Starkiller: A Static Type Inferencer and Compiler for Python

Michael Salib
msalib@alum.mit.edu
EECS Masterworks
April 28, 2004

The problem

- Software sucks
  - too buggy, too dangerous
  - too expensive and slow to build
  - too pervasive
  - internet makes it all worse
  - bad software kills people
  - it is going to get worse before it gets better

The solution

- Use a High level language
  - fewer lines of code needed
  - fewer lines mean
    - fewer bugs
    - less time/money to build
  - make the worst brain damage impossible
    - no buffer overflows in Perl/Java/Python programs
- HLLs have their own problems
  - surrounded by sinister death cults
  - slow
  - weird and different...must be evil

Nobody expects the spanish inquisition!

- I like Python
  - the death cult isn't as sinister
  - not that weird
  - but still slow
- Thus we have a new problem: make Python fast!

The (new) problem: why Python is slow

- Very dynamic: no compile time info
- No declarations
  - types are latent
  - function/class definitions occur at runtime
  - attributes can be added anytime by anyone
- Function and data polymorphism
- Methods can be replaced anytime
- Run time type checks everywhere
- Dynamic inheritance and class membership
- Dynamic dispatch and binding
- eval and dynamic module loading

Making Python fast

- Apply last 30 years of optimization research
- Run time code choices actively foil optimization
  - dynamic dispatch causes same problem for Java and C++
- Starkiller is a compiler: Python to C++
- Reject programs containing eval or dynamic module loading
- Type inferencer statically resolves most uses of dynamic dispatch and binding

Cutting open the heart

- Model program as a constraint network
- Nodes are variables
  - each node has a set of types it may achieve at runtime
- Constraints connect nodes
  - model data flow in the program
- Types flow along constraints
  - constraints enforce a subset relationship
- Initially, all type sets are empty
  - except constants
- Types propogate through the network until it stabilizes

Three problems Starkiller solves

- Parametric polymorphism
  - max(1, 2) and max(3.14, 2.78) cause loss of precision
- Data polymorphism
  - bob.age = 15
  - bob.age = 15.4
  - bob.age = TimeDelta(15, 0, 0)
- Foreign code
  - Python is sucessful because it interoperates well with large C/C++/Fortran code bases

Wrapup

- Results
  - Type inferencer is mostly done
  - Compiler is very very young
  - Factor of 60 speedup on short numeric pro-grams
- Acknowledgements
  - Jonathan Bachrach, Howie Shrobe, Greg Sulli-van and everyone at Dynlangs
  - DARPA and all my financial benefactors, the American taxpayers