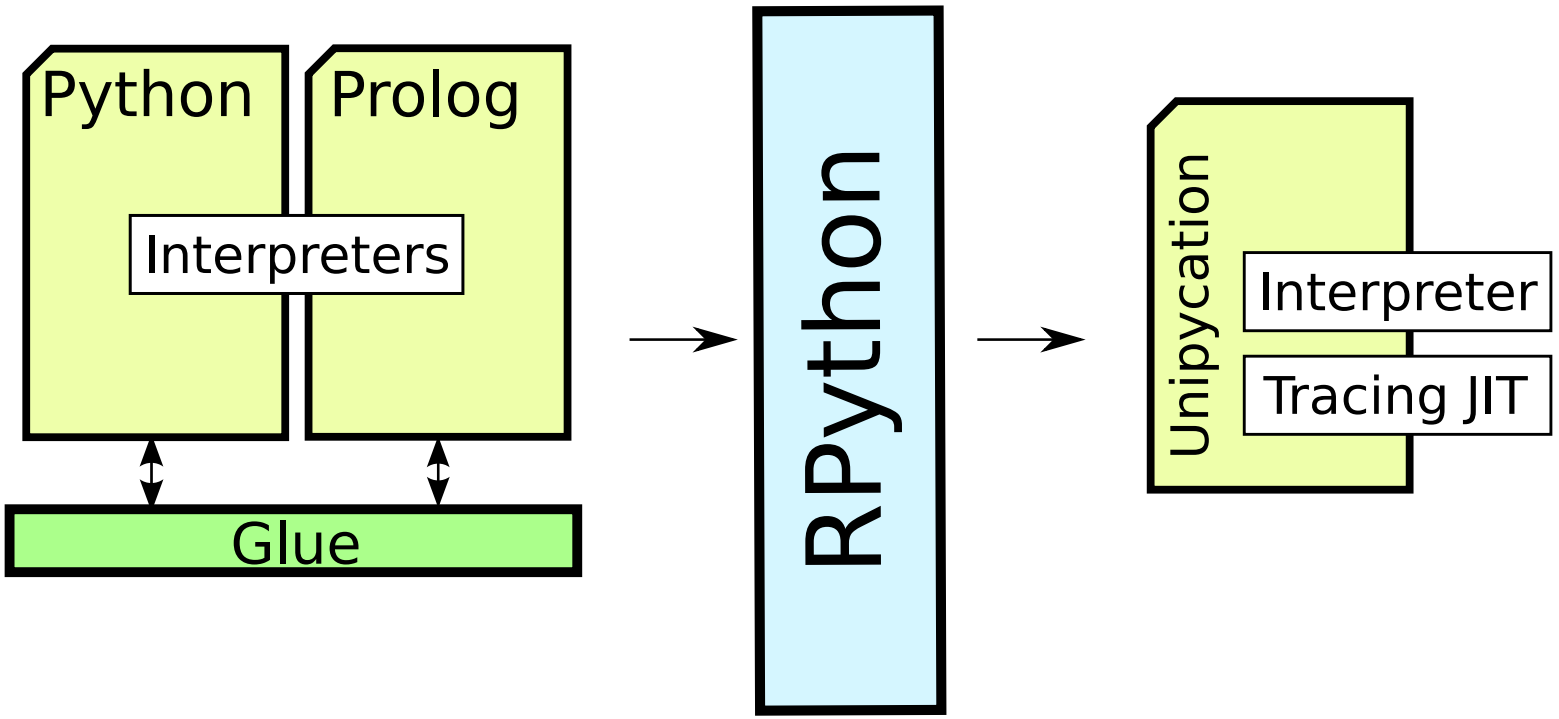
Meta-tracing



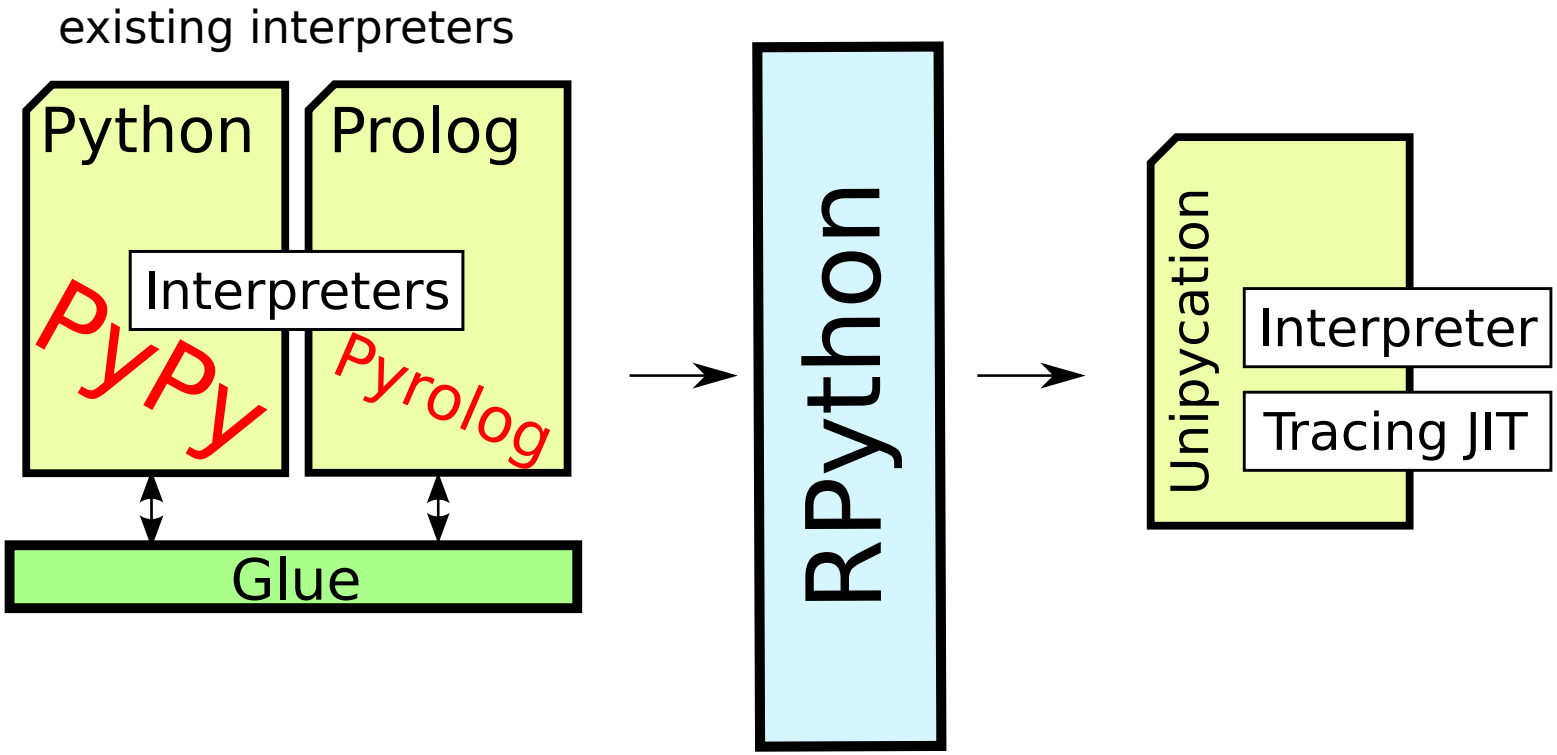
Meta-tracing



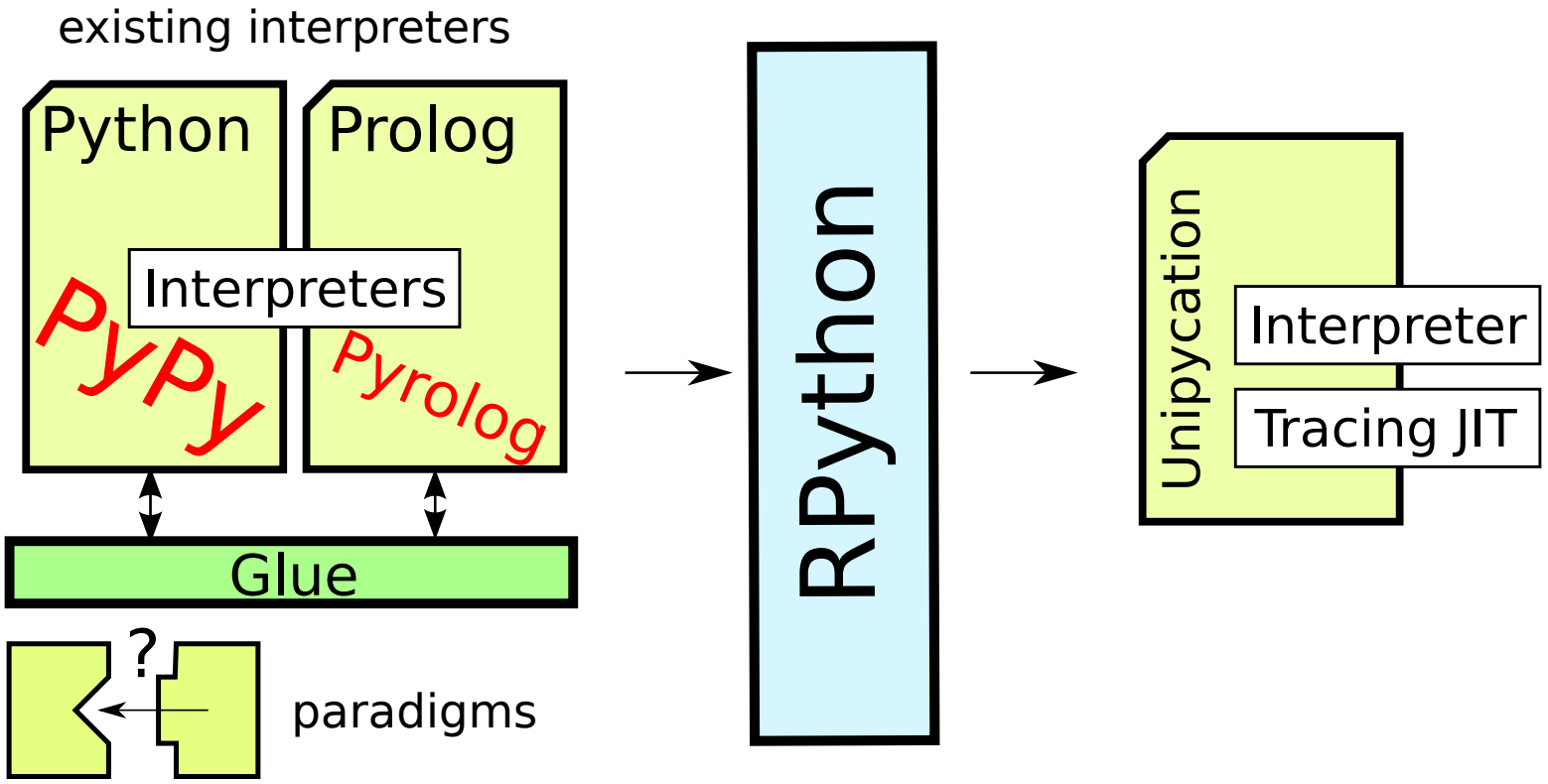
Unipycation



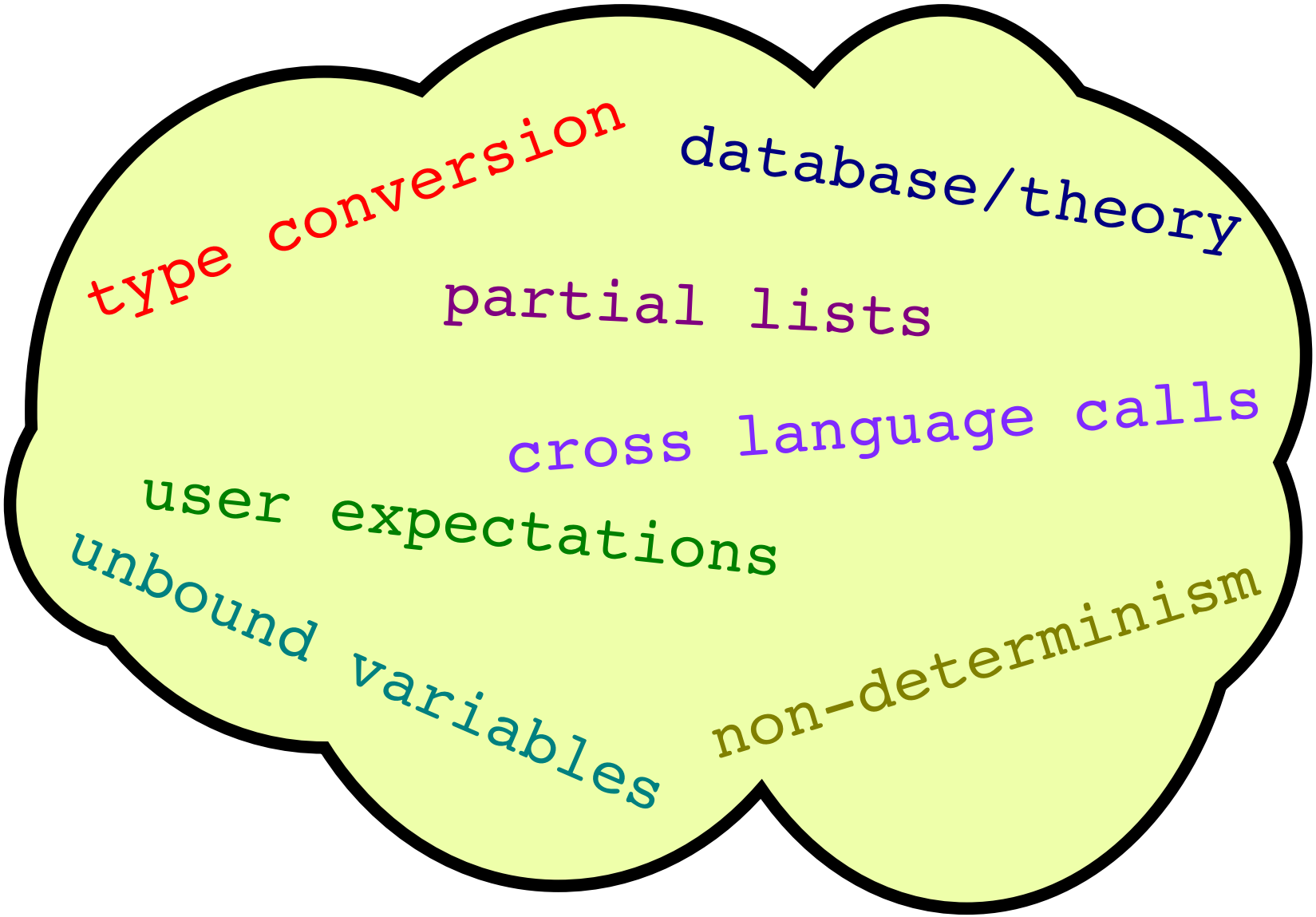
Unipycation



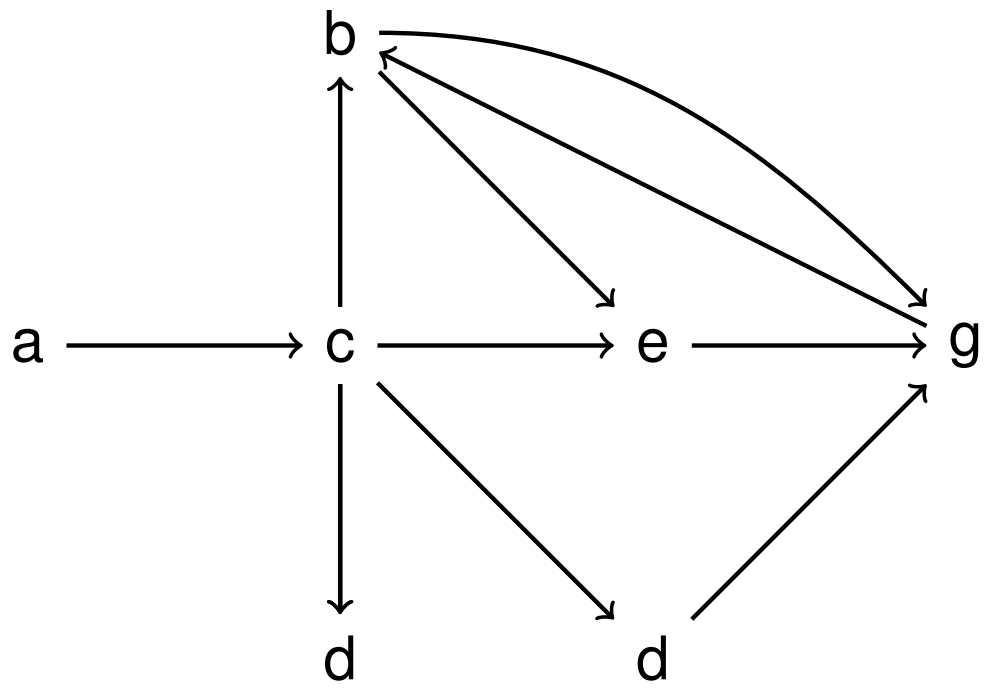
Unipycation



Design Considerations

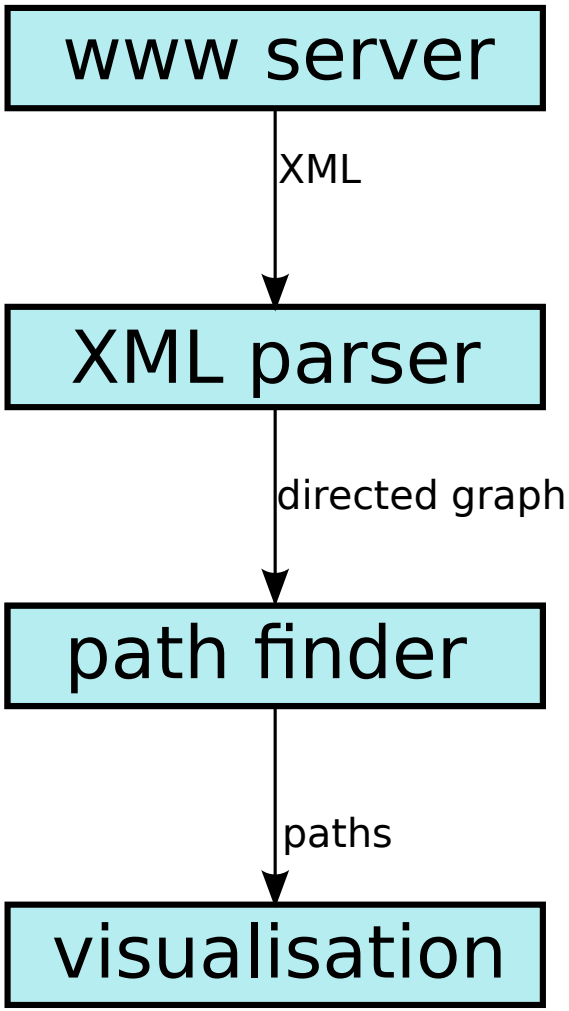


Example: Underground/Metro System

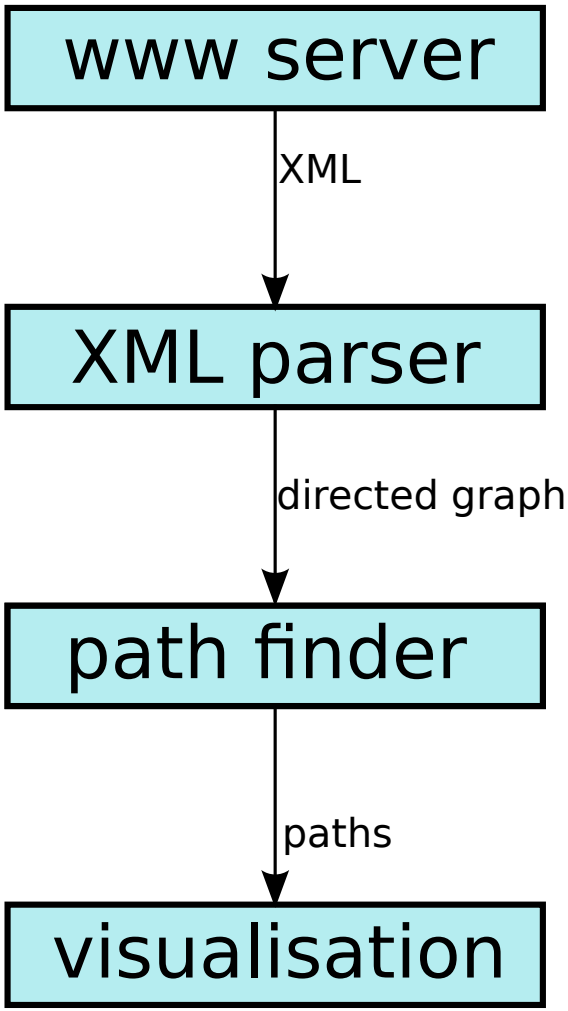


Where can I get to from ’b’ via at most 4 nodes and how?

Example: Underground/Metro System

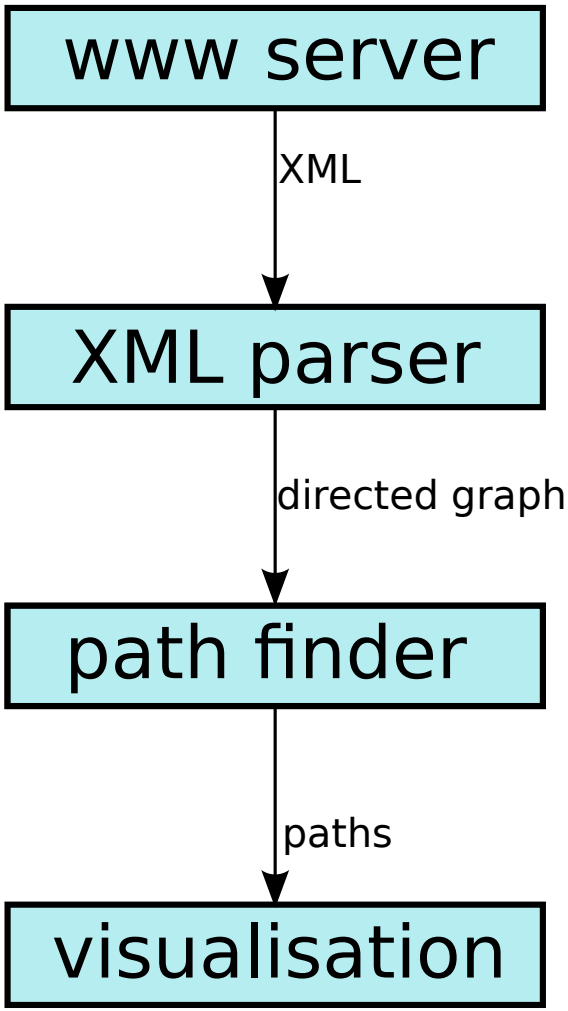


Example: Underground/Metro System

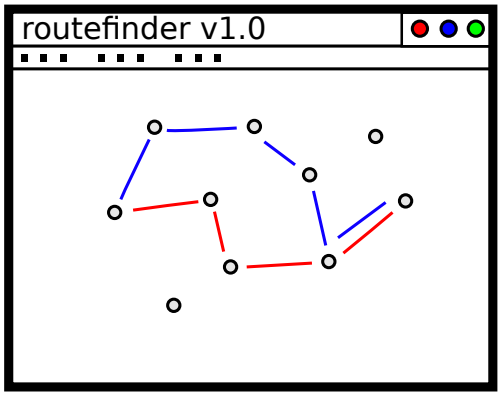



```
<edge src="a" dest="b" />
<edge src="a" dest="c" />
<edge src="b" dest="e" />
```

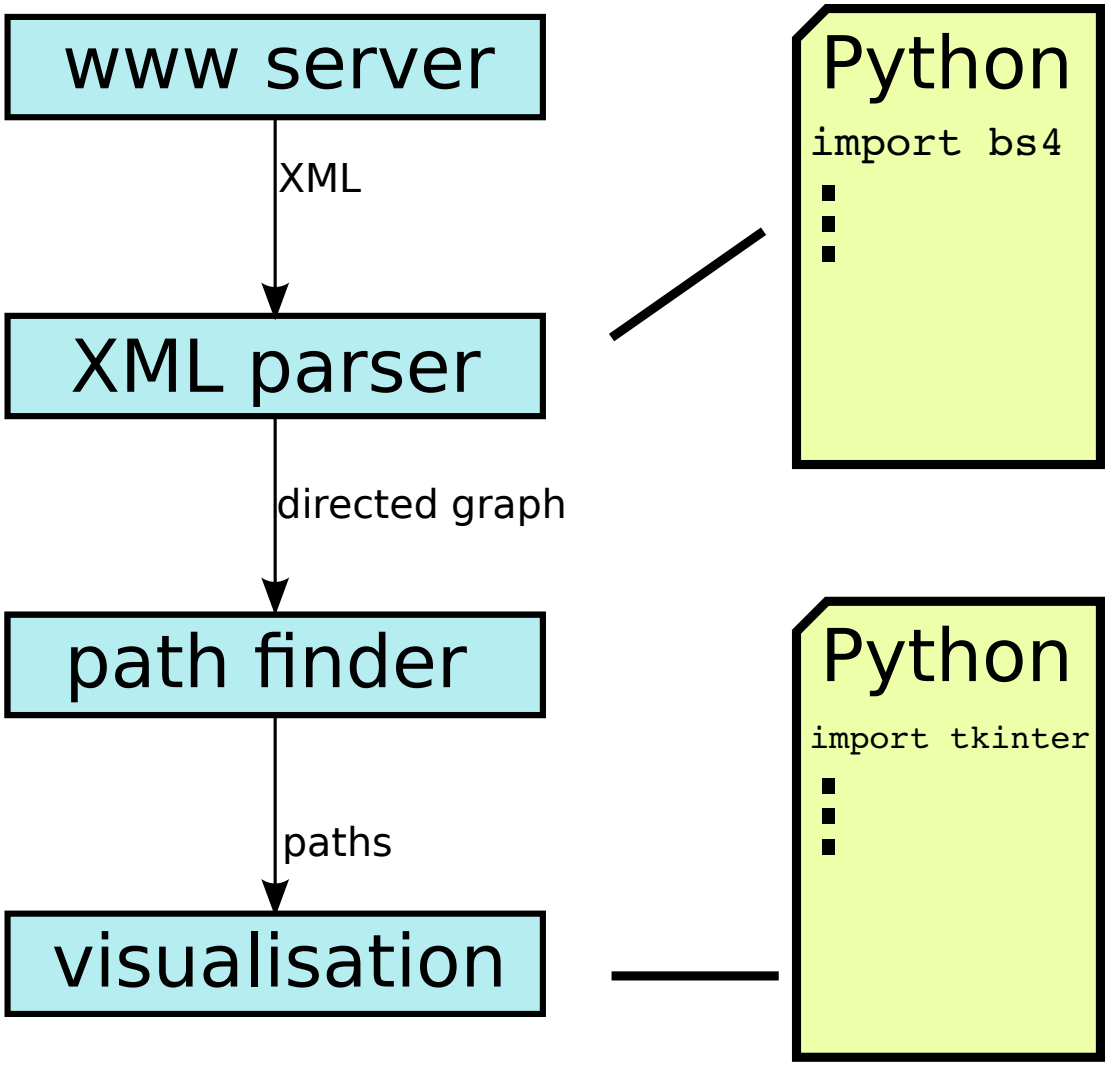
Example: Underground/Metro System



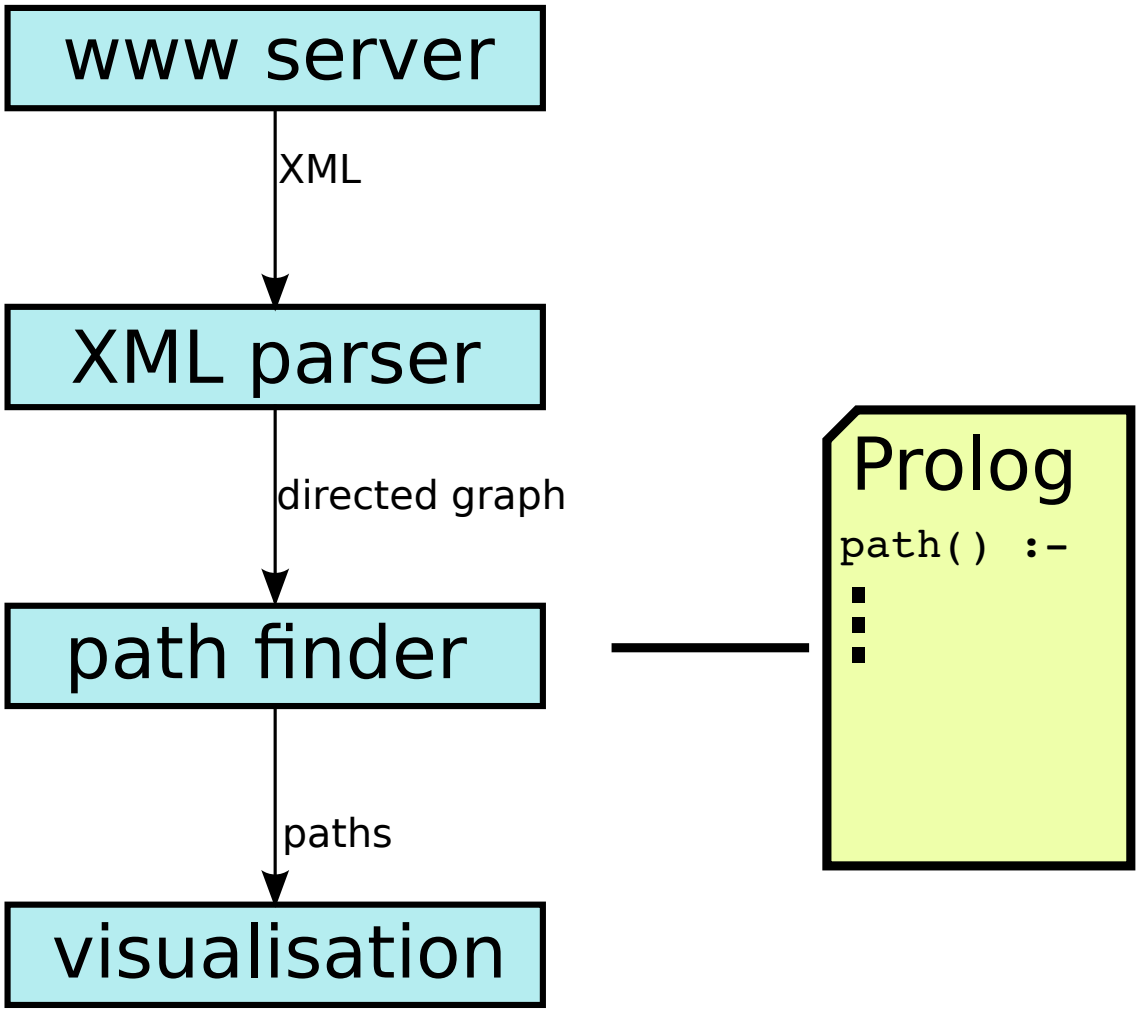

```
<edge src="a" dest="b" />
<edge src="a" dest="c" />
<edge src="b" dest="e" />
```



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

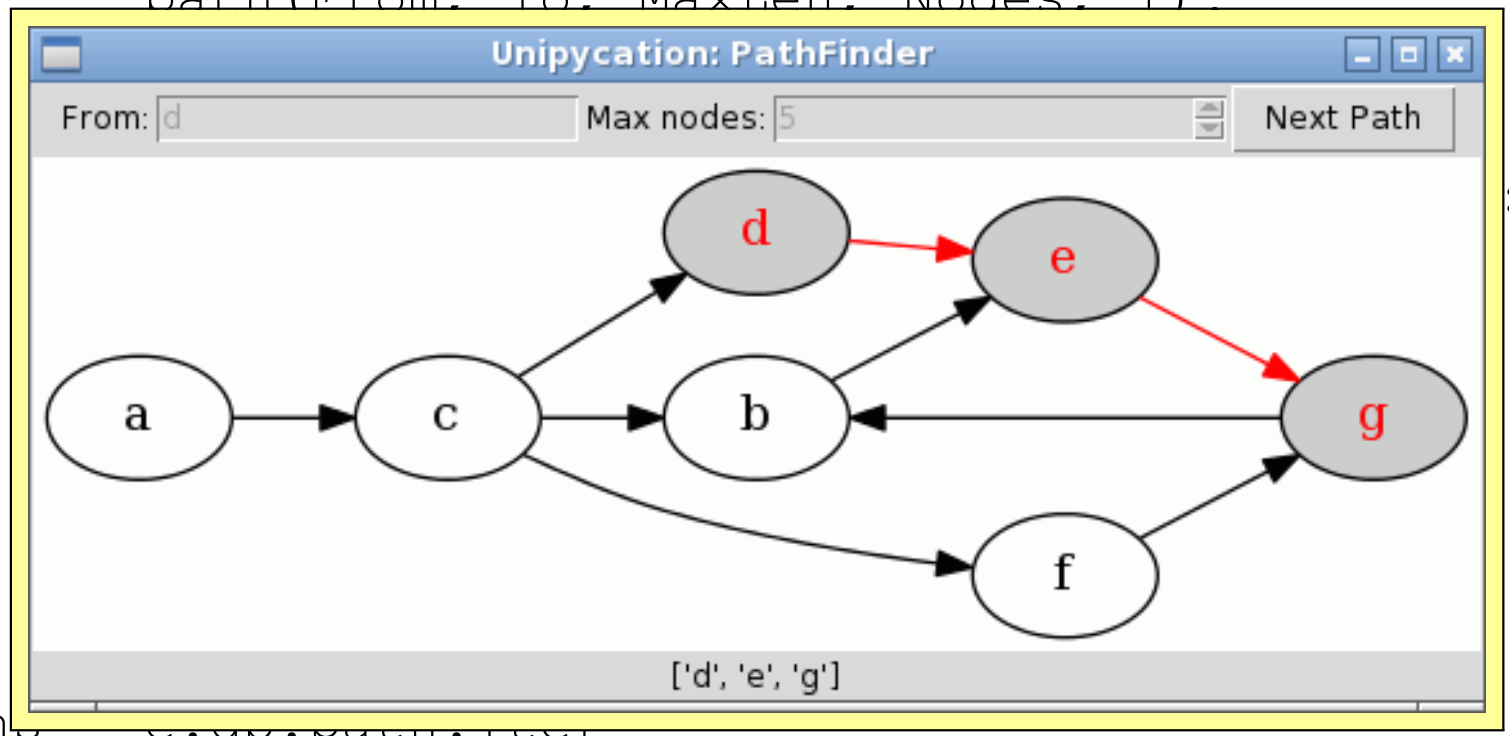
```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).

    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).
""")

def get_edges(src_node):
    return iter(edges[src_node])

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

Example: Underground/Metro System

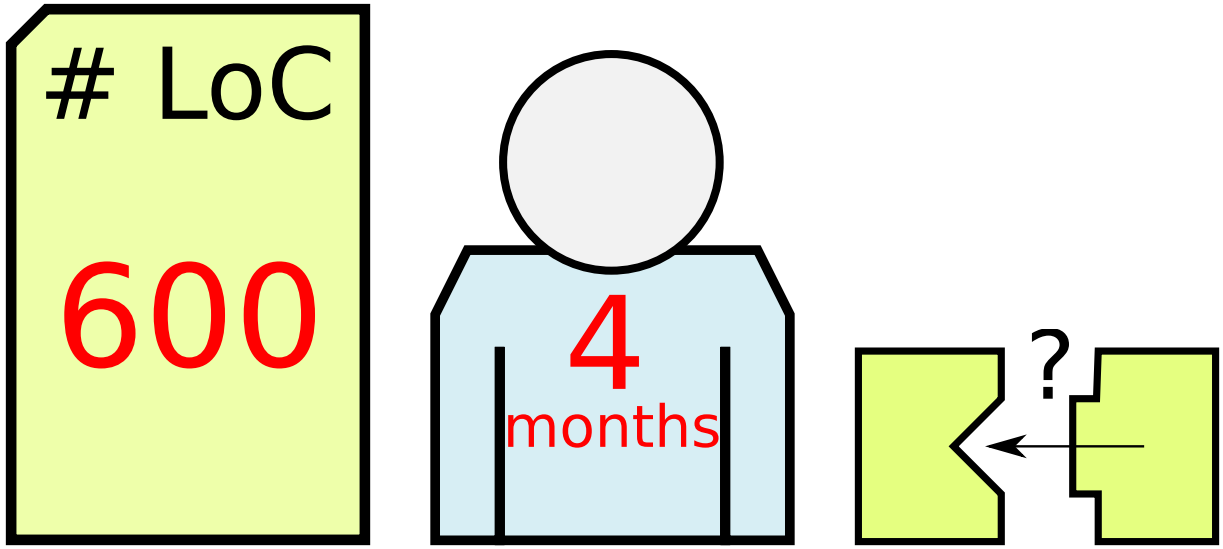
```
e = uni.Engine("""
    path(From, To, MaxLen, Nodes) :-
        path(From, To, MaxLen, Nodes, 1).
    path(From, To, MaxLen, [From], _) :-
        From = To.
    path(From, To, MaxLen, [From|Nodes], Len) :-
        Len < MaxLen,
        edge(From, Next),
        path(Next, To, MaxLen, Nodes, Len+1).
    edge(a, c).
    edge(c, b).
    edge(c, d).
    edge(d, e).
    edge(e, b).
    edge(e, g).
    edge(b, g).
    edge(c, f).
    edge(f, g).
""")

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



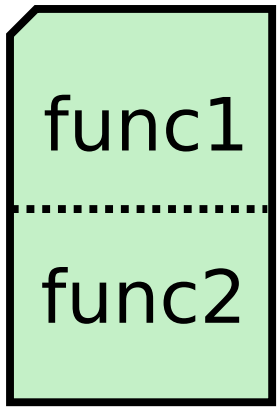
Development Effort

How much effort was our composition?

Development Effort

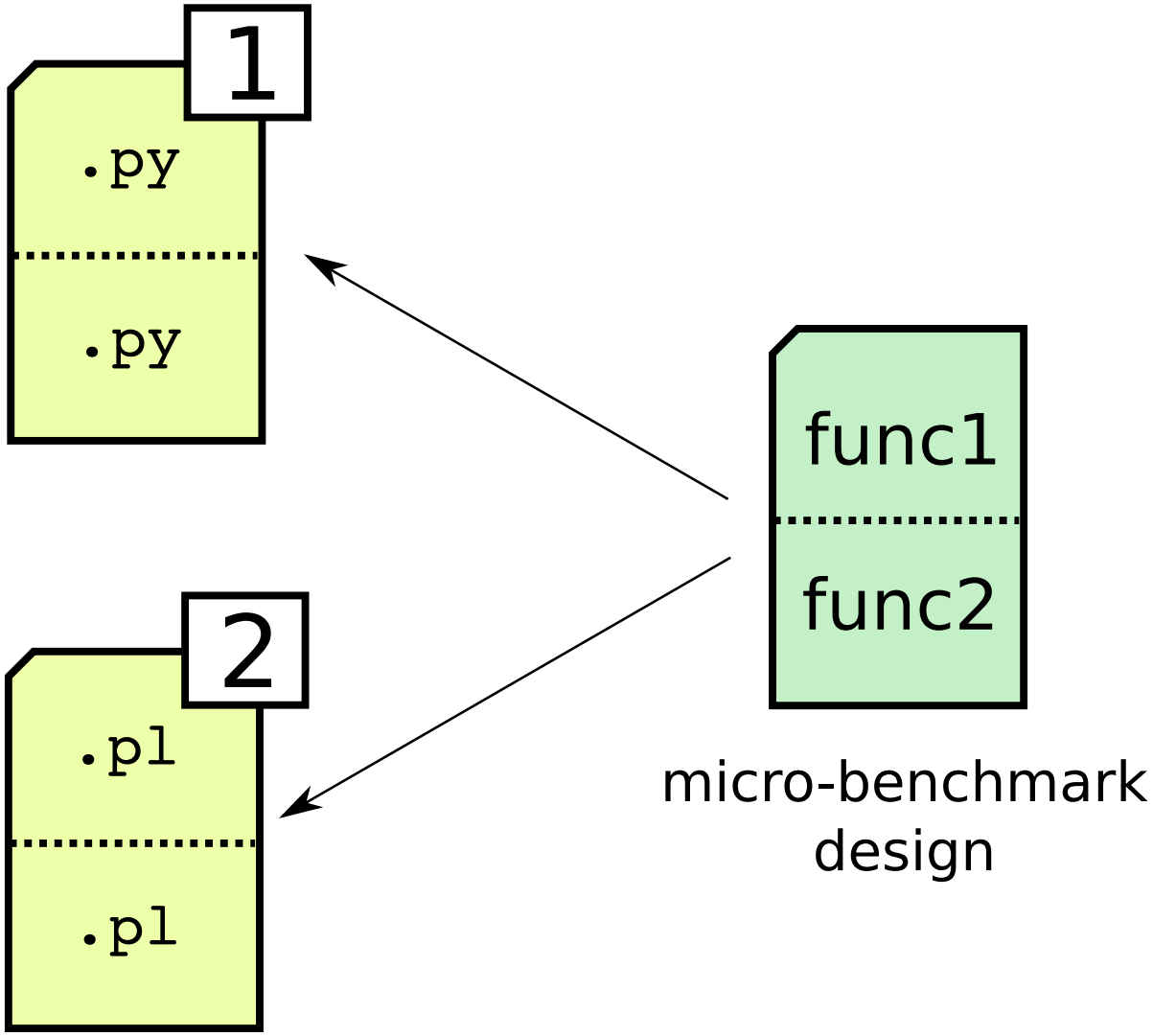


Experimental Methodology

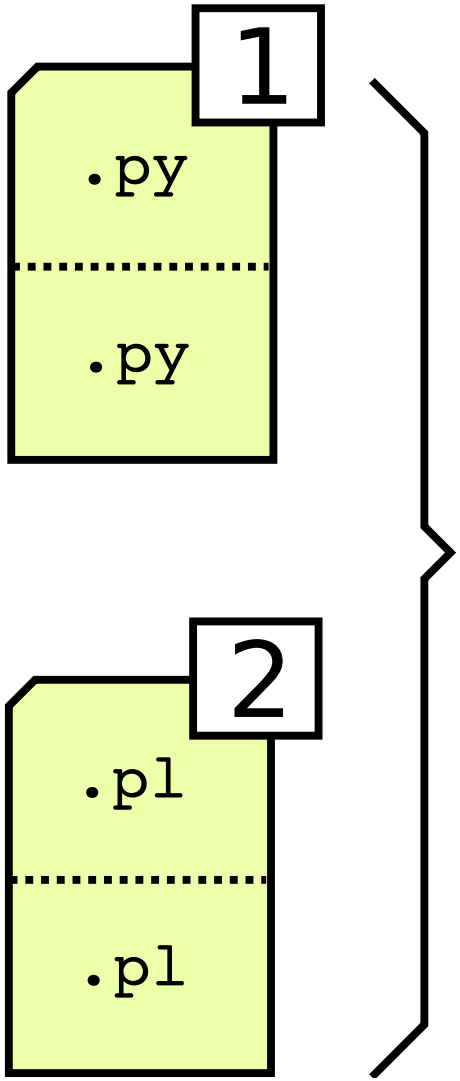


micro-benchmark
design

Experimental Methodology

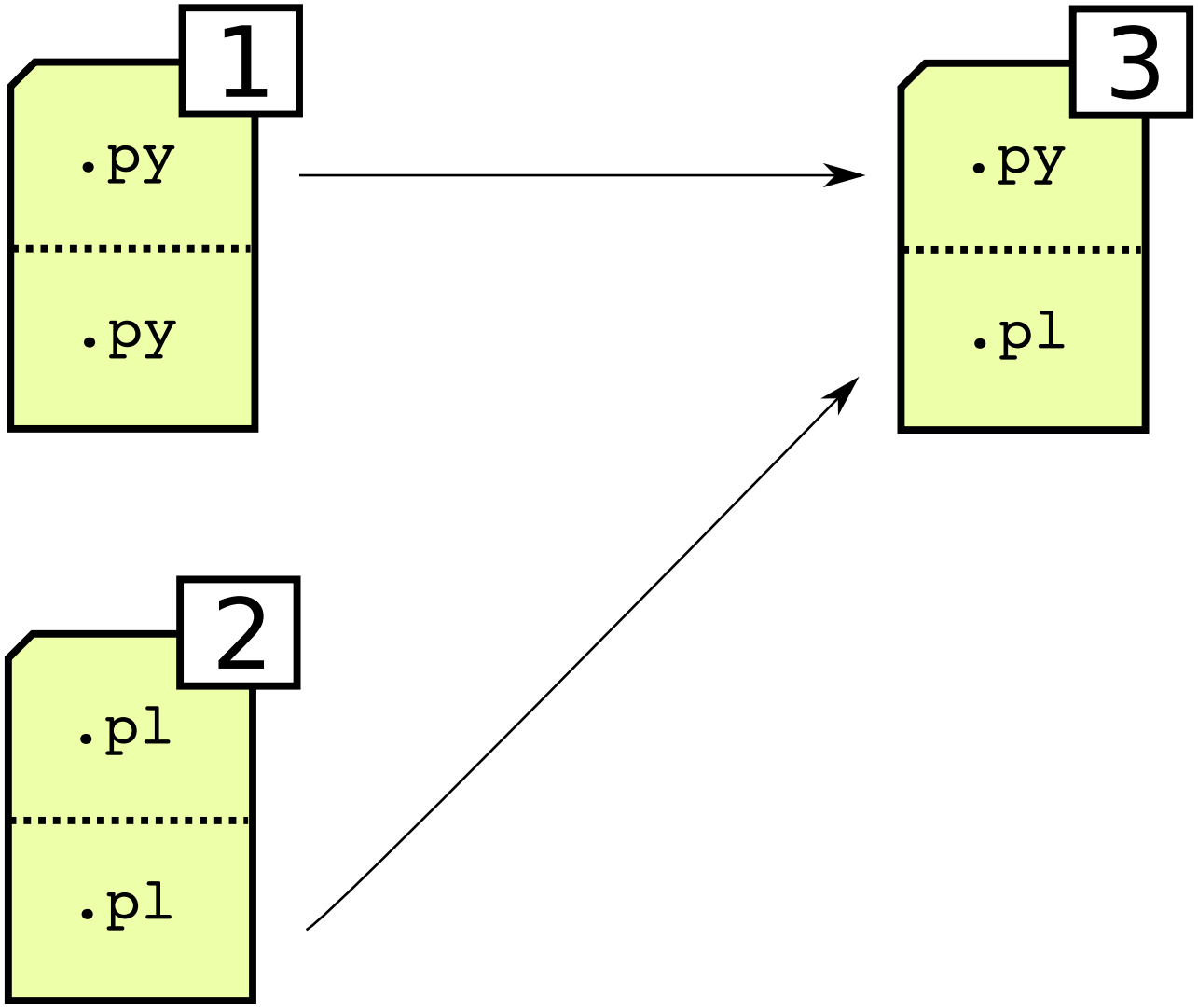


Experimental Methodology

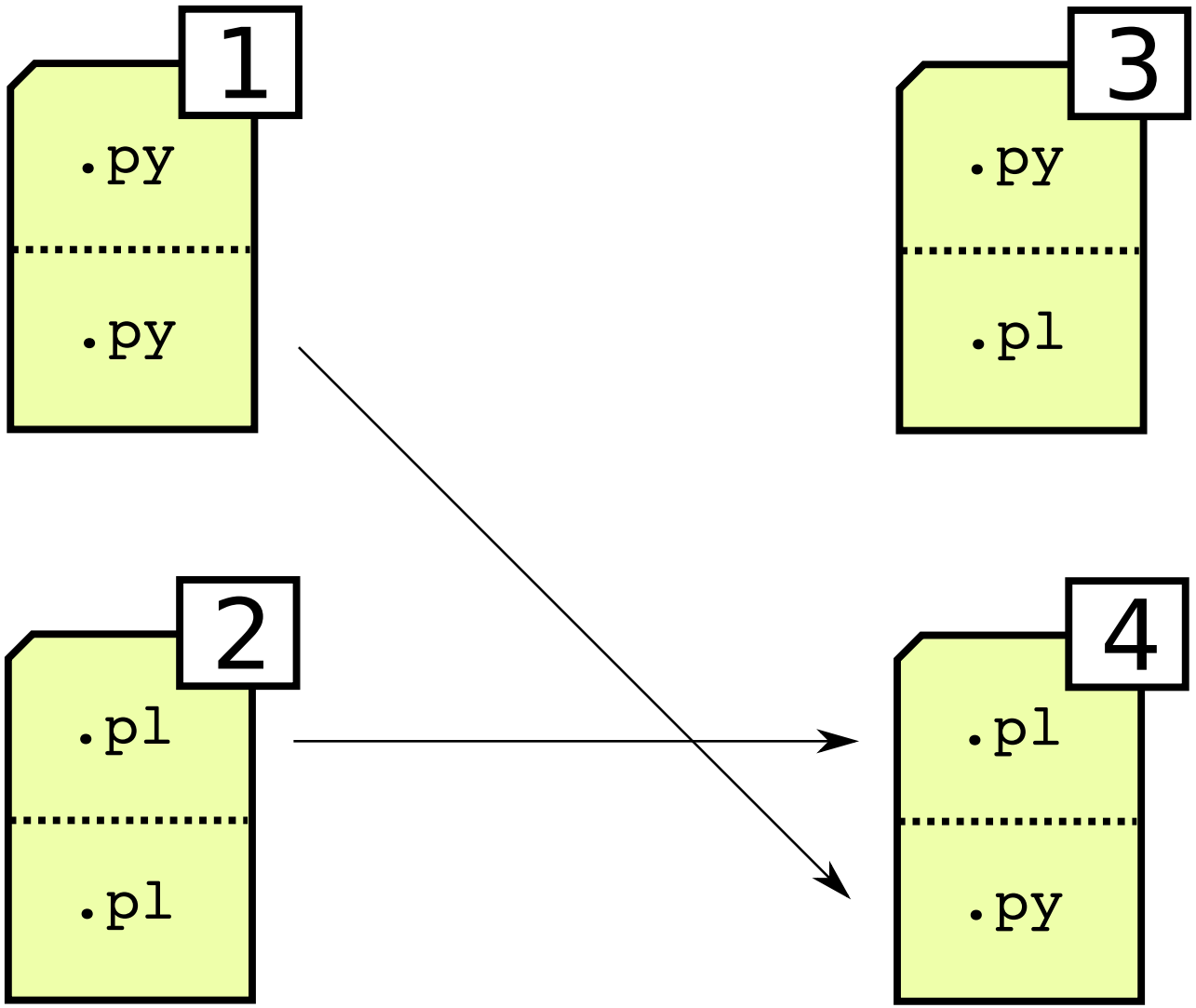


Baseline measurements

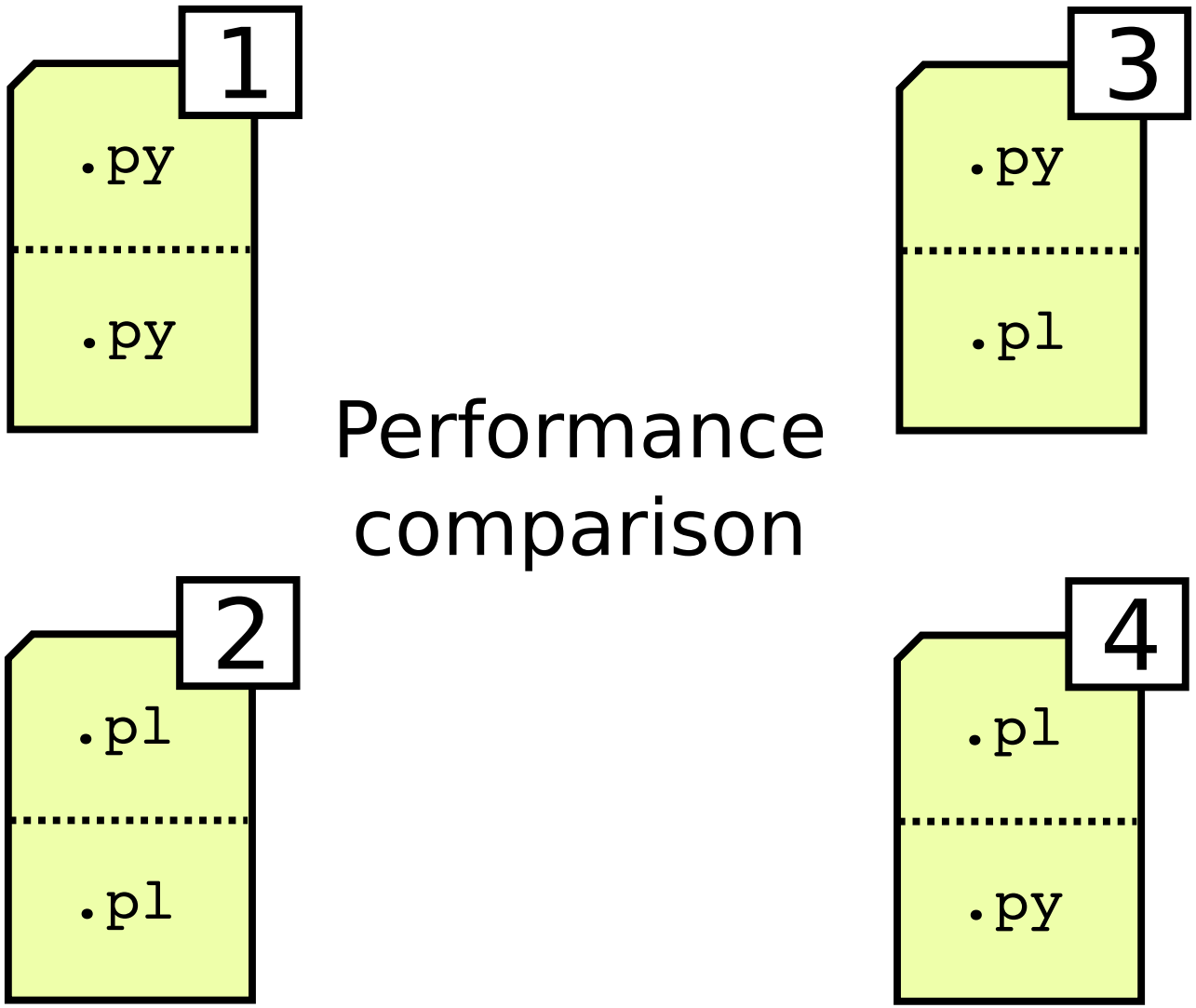
Experimental Methodology



Experimental Methodology



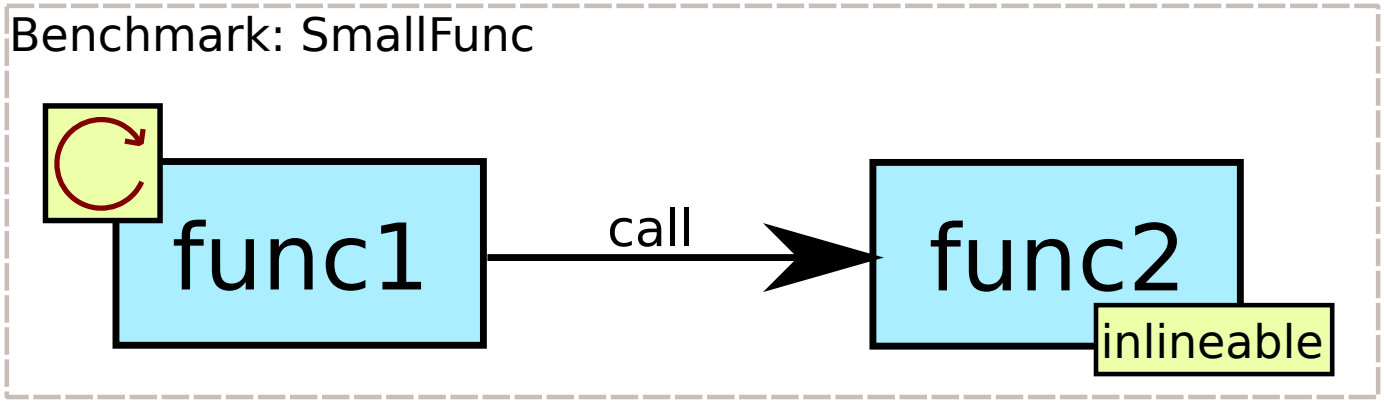
Experimental Methodology



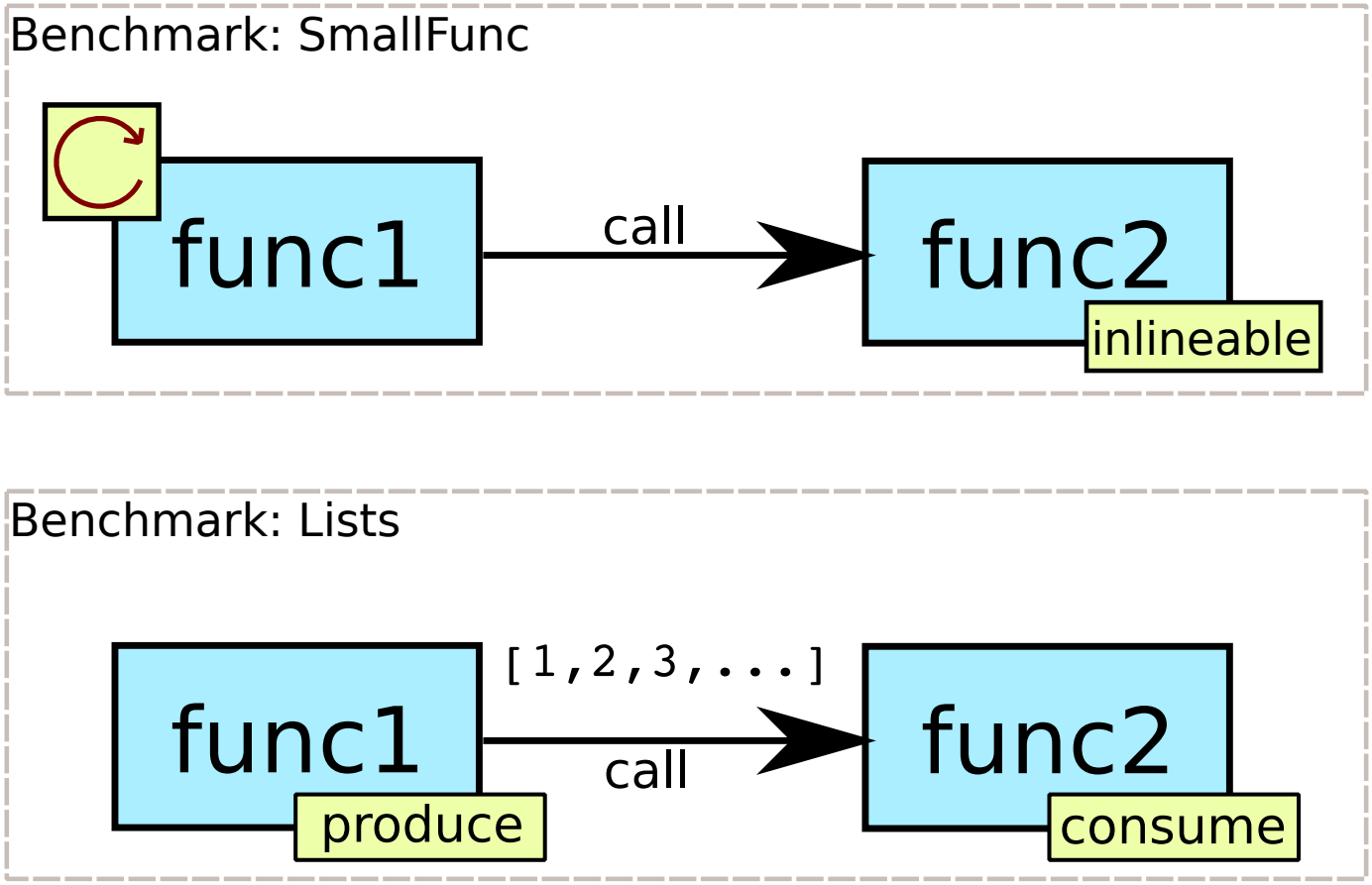
Experimental Methodology

7 micro-benchmarks, e.g.:

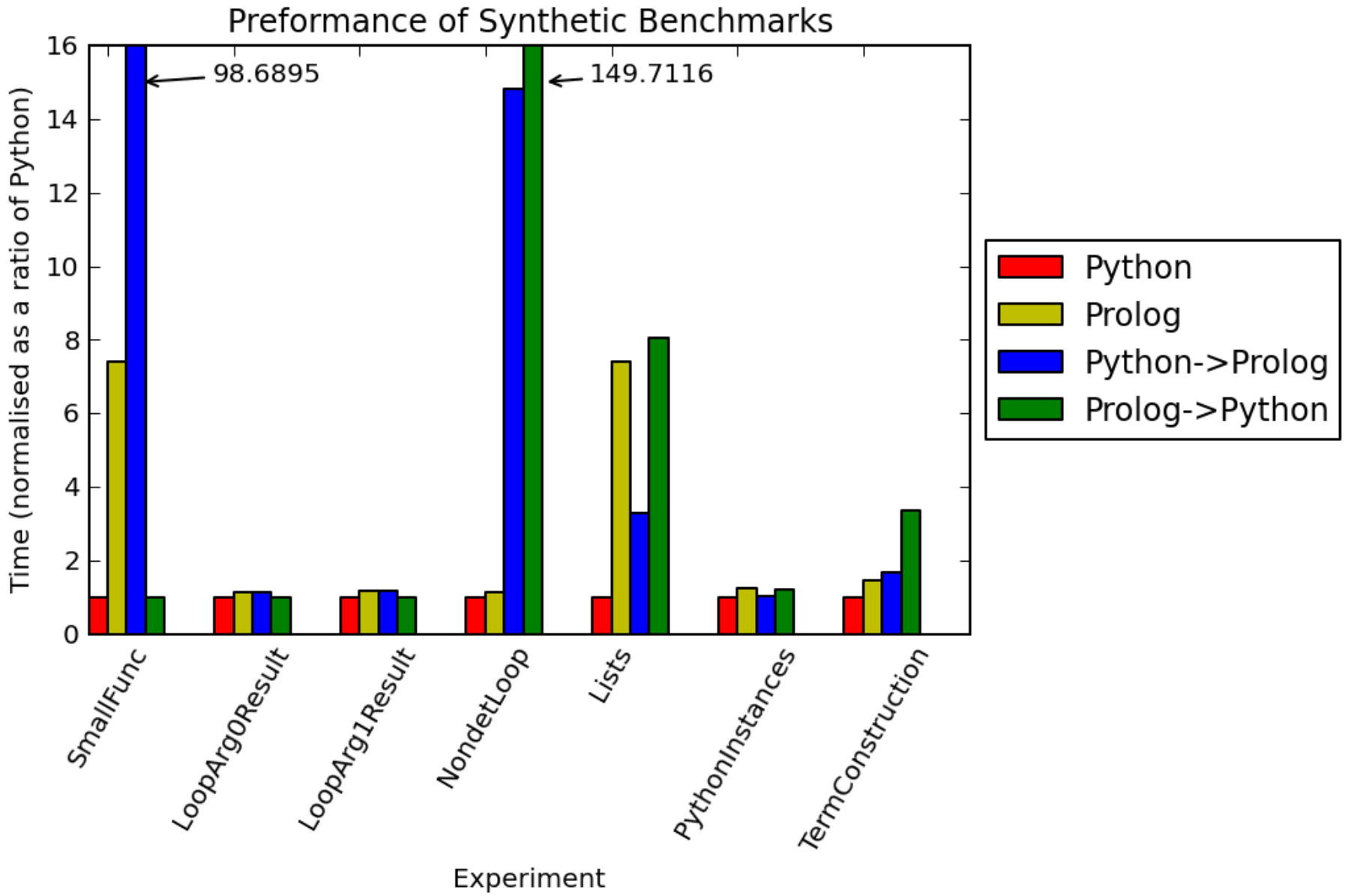
7 micro-benchmarks, e.g.:



7 micro-benchmarks, e.g.:

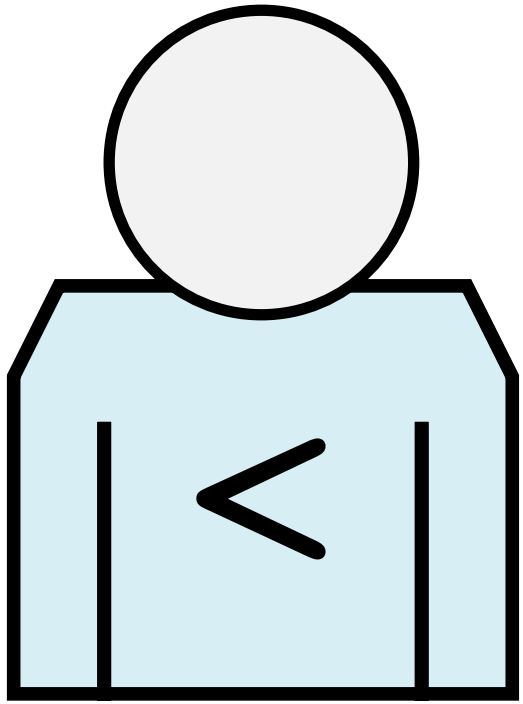


Experimental Evaluation



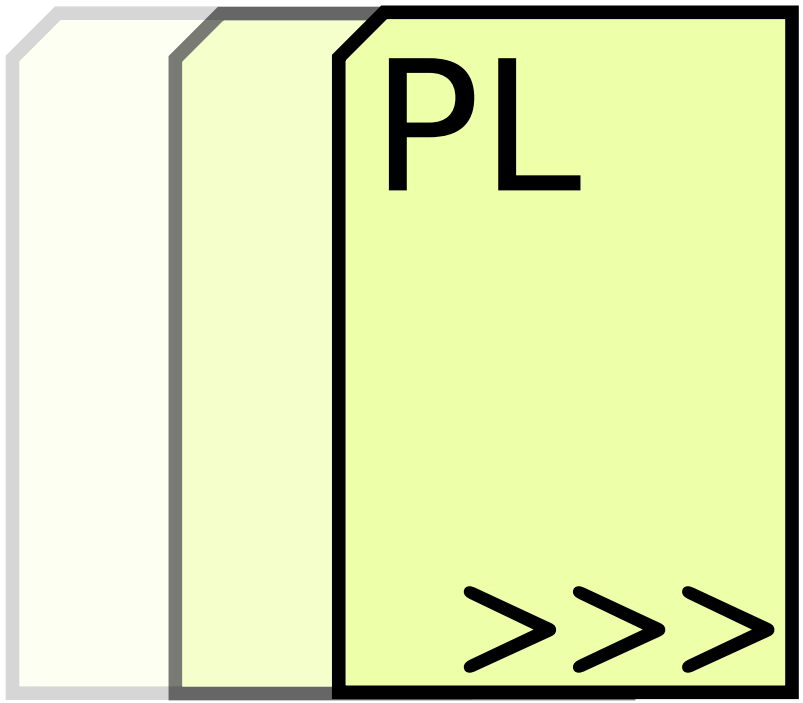
In Summary

In Summary

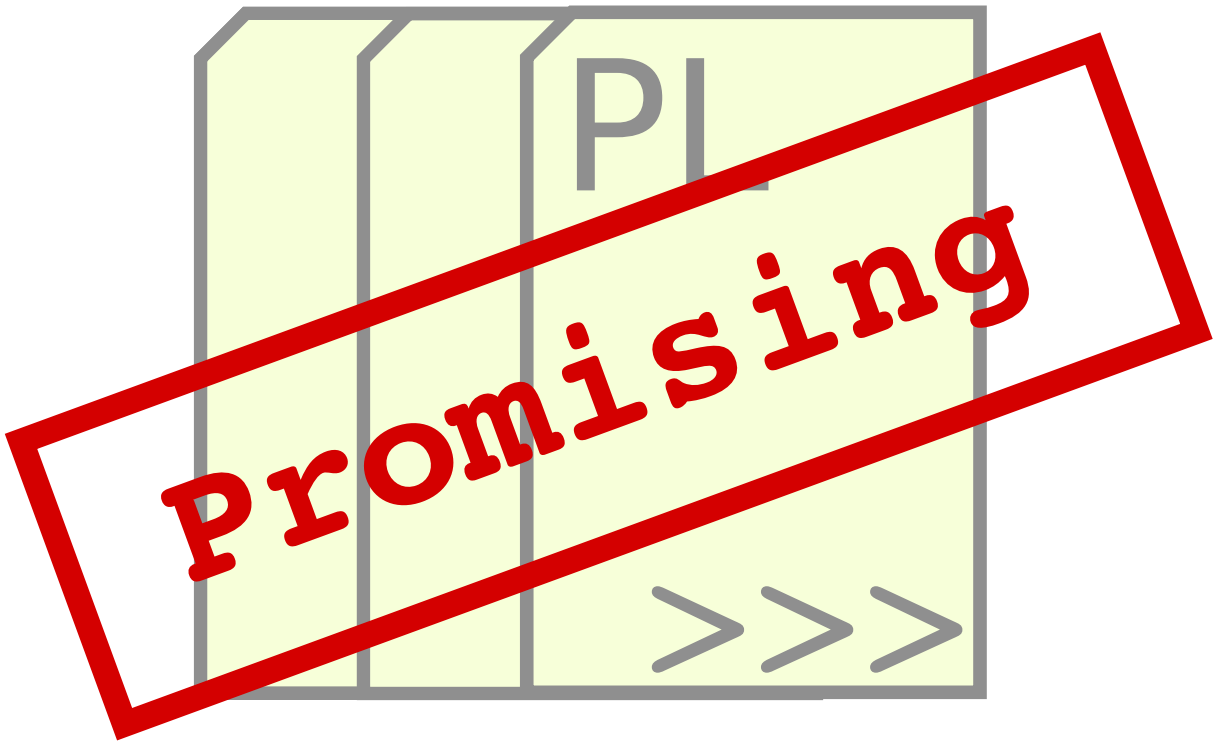


In Summary

In Summary

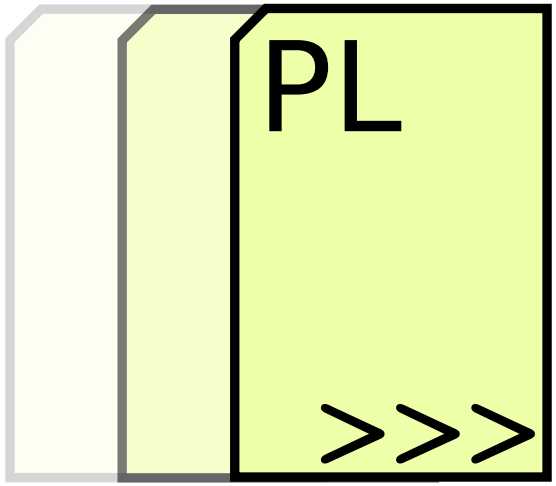


In Summary

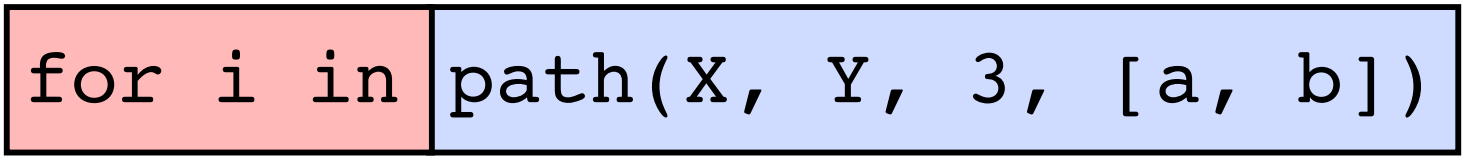


What is next?

Future Work

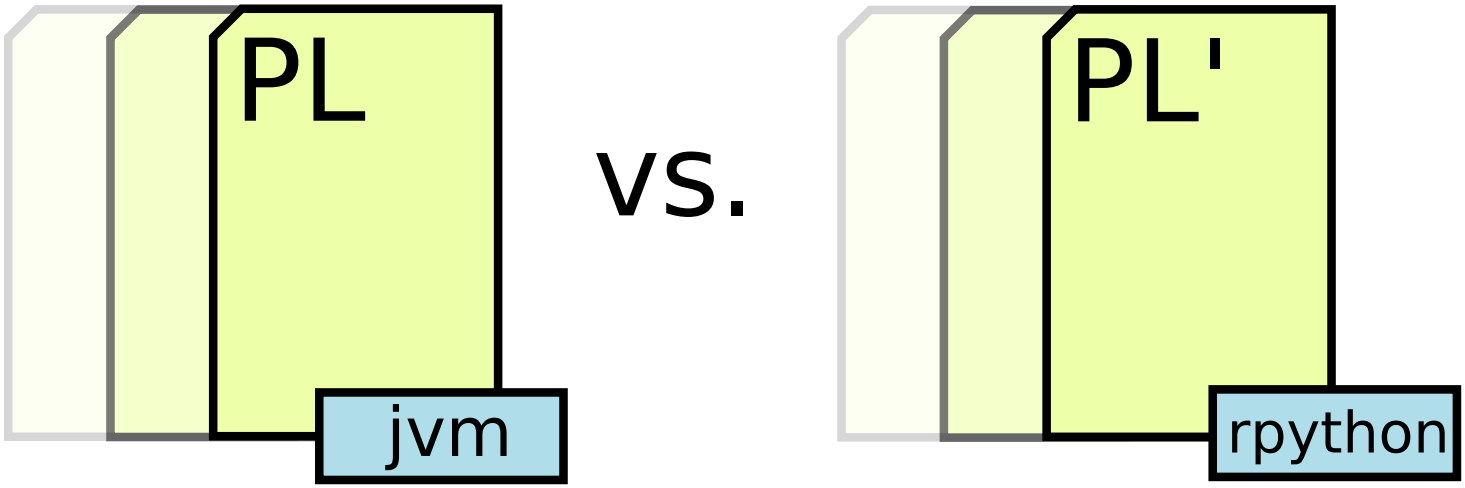


Faster

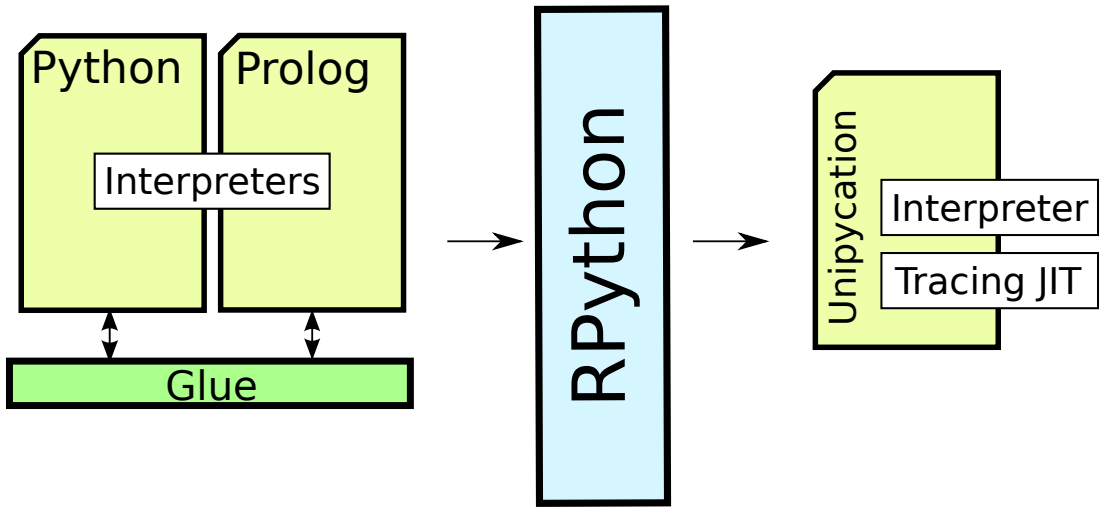


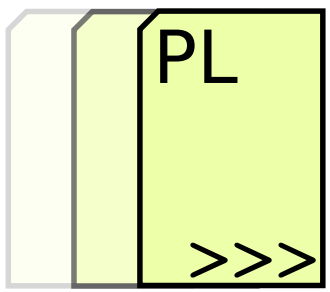
Syntax

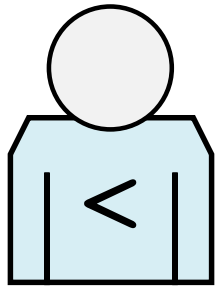
Future Work



Demo/Question Time







http://soft-dev.org