Design and Evaluation of Gradual Typing for Python

Michael M. Vitousek, Andrew M. Kent, Jeremy G. Siek, and Jim Baker*
Indiana University Bloomington *Rackspace, Inc

Reticulated Python is a platform for experimenting with **gradual typing** in an existing dynamic language. It is implemented as a **source to source translator** which performs static typechecking and **inserts casts**. Reticulated Python has **three approaches to casts on mutable objects**: the traditional approach using **proxies** (based on Herman et al., 2007), the transient approach which uses pervasive **use-site checks** to ensure type-safety, and the monotonic approach which attaches **runtime type information** to objects and ensures that their runtime types are at least as precise as all references to them. We performed several **case studies** of annotating existing Python programs and running them with Reticulated.

Case studies:

- **CherryPy**
  - OO Web application framework
  - Annotated APIs
- **`stats.py` Statistics library**
  - Math, lists, eval
  - Annotated throughout
- **SlowSHA**
  - Objects and math
  - Annotated throughout

Dynamic semantics of mutable objects

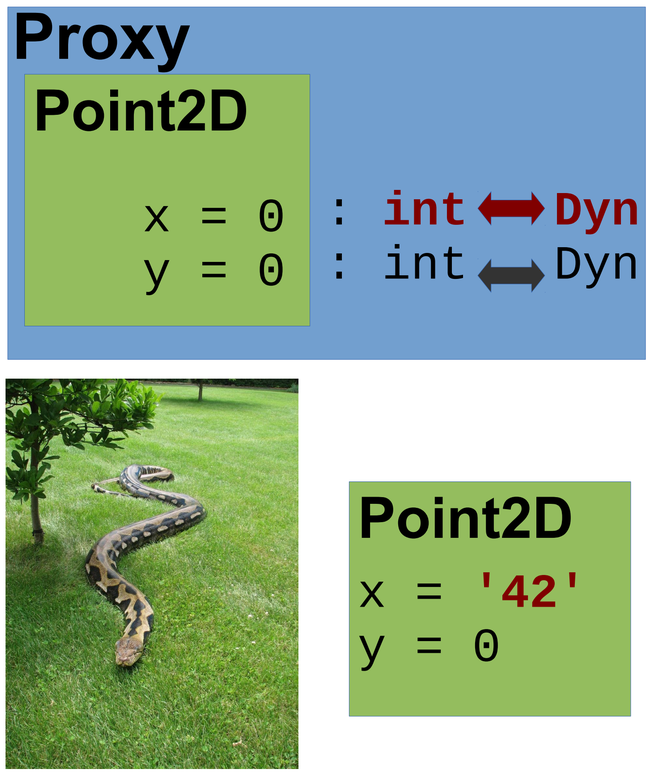
```
@fields({'x': int, 'y': int})
class Point2D:
  x = 0
  y = 0

def bad_update(pt):
  pt.x = '42'
def update_x(pt:Point2D)->int:
  bad_update(pt)
  return pt.x
update_and_return_x(Point2D())
```

- Type `Point2D ≈ Object({'x':int,'y':int})`
- No static type error
- If no runtime checks occur, bad_update returns a string with type int (type error!)
- Three solutions for checks

Guarded semantics

Proxies perform runtime checks, breaks pointer equality

1) `bad_update(pt):` Proxy installed on `pt`
2) `pt.x = '42':` Proxy casts '42' to int, causes runtime cast error



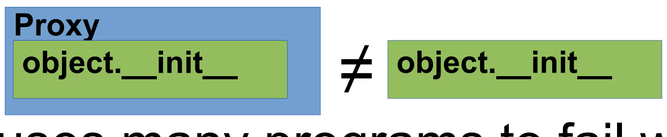
Transient semantics

Inserts pervasive use-site checks in program, no proxies

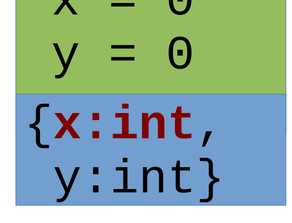
1) `bad_update(pt):` No proxy
2) `pt.x = '42':` Succeeds, no error
3) `return pt.x:` Use-site check fails (since pt.x not an int)

Object identity is a problem with **guarded**

- Proxies do not preserve pointer equality with underlying object



- Causes many programs to fail with guarded



Monotonic semantics

RTTI which becomes monotonically stronger; no object proxies but rejects more programs

1) `Point2D():` RTTI on object set to `{'x':int,'y':int}`
2) `pt.x = '42':` Object casts '42' to int, causes runtime cast error

Check it out! `github.com/mvitousek/reticulated`