A case for:

Sound Gradual Typing

Ben Greenman @ Northeastern University

Goal: make gradual typing fast

Goal: make gradual typing fast (or at least usable)

"The end-product appears to be a 50%
performance hybrid due to boundary contracts"

"So far Typed Quad is running about 10x
slower than regular"

"From 1 ms to 12 seconds ...
I feel like I got a bit burned here"

If sound then slow

If sound then slow

Why bother?

Unsound Types are Dangerous!

```
# untyped

def addVotes(n):
    assert!(n >= 0)
    this.numVotes += n
```

Unsound Types are Dangerous!

```
# untyped

def addVotes(n):
    assert!(n >= 0)
    this.numVotes += n
```

```
# typed, but unsound

def addVotes(n : Natural):
    assert!(n >= 0)
    this.numVotes += n
```

Unsound Types are Dangerous!

```
# untyped

def addVotes(n):
    assert!(n >= 0)
    this.numVotes += n
```

```
# typed, but unsound

def addVotes(n : Natural):
    assert!(n >= 0)
    this.numVotes += n
```

Unsound Types are Dangerous!

```
# untyped

def addVotes(n):
    assert!(n >= 0)
    this.numVotes += n
```

```
# typed, but unsound

def addVotes(n : Natural):
    assert!(n >= 0)
    this.numVotes += n
```

Unsound Types are Dangerous!

```
# untyped

def addVotes(n):
    assert!(n >= 0)
    this.numVotes += n
```

```
# typed, but unsound

def addVotes(n : Natural):
    assert!(n >= 0)
    this.numVotes += n
```

addVotes(-1)

Unsound Types are Dangerous!

Unsound Types are Dangerous!

```
def getWords(str : String):
    return str.split()
```

Unsound Types are Dangerous!

```
def getWords(str : String):
    return str.split()
```

```
class Atom:
    def split():
        ....
```

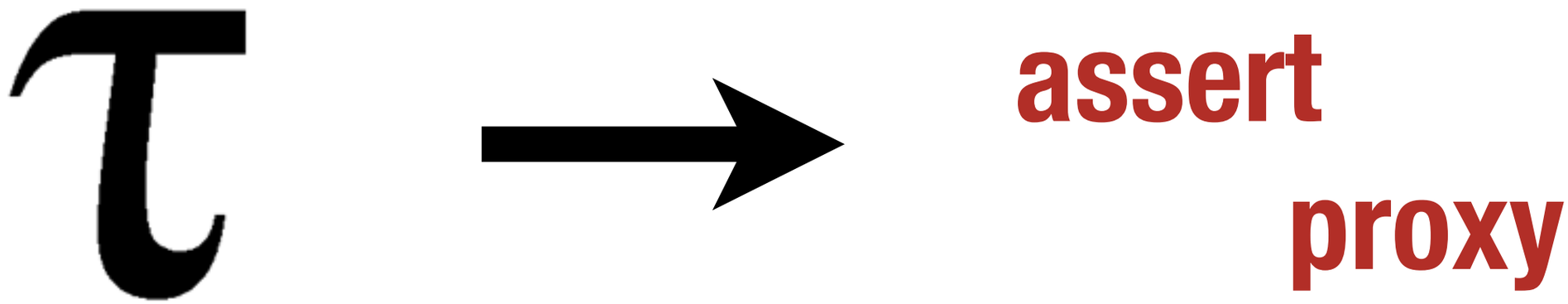
Unsound Types are Dangerous!

```
def getWords(str : String):
    return str.split()
```

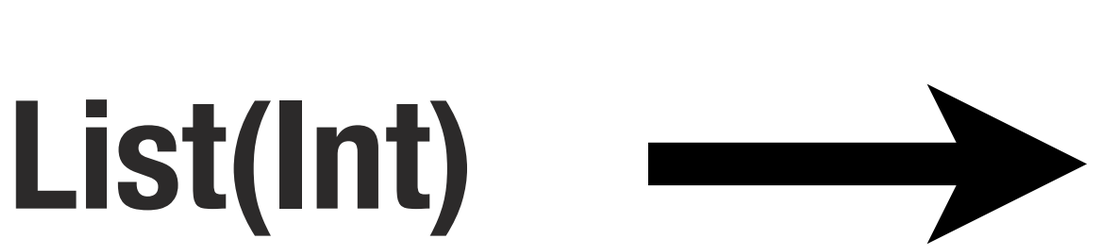
```
class Atom:
    def split():
        ....
```

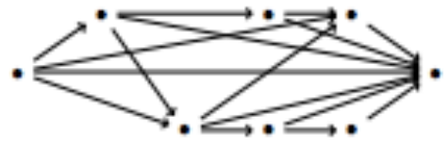
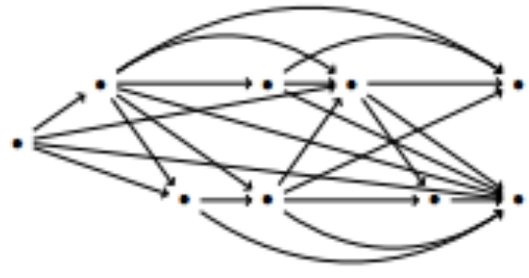
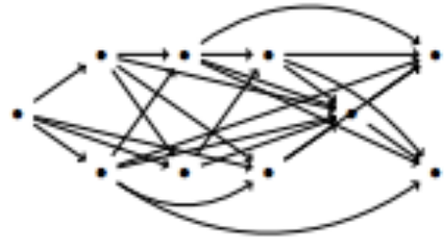
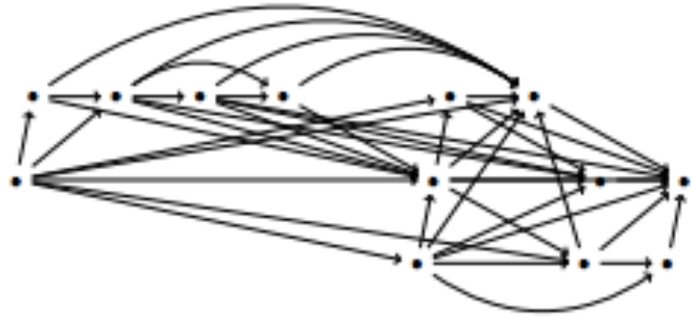
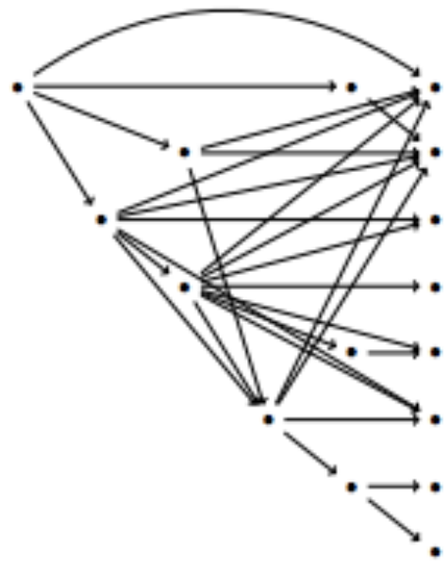
```
uranium = Atom()

getWords(uranium)
```

Soundness Strategy

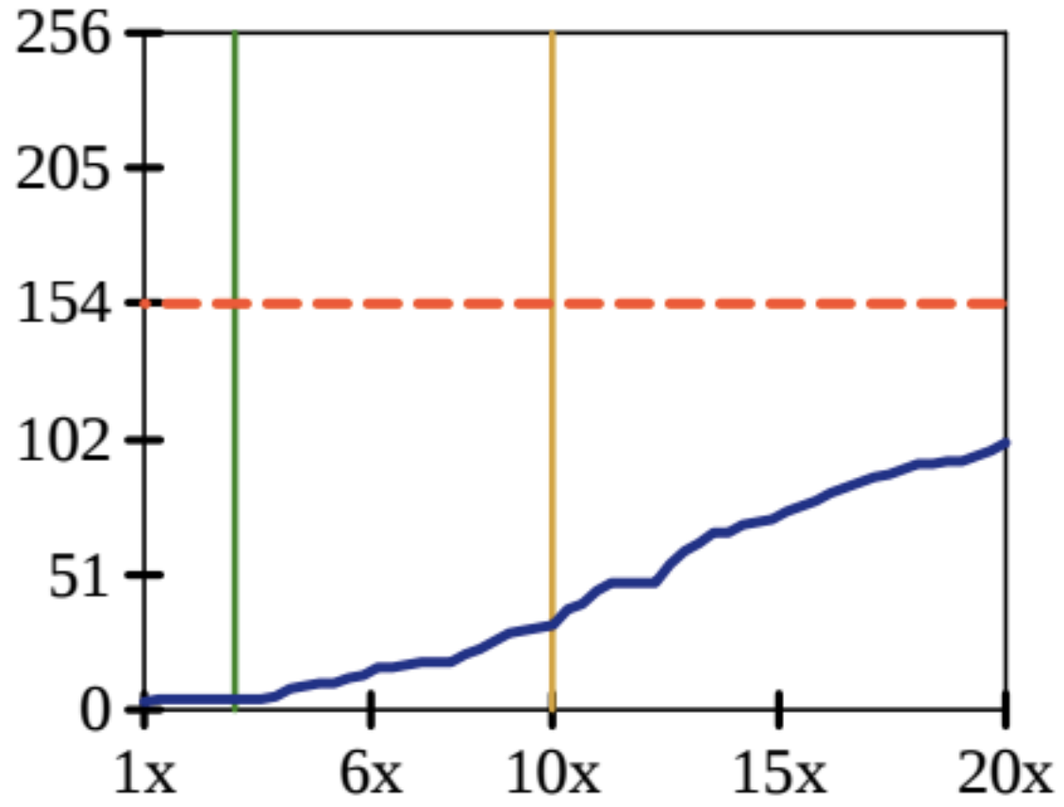


Soundness Strategy

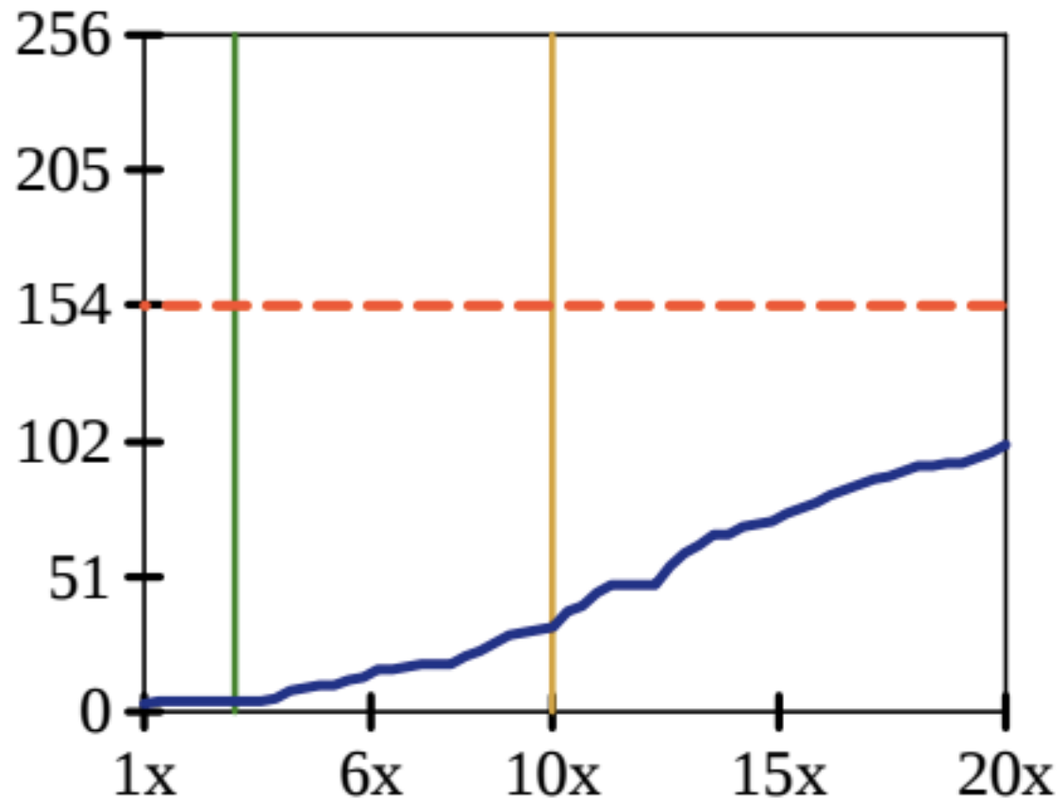


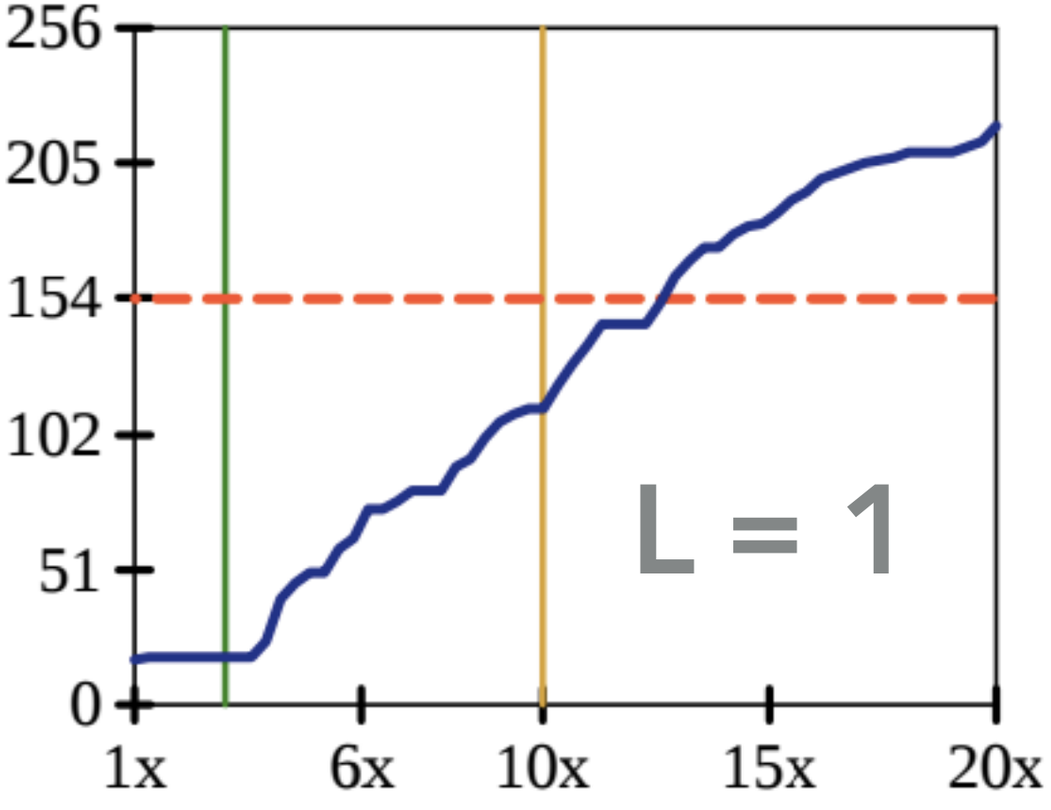
Project name	Modules	Untyped LOC	Module structure	Mean Overhead	Max Overhead
snake	8	161		32x	121x
tetris	9	305		33x	117x
synth	10	837		40x	86x
gregor	13	996		3x	5x
quad	16	6722		31x	56x

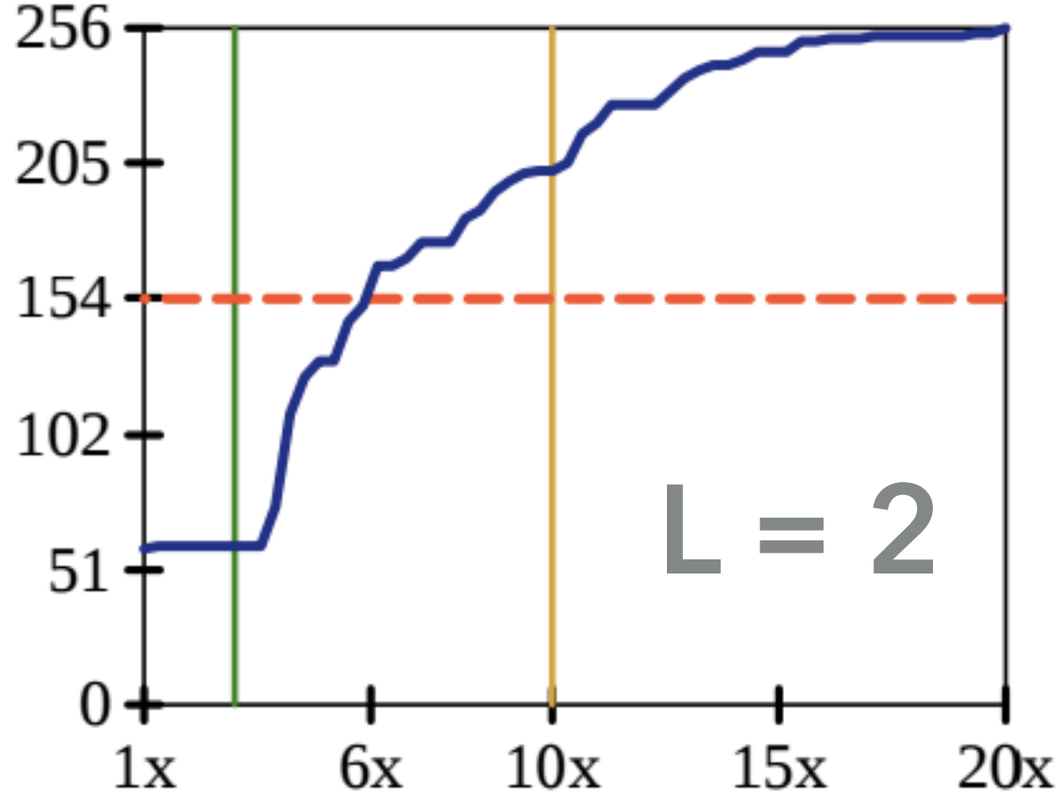
snake	(8 modules)
typed/untyped ratio	0.92x
max. overhead	121.51x
mean overhead	32.30x
3-deliverable	4 (2%)
3/10-usable	28 (11%)



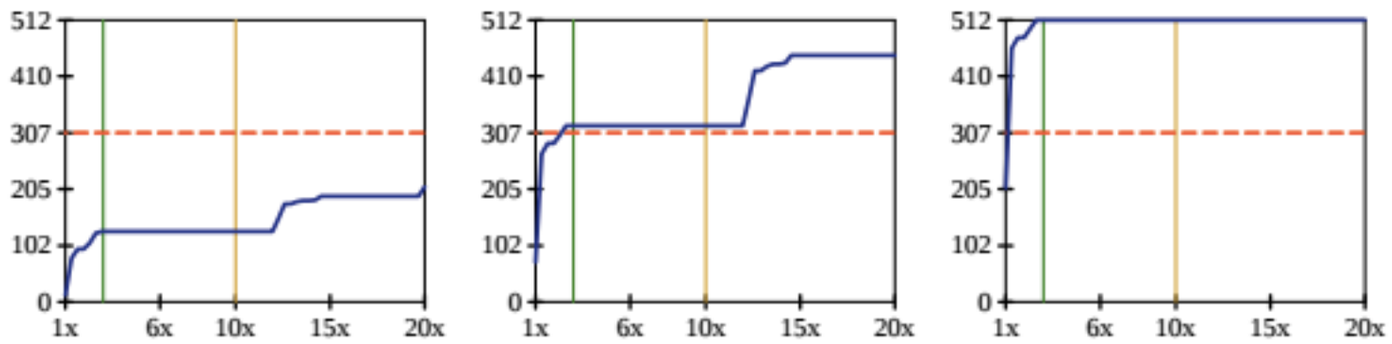
snake	(8 modules)
typed/untyped ratio	0.92x
max. overhead	121.51x
mean overhead	32.30x
3-deliverable	4 (2%)
3/10-usable	28 (11%)



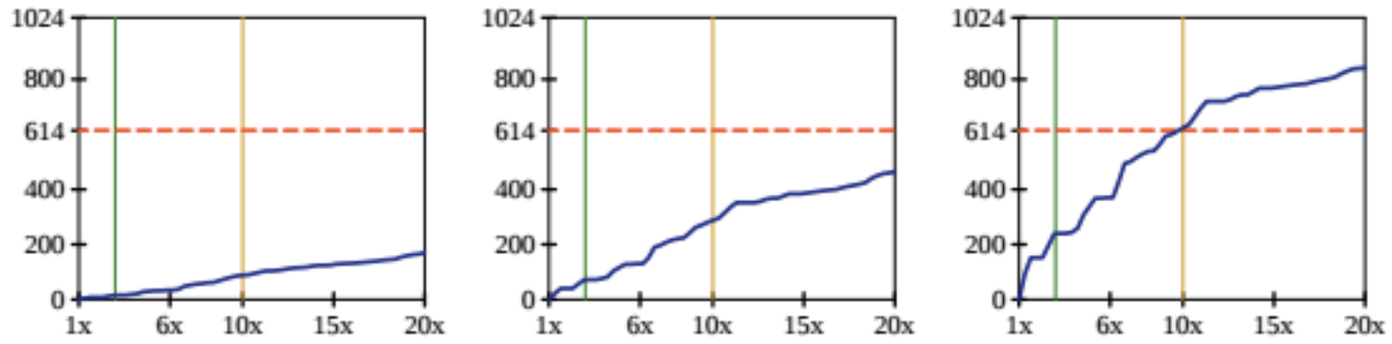




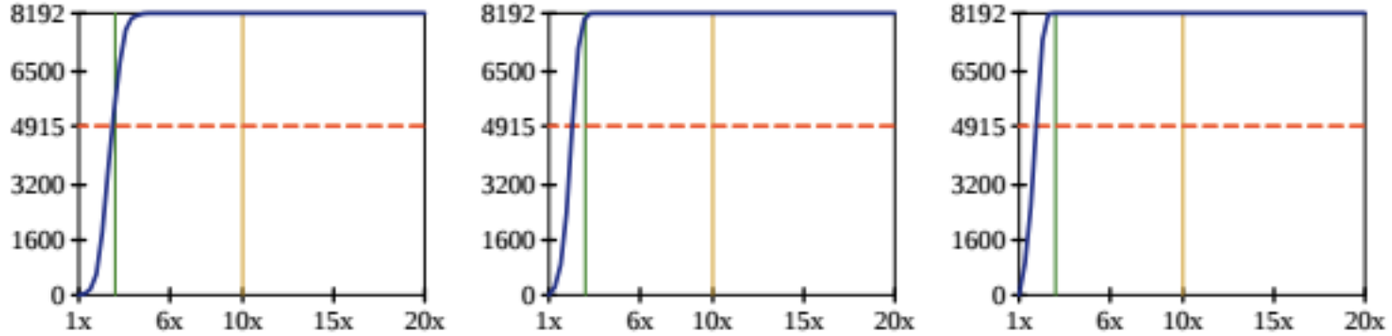
tetris	(9 modules)
typed/untyped ratio	0.97x
max. overhead	117.28x
mean overhead	33.34x
3-deliverable	128 (25%)
3/10-usable	0 (0%)



synth	(10 modules)
typed/untyped ratio	1.03x
max. overhead	85.90x
mean overhead	39.69x
3-deliverable	15 (1%)
3/10-usable	73 (7%)



gregor	(13 modules)
typed/untyped ratio	1.22x
max. overhead	4.72x
mean overhead	2.72x
3-deliverable	5644 (69%)
3/10-usable	2548 (31%)



quad	(16 modules)
typed/untyped ratio	13.34x
max. overhead	56.43x
mean overhead	31.50x
3-deliverable	2046 (3%)
3/10-usable	5637 (9%)