



Optimisations in the HotPy Virtual Machine



University
of Glasgow

Mark Shannon

University of Glasgow

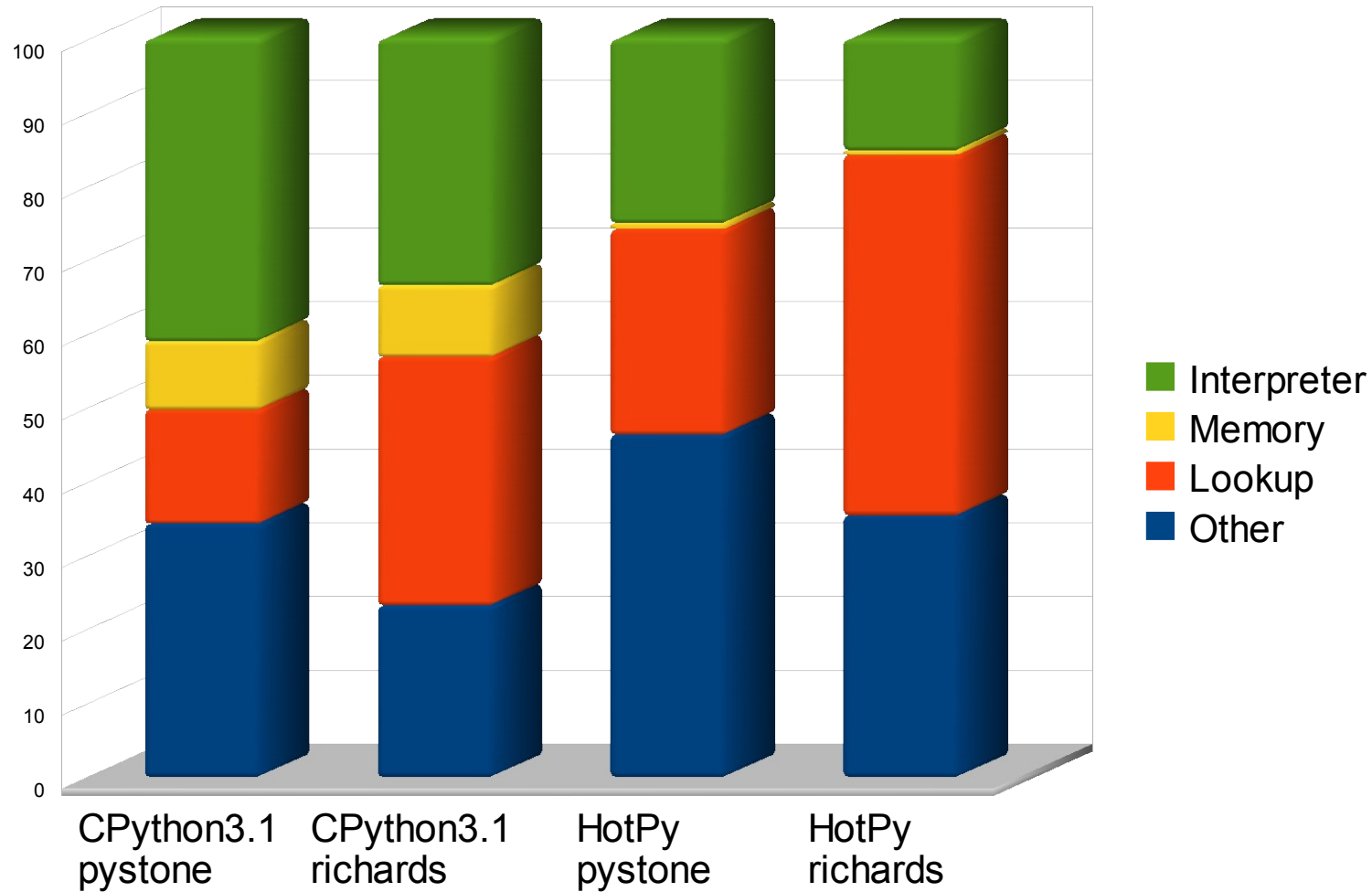


The GVMT

- Glasgow Virtual Machine Toolkit
- Provides an abstract stack machine layer
 - Garbage collection
 - Exceptions
 - Translation of stack code to underlying machine
- Automatic generation of interpreter *and* compiler from same specification
- HotPy is built with the GVMT

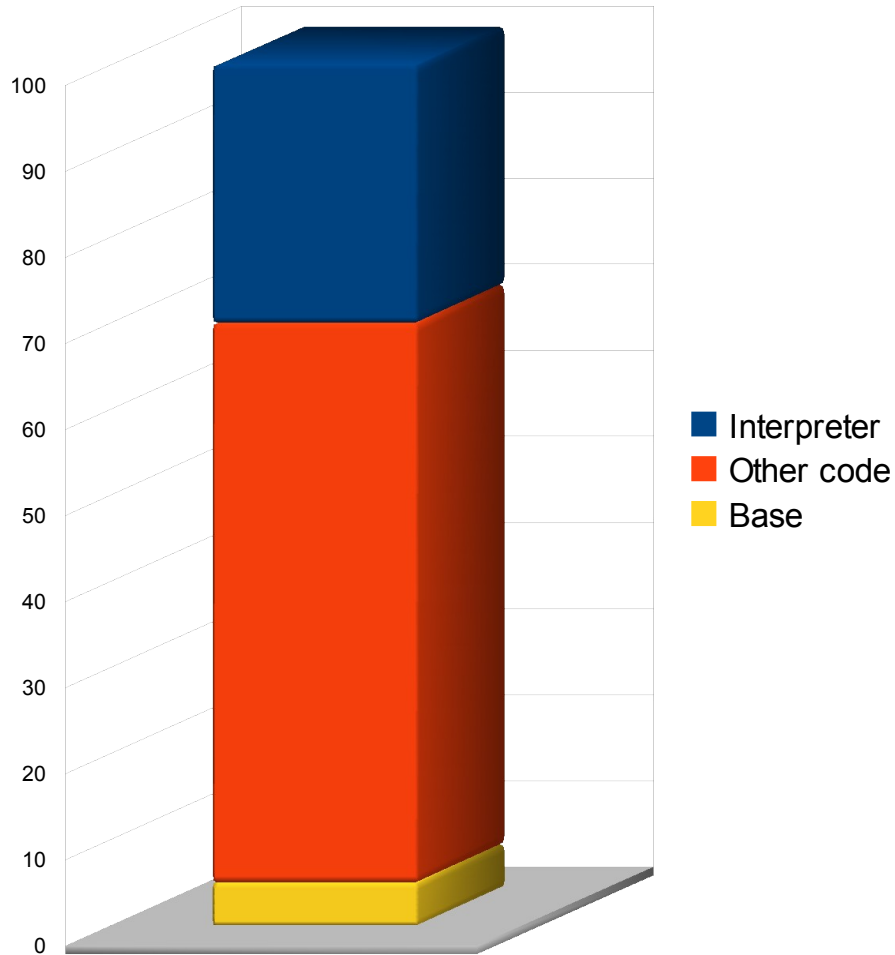


Profiling the VM





Optimisation and Compilation



- “Typical” application
- 30% interpreter
- 65% other overhead
- 5% “real work”

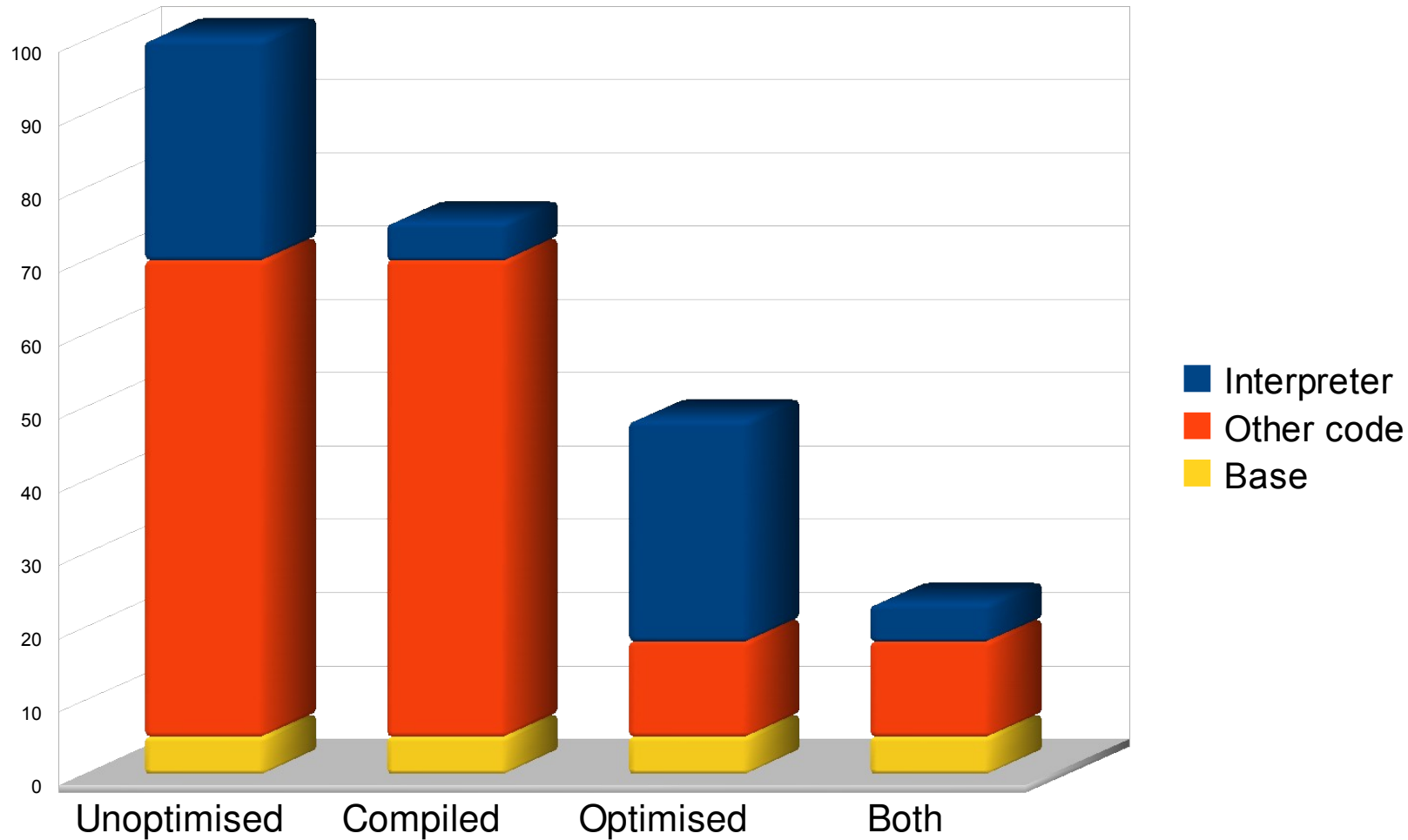
- Expected Speedups:
 - Compiler: $\times 6$
 - Optimiser: $\times 5$



Optimisation and Compilation(2)



University
of Glasgow

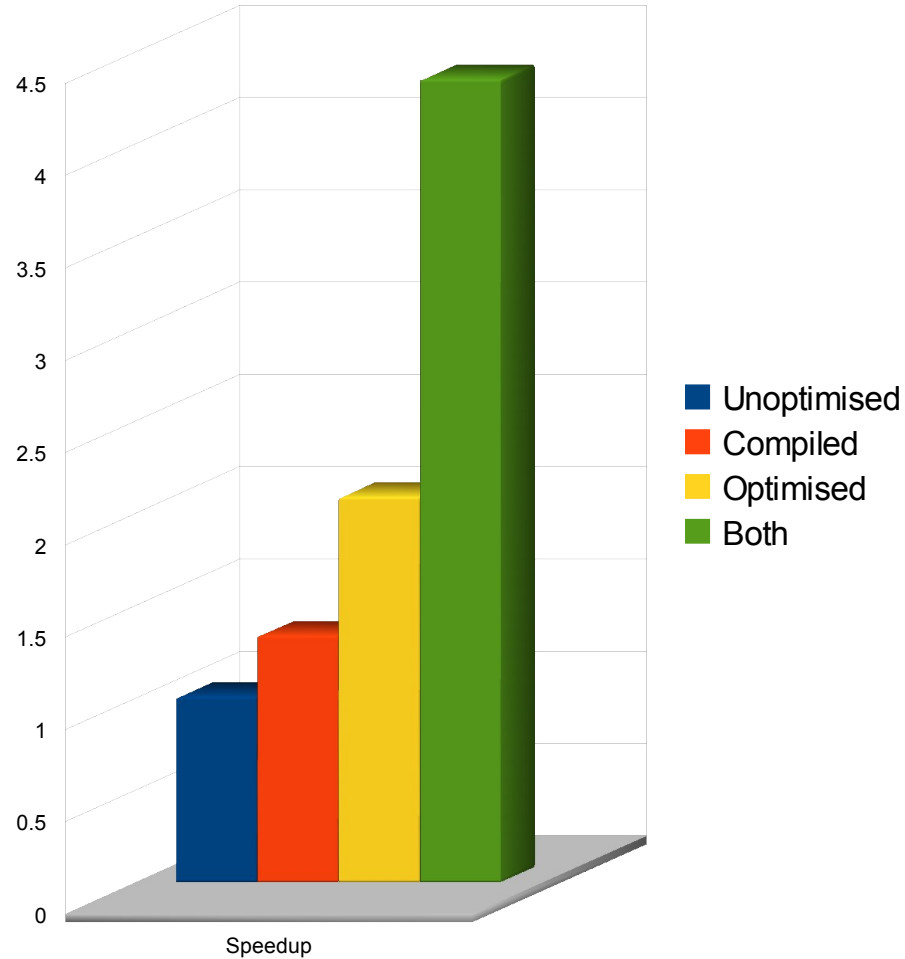




Optimisation and Compilation (3)



University
of Glasgow





Speculative Optimisations



University
of Glasgow

- Hypothesis (probably true)
- Transformation
- Guard(s)
 - Location
 - Action



Simple example



- $c = a + b$
- Hypothesis: a & b are integers
- Guard: Location: inlined, Action: $c = a + b$
- Transformation:

```
if a.__class__ is not int or b.__class__ is not int:
```

```
    c = a + b
```

```
else:
```

```
    c = int_add(a, b)
```



Guard



Example Optimisations



University
of Glasgow

- Attribute lookup
 - Fast slot lookup for expected types
- Operator
 - Fast dispatch for expected types
- Boolean
 - Optimising tests and branches
- Integer
 - Specialised versions of above for ints



Type profiling and inference



University
of Glasgow

- Very useful to know the types of variables
 - Its what makes statically typed languages fast
 - Can't know types for certain
- Type inference can infer a few types
- Type profiling can tell us probable types
 - Combining the two gives complete probabilistic type information
 - Can generate code that's almost as fast as if we knew the types statically



Slot attribute access

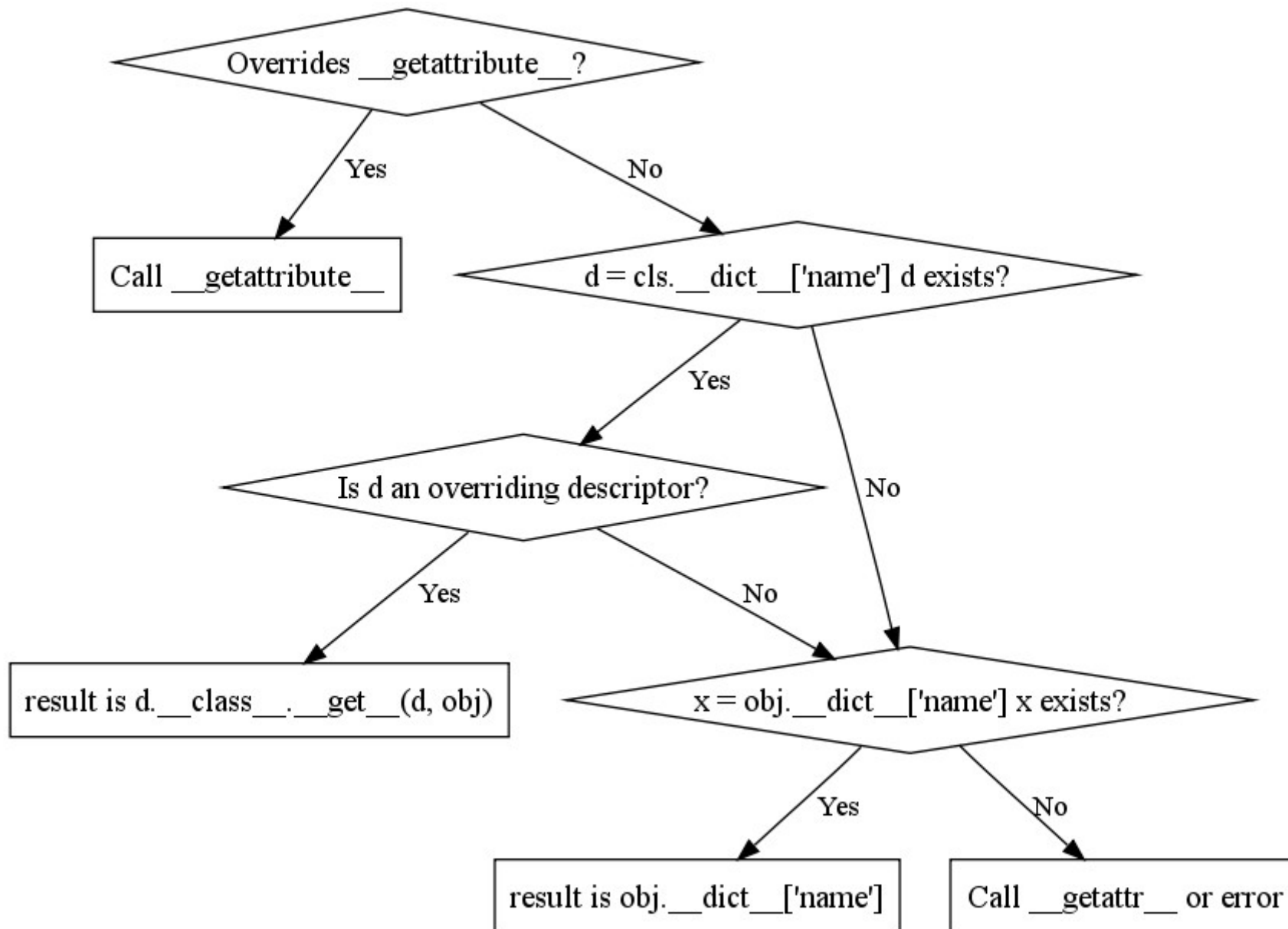


University
of Glasgow

- Example is the simple expression `x.name`
 - Fetch the attribute “name” in the object `x`
- In this example, assume that type profiling has told us that `x` will probably be of class `C`.
- Also `C.__dict__['name']` is `S`.
- `S` is a slot descriptor, i.e. `x.name` refers to a slot in object `x`.

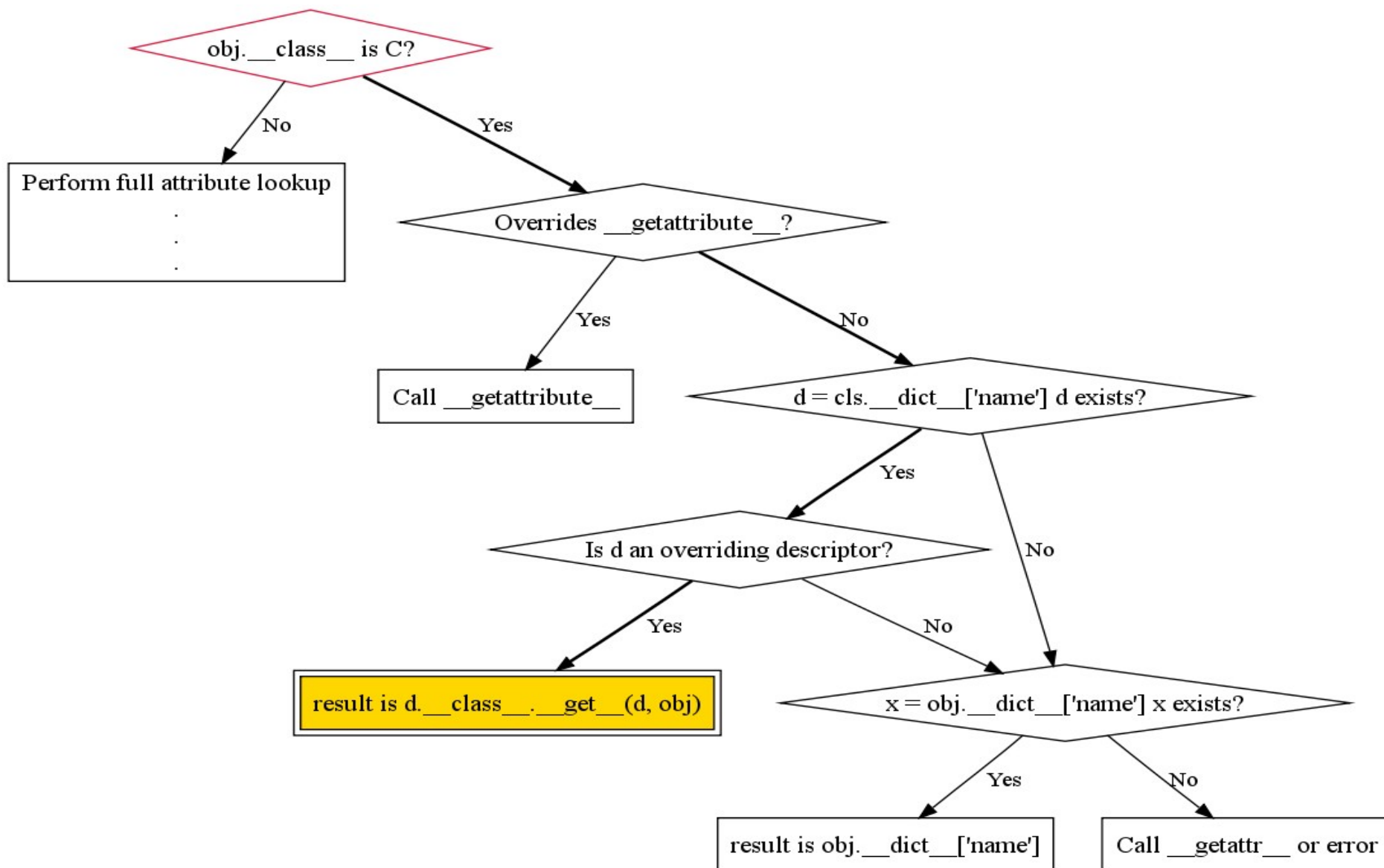


Python attribute look up





Add an inline guard





Out-of-line Guards

- A guard is a triple (code, class, name)
- Guards are formed into rings, one for each optimised code object
- Triggered before (class, name) is altered.
- When a guard is triggered, it does the following:
 - Marks a flag informing running optimised code to revert to the original.
 - Frees the optimised code
 - Frees itself and all other guards in the ring.



Add out-of-line guards

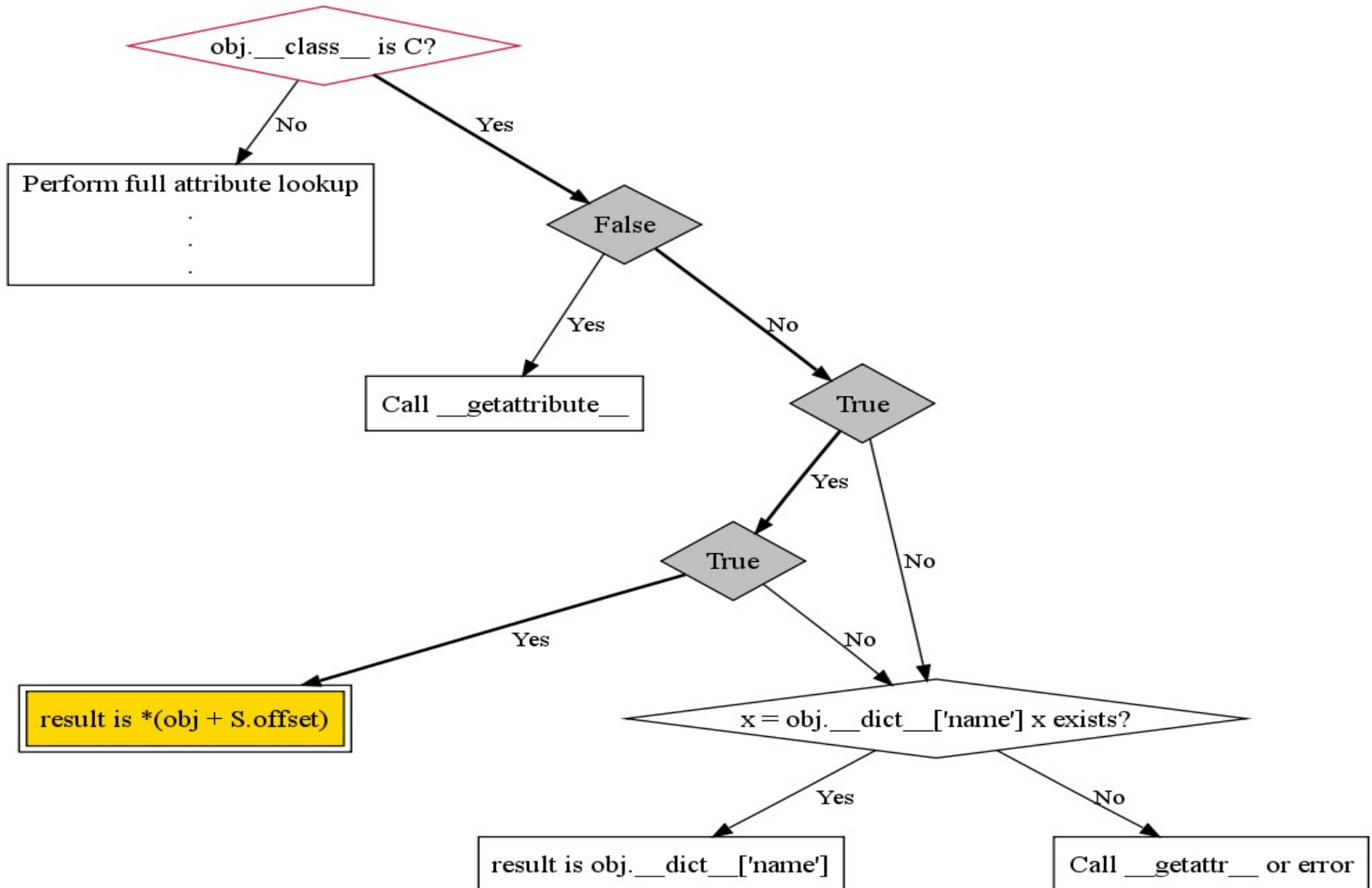


University
of Glasgow

- Add guards to C's dictionary
 - “`__getattr__`”
 - “name”
- Means that we can treat `C.__dict__['name']` as a constant, S.

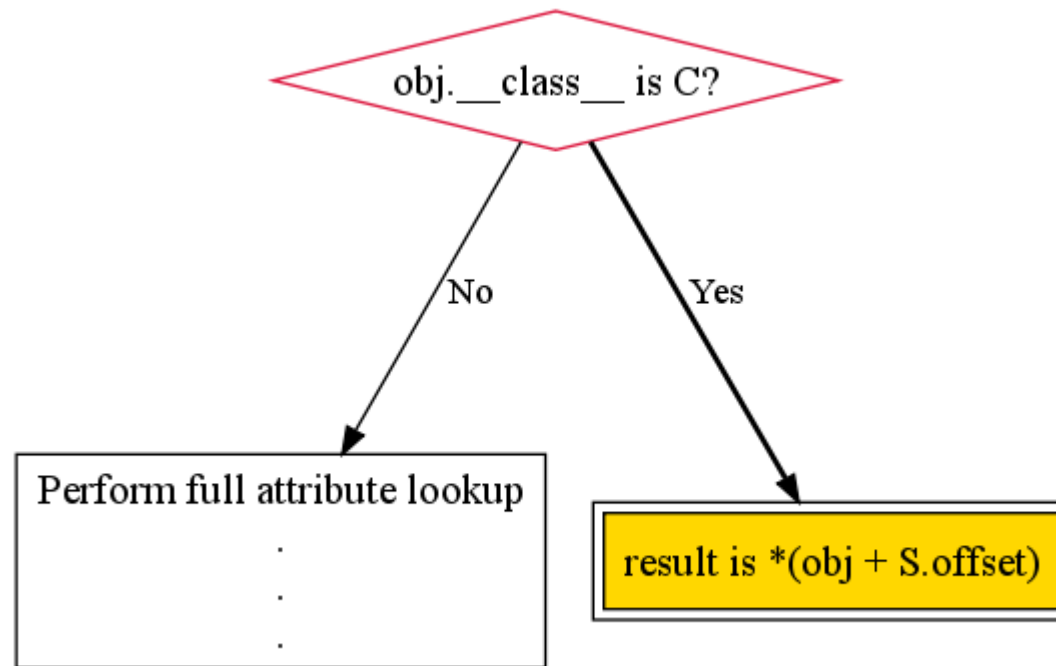


With all guards





Which becomes...





Current Optimiser



University
of Glasgow

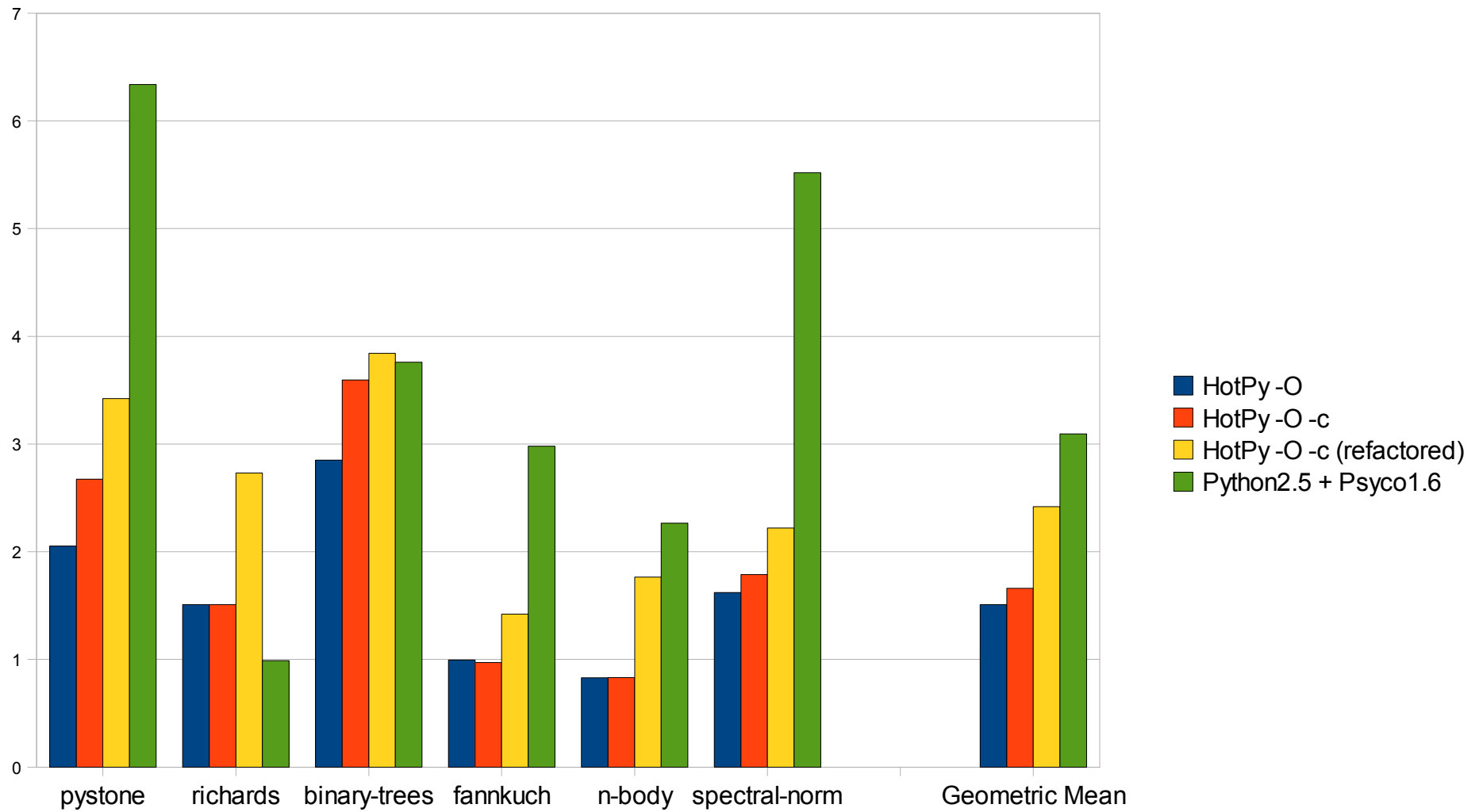
- Quite simple and naive.
- Keeps track of function execution count
- Once this reaches a threshold:
 - start type-profiling
- Once execution count reaches a second threshold:
 - Optimise (and optionally compile)



Benchmark results



University
of Glasgow





Future Work



University
of Glasgow

- Trace-based optimiser
- Reduced overhead
 - Reduced amount of code to compile
 - Reduced memory use
- More effective
 - Can track types through function calls
 - Reduced overhead in checking for deoptimisation
 - Expect to be at least 10% faster than refactored benchmarks on unmodified benchmarks.



Help needed



University
of Glasgow

- If HotPy is ever to be useful then your help is required
- I need to write up my PhD
 - Limits the amount of time I can spend on HotPy
- Money!
 - Fund continuing working on HotPy



Any questions?



University
of Glasgow

- Thanks for listening
- Any questions?
- <http://www.dcs.gla.ac.uk/~marks>
- <http://gvmt.sourceforge.net>