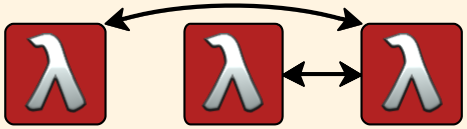
Three Approaches to Gradual Typing

Ben Greenman, Justin Pombrio, Matthias Felleisen, Preston Tunnell Wilson, Shriram Krishnamurthi, and many others

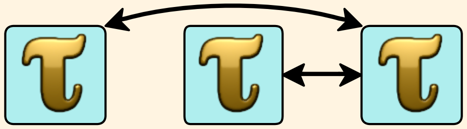
Dynamic Typing

value-level abstractions, enforced at run-time



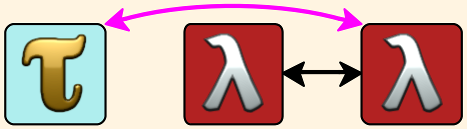
Static Typing

type-level abstractions, checked before run-time



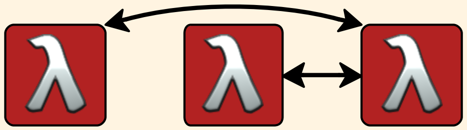
Gradual Typing

mix of static & dynamic typing ... somehow



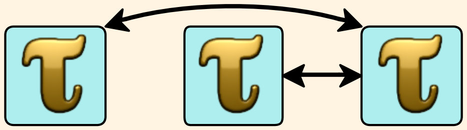
Dynamic Typing

value-level abstractions,
enforced at run-time



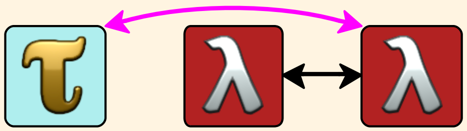
Static Typing

type-level abstractions,
checked before run-time



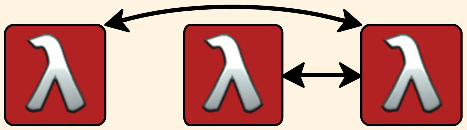
Gradual Typing

mix of static & dynamic
typing ... somehow



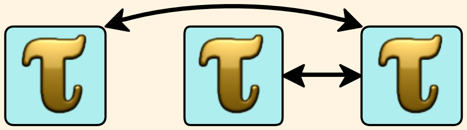
Dynamic Typing

value-level abstractions, enforced at run-time



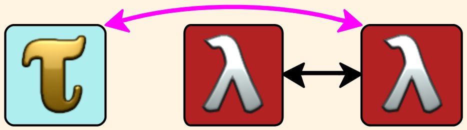
Static Typing

type-level abstractions, checked before run-time



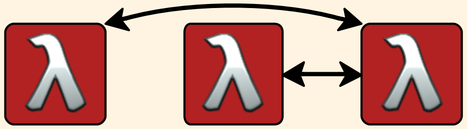
Gradual Typing

mix of static & dynamic typing ... somehow



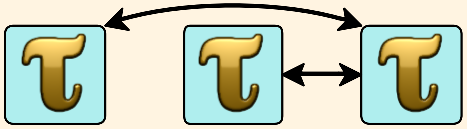
Dynamic Typing

value-level abstractions, enforced at run-time



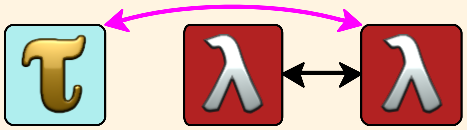
Static Typing

type-level abstractions, checked before run-time



Gradual Typing

mix of static & dynamic typing ... somehow



Gradual Typing is growing ...

Over 80 publications

Over 20 implementations

Gradual Typing is growing ...

Over 80 publications

Over 20 implementations

But NO common definition of gradual typing – due to **different** goals and priorities

Gradual Typing is growing ...

Over 80 publications

Over 20 implementations

But NO common definition of gradual typing – due to **different** goals and priorities

Little acknowledgment (or analysis!) of the differences

One kind of gradual typing:
Migratory Typing (SNAPL'17)

1. Begin with an existing, dynamically-typed language

2. Design an idiomatic type system

3. Allow interaction between the two languages

a few Migratory Typing systems

Gradualtalk Typed Racket TPD Pycket

Pallene Grace SafeTS Reticulated

mypy Flow Hack Pyre Pytype rtc MACLISP
Common Lisp Strongtalk TypeScript
Typed Clojure Typed Lua

A few Migratory Typing systems

Deep

Gradualtalk Typed Racket TPD Pycket

Shallow

Pallene Grace SafeTS Reticulated

Erasure

mypy Flow Hack Pyre Pytype rtc MACLISP

Common Lisp Strongtalk TypeScript

Typed Clojure Typed Lua

three approaches to Migratory Typing

Deep

Shallow

Erasure

three approaches to Migratory Typing

Deep (behavioral)

Shallow (transient)

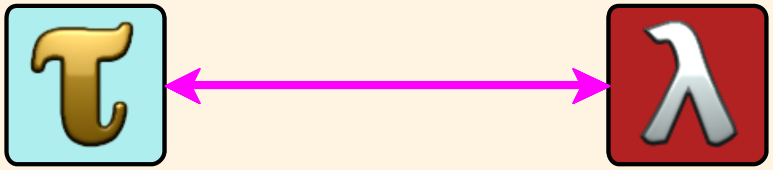
Erasure (optional)

three approaches to Migratory Typing

Deep (behavioral)

Shallow (transient)

Erasure (optional)



Three strategies for enforcing types at a boundary

three approaches to Migratory Typing

Deep

Shallow

Erasure

three approaches to Migratory Typing

Deep

types are sound/enforced

Shallow

Erasure

three approaches to Migratory Typing

Deep

types are sound/enforced

Shallow

typed code cannot get stuck

Erasure

three approaches to Migratory Typing

Deep	types are sound/enforced
Shallow	typed code cannot get stuck
Erasure	types do not affect behavior

Deep

Shallow

Erasure

Deep

Shallow

Erasure

Type Soundness (simplified):

if ⊢e:t then either:

- e ->* v and ⊢v:t
- e diverges
- e ->* Error

Deep

Shallow

Erasure

Type Soundness (simplified):

if $\vdash e:t$ then either:

- $e \to^* v$ and $\vdash v:t$
- e diverges
- $e \to^*$ Error

Deep

Shallow

Erasure

Type Soundness (simplified):

if ⊢e:t then either:

- e ->* v and ⊢v:t
- e diverges
- e ->* Error

Deep

Shallow

Erasure

Type Soundness (simplified):

if ⊢e:t then either:

- e ->* v and ⊢v:t
- e diverges
- e ->* Error

Deep

Shallow

Erasure

Deep

if ⊢e:t then either:

- e ->* v and ⊢v:t
- e diverges
- e ->* Error

Deep

Shallow

Erasure

Deep

if ⊢e:t then either:

- e ->* v and $\underline{\vdash v:t}$
- e diverges
- e ->* Error

Shallow

if ⊢e:t then either:

- e ->* v and $\underline{\vdash v:C(t)}$
- e diverges
- e ->* Error

Deep

Shallow

Erasure

Deep

if ⊢e:t then either:

- e ->* v and $\underline{\vdash v:t}$
- e diverges
- e ->* Error

Shallow

if ⊢e:t then either:

- e ->* v and $\underline{\vdash v:C(}t)$
- e diverges
- e ->* Error

Erasure

if ⊢e:t then either:

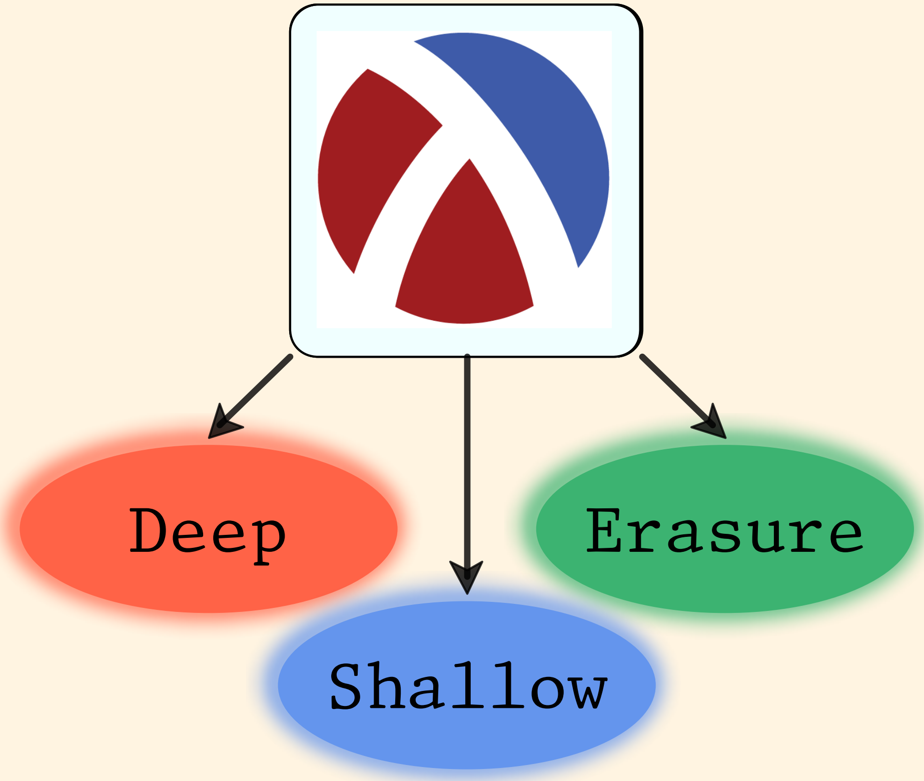
- e ->* v and $\underline{\vdash v\ }$
- e diverges
- e ->* Error

Is type soundness all-or-nothing?

Is type soundness all-or-nothing?

No! (in a mixed-typed language)

Implementation



Three compilers for the Typed Racket surface language

i.e. three ways of running **the same code**

How to measure performance?

ττττ

τττλ　ττλτ　τλττ　λτττ

τλτλ　λττλ　τλλτ　λτλτ

τλλλ　λτλλ　λλτλ　λλλτ

λλλλ

529 ms
τ τ τ τ

602 ms
τ τ τ λ

829,048 ms
τ τ λ τ

821,285 ms
τ λ τ τ

891 ms
λ τ τ τ

829,779 ms
τ λ τ λ

963 ms
λ τ τ λ

575 ms
τ λ λ τ

711,000 ms
λ τ λ τ

7

592 ms
τ λ λ λ

709,770 ms
λ τ λ λ

716,637 ms
λ λ τ λ

560 ms
λ λ λ τ

548 ms
λ λ λ λ

τ τ τ τ

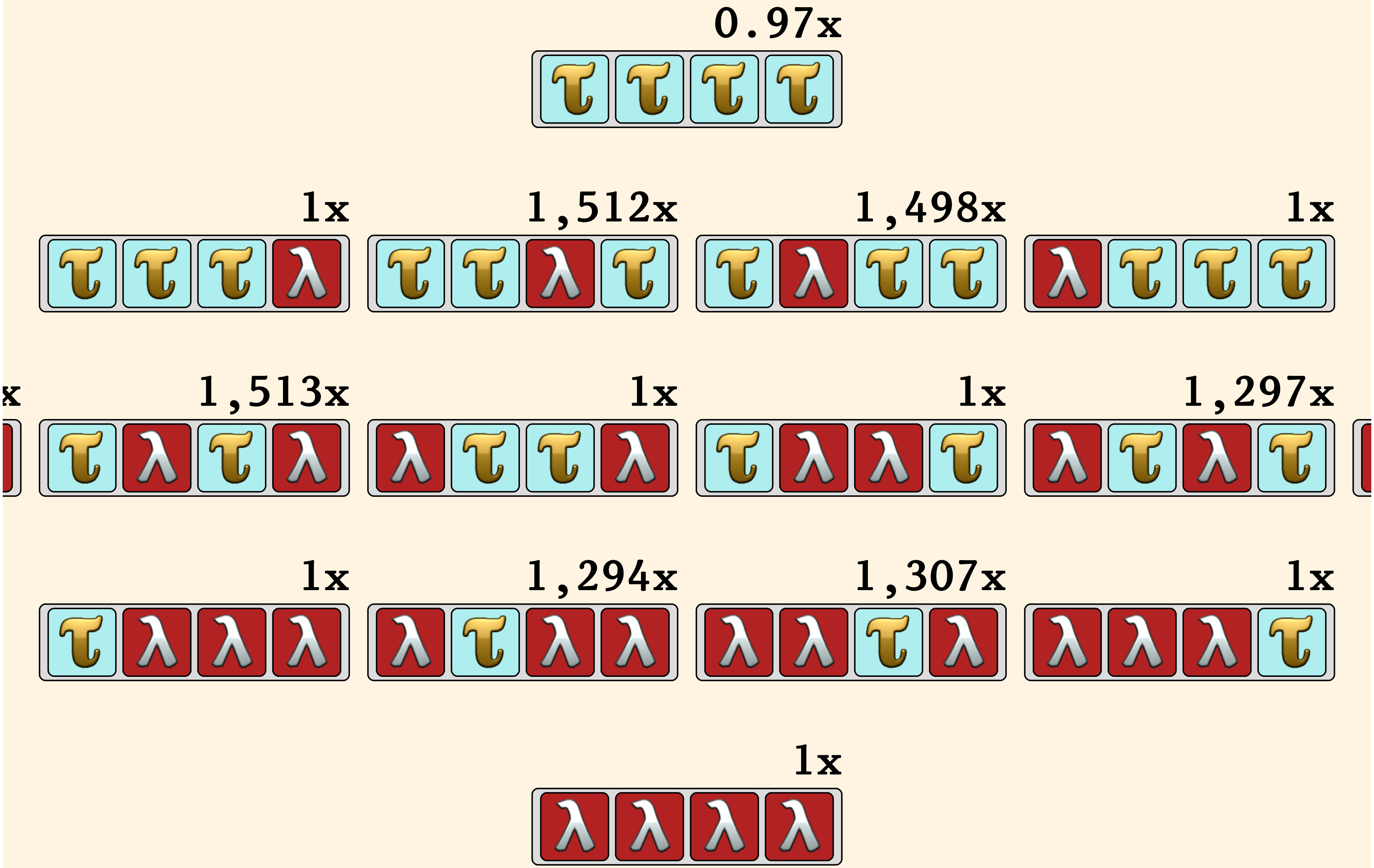
τ τ τ λ   τ τ λ τ   τ λ τ τ   λ τ τ τ

τ λ τ λ   λ τ τ λ   τ λ λ τ   λ τ λ τ

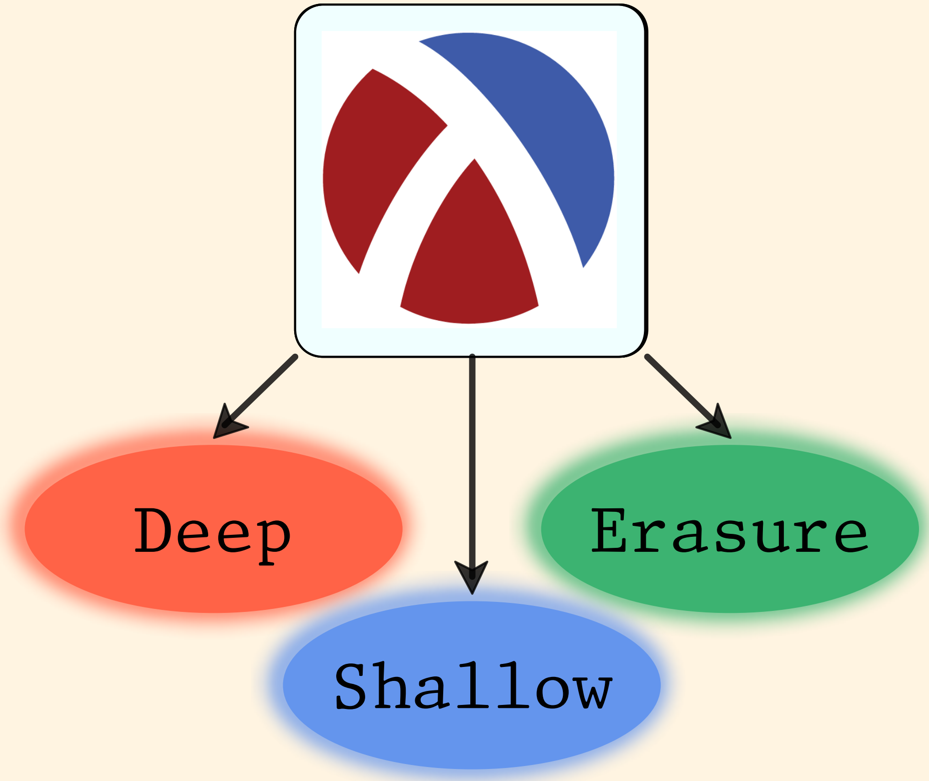
τ λ λ λ   λ τ λ λ   λ λ τ λ   λ λ λ τ

1x

λ λ λ λ


0.97x
ττττ
1x
τττλ
1,512x
ττλτ
1,498x
τλττ
1x
λτττ
x
1,513x
τλτλ
1x
λττλ
1x
τλλτ
1,297x
λτλτ
1x
τλλλ
1,294x
λτλλ
1,307x
λλτλ
1x
λλλτ
1x
λλλλ

Experiment



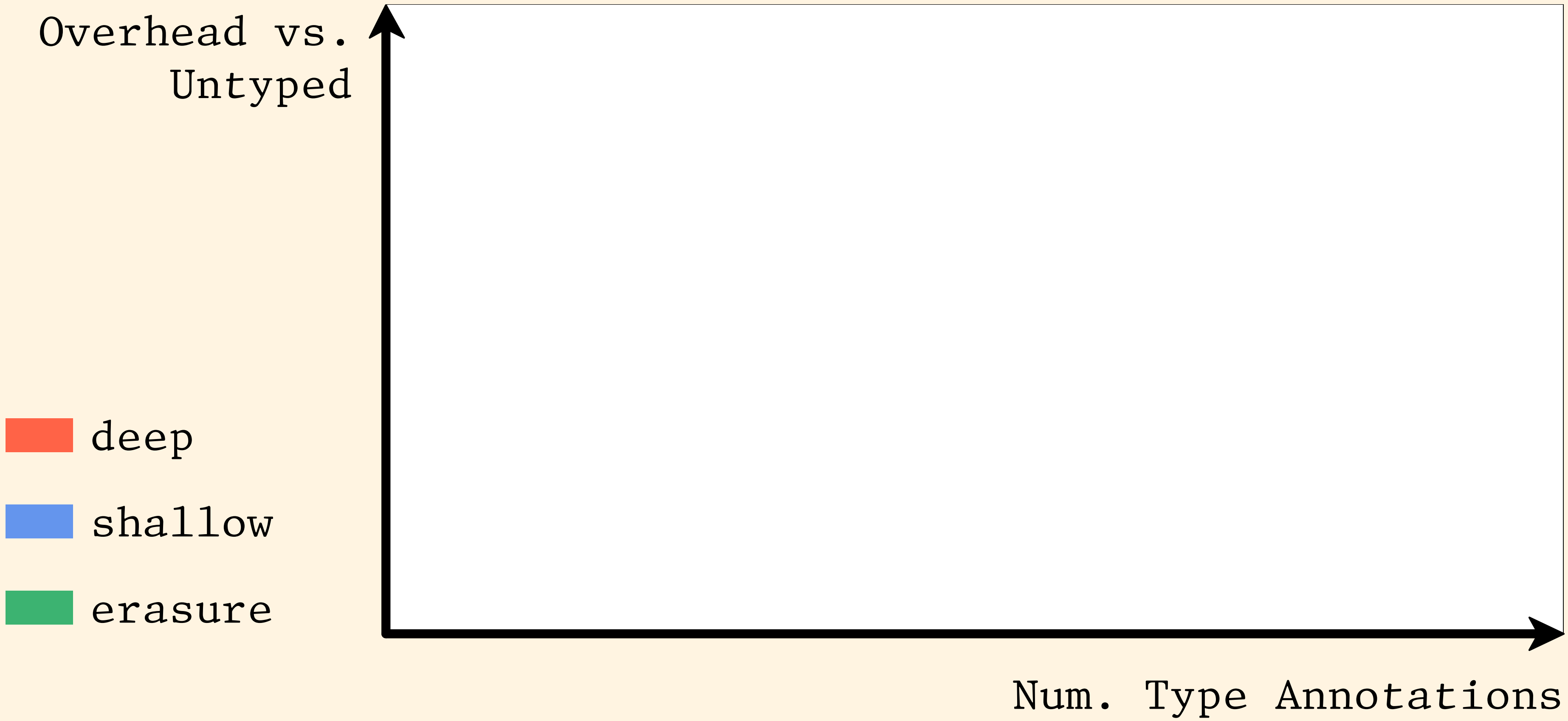
10 benchmark programs

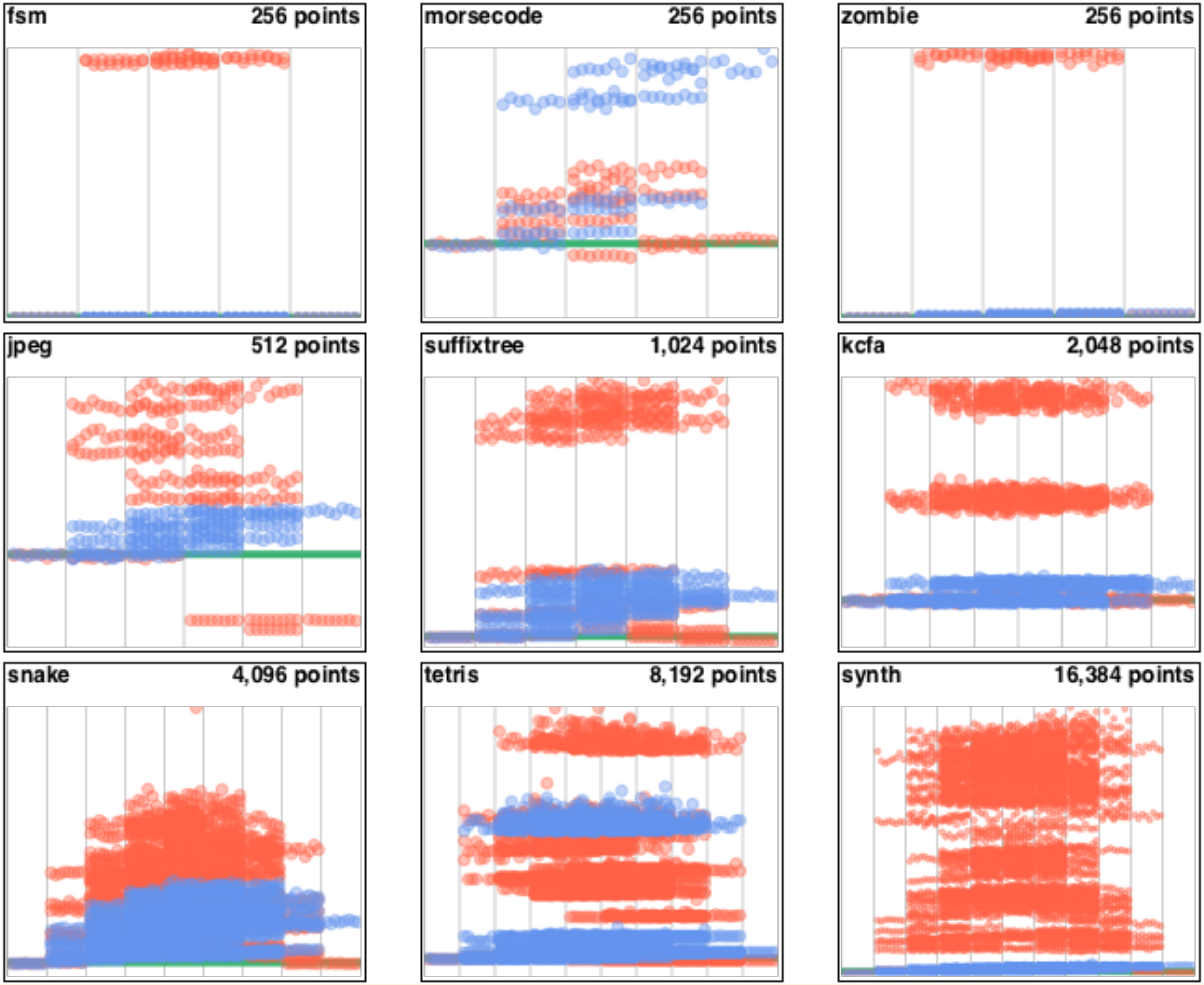
2 to 10 modules each

4 to 1024 configurations each

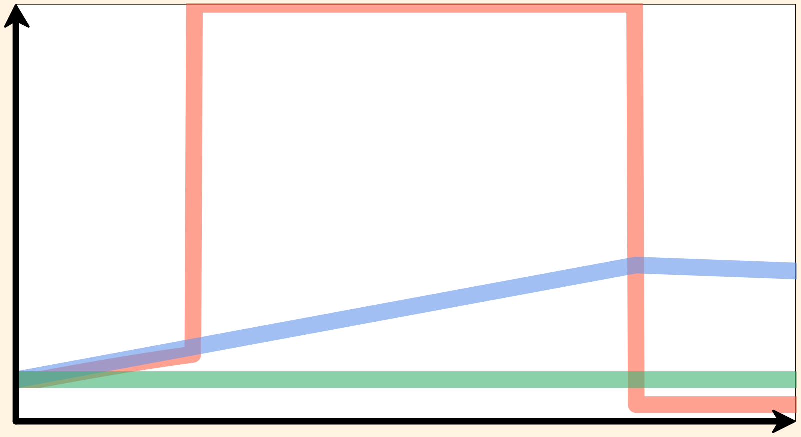
docs.racket-lang.org/gtp-benchmarks

Performance

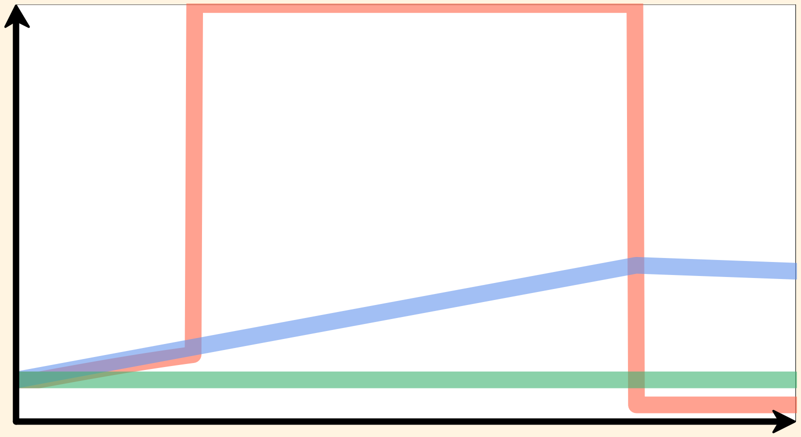



fsm
256 points
morsecode
256 points
zombie
256 points
jpeg
512 points
suffixtree
1,024 points
kcfa
2,048 points
snake
4,096 points
tetris
8,192 points
synth
16,384 points

Performance Implications



Performance Implications



🟥 add types to remove all critical boundaries

🟦 add types sparingly

🟩 add types anywhere, doesn't matter

three approaches to Migratory Typing

Deep

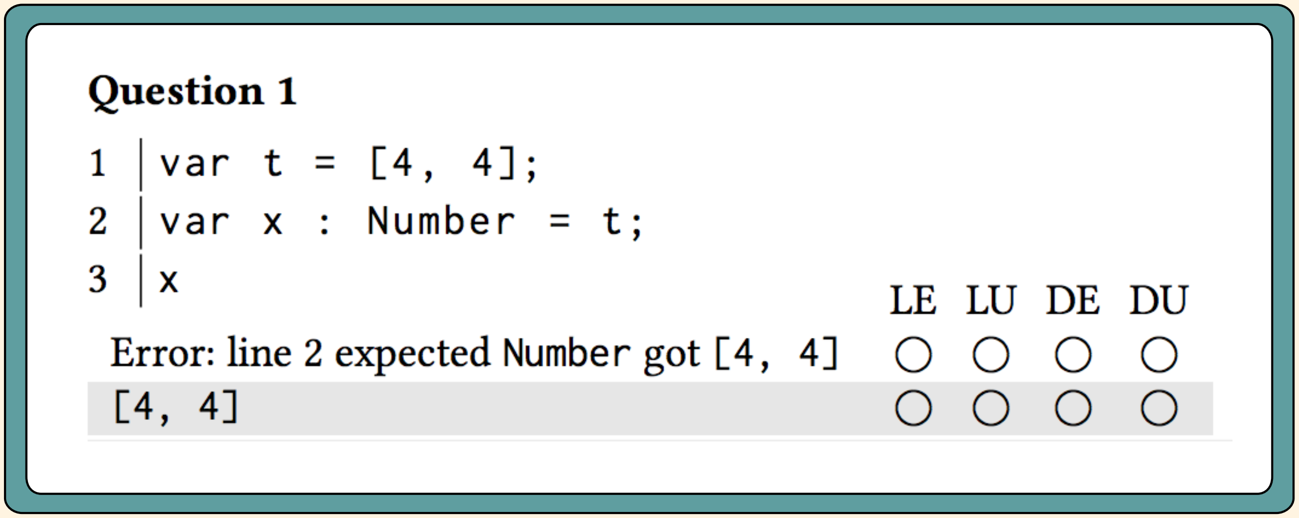
Shallow

Erasure

Soundness

Performance

... Users?



Question 1

```
1 | var t = [4, 4];
2 | var x : Number = t;
3 | x
```

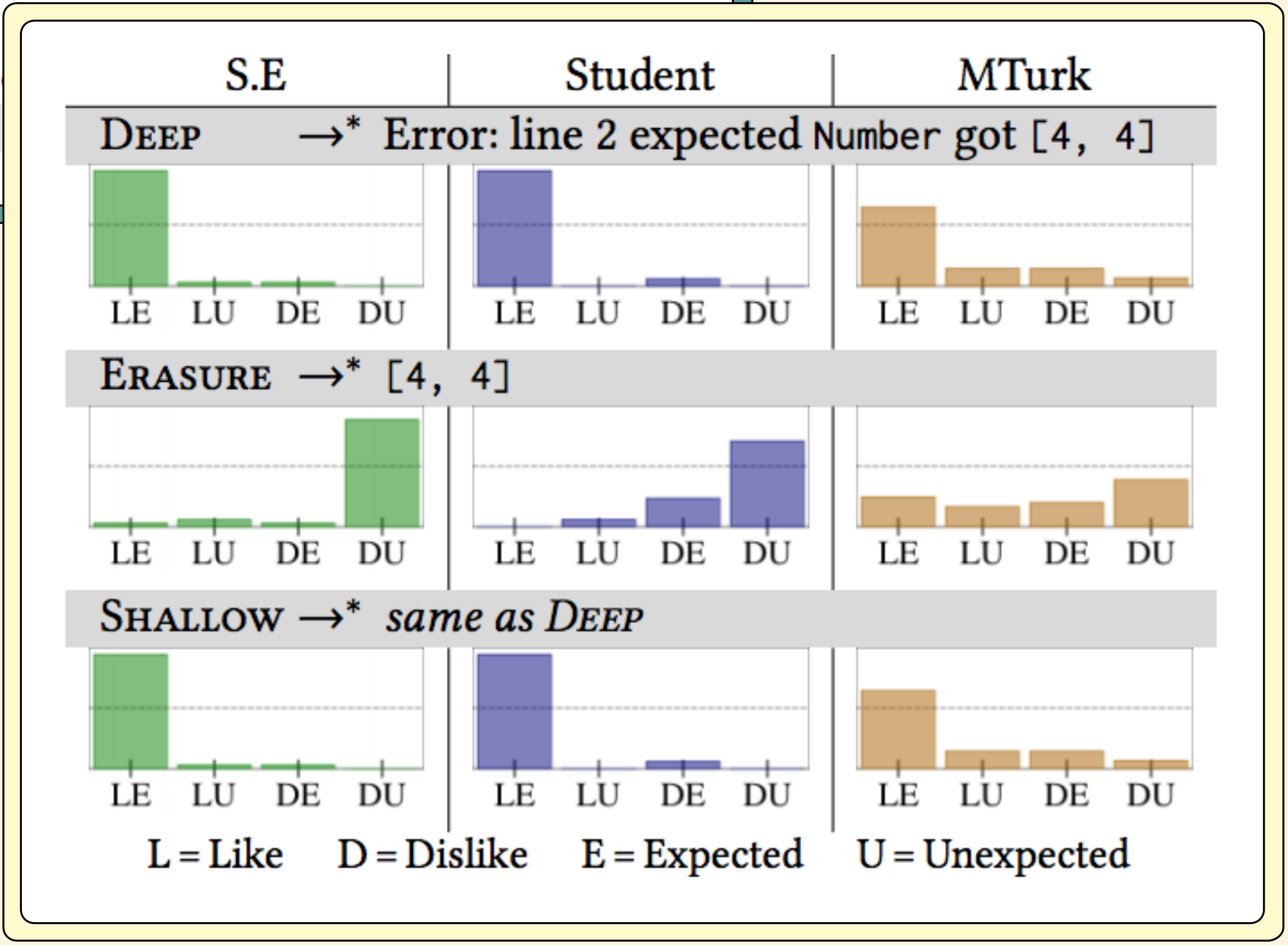
	LE	LU	DE	DU
Error: line 2 expected Number got `[4, 4]`	○	○	○	○
`[4, 4]`	○	○	○	○

Question 1

```
1 | var t = [4, 4];
2 | var x : Number = t;
3 | x
```

Error: line 2 expecte

`[4, 4]`



Developer Survey

Asked software engineers, students, and MTurk workers to rate potential **different behaviors** for programs

Results show a preference for Deep

More at DLS Tuesday 10:30am The Loft

cs.brown.edu/research/plt/dl/dls2018