Shallow Typed Racket

Ben Greenman

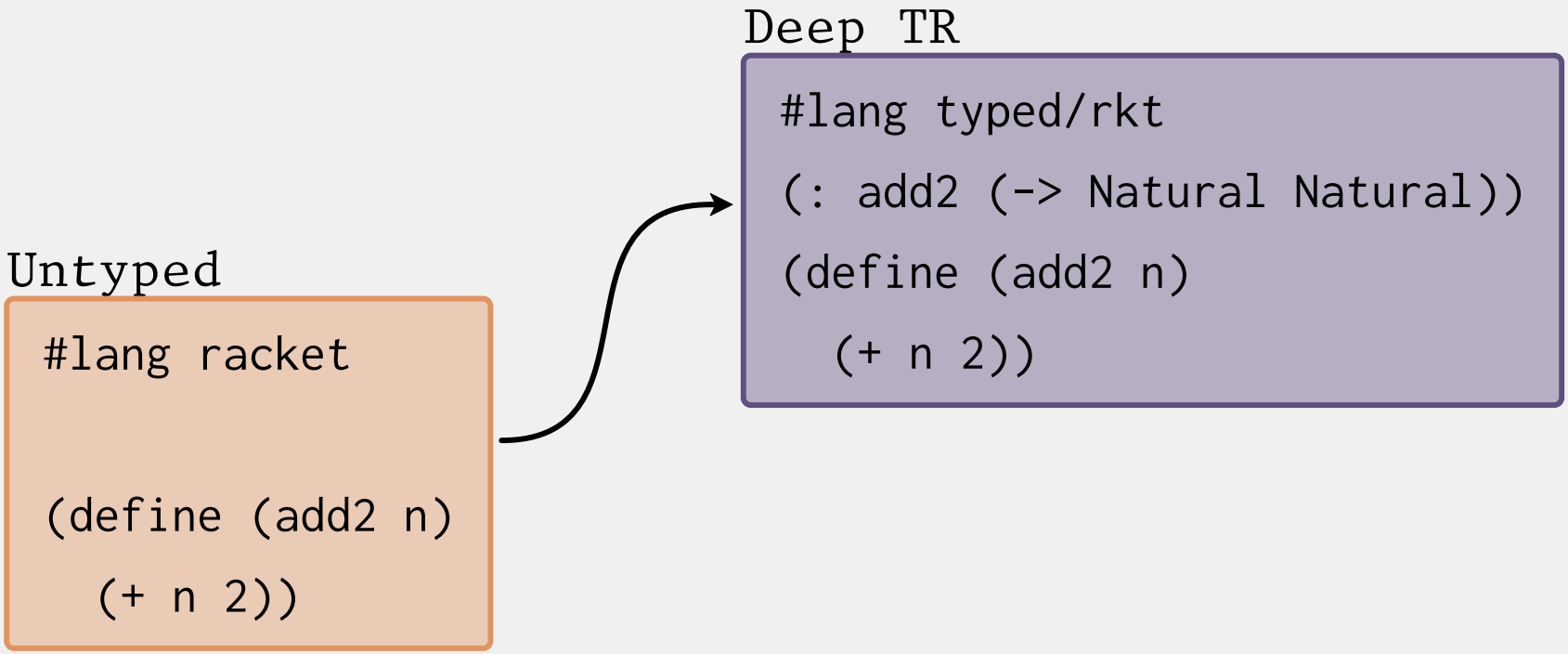
2020-10-18

Shallow Typed Racket

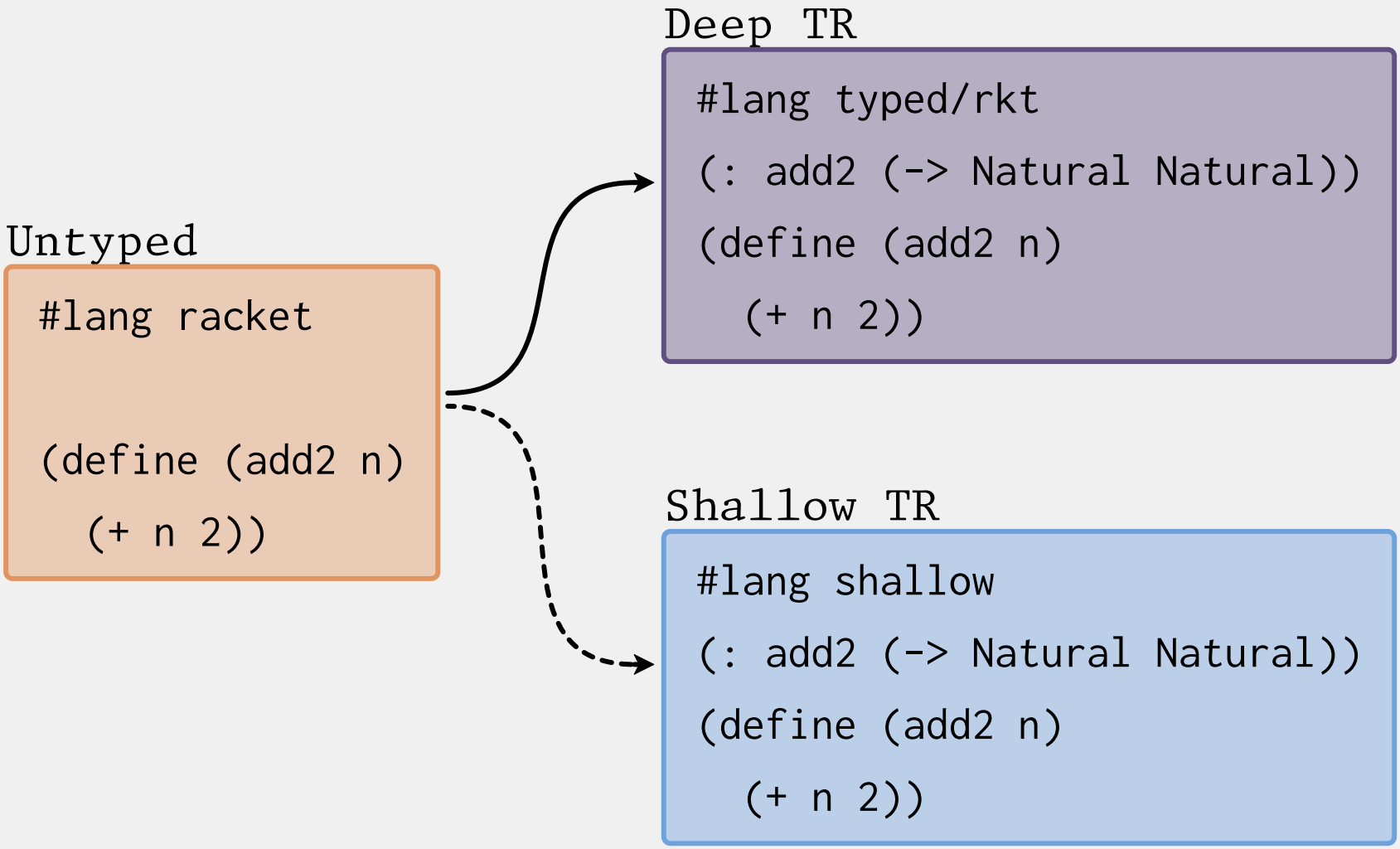
Shallow Typed Racket

"same types, but weaker"

same static types ...



same static types ...



... but weaker

Deep TR

```
(define (make-random-list)
  : (Listof String)
  ....)
```

repl

```
> (make-random-list)

> (make-random-list)

> (make-random-list)
```

... but weaker

Deep TR

```
(define (make-random-list)
  : (Listof String)
  ....)
```

repl

```
> (make-random-list)
'()
> (make-random-list)

> (make-random-list)
```

... but weaker

Deep TR

```
(define (make-random-list)
  : (Listof String)
  ....)
```

repl

```
> (make-random-list)
'()
> (make-random-list)
'("A" "B" "C")
> (make-random-list)
```

... but weaker

Deep TR

```
(define (make-random-list)
  : (Listof String)
  ....)
```

repl

```
> (make-random-list)
'()
> (make-random-list)
'("A" "B" "C")
> (make-random-list)
contract error
```

... but weaker

Shallow TR

```
(define (make-random-list)
  : (Listof String)
  ....)
```

repl

```
> (make-random-list)

> (make-random-list)

> (make-random-list)

> (make-random-list)
```

... but weaker

Shallow TR

```
(define (make-random-list)
  : (Listof String)
  ....)
```

repl

```
> (make-random-list)
'("A")
> (make-random-list)
contract error
> (make-random-list)

> (make-random-list)
```

... but weaker

Shallow TR

```
(define (make-random-list)
  : (Listof String)
  ....)
```

repl

```
> (make-random-list)
'("A")
> (make-random-list)
contract error
> (make-random-list)
'(a 2 "c")
> (make-random-list)
'(cheese (pizza))
```

... but weaker

Deep TR

```
(define (make-random-fn)
  : (-> String)
  ....)
```

repl

```
> (make-random-fn)

> ((make-random-fn))

> ((make-random-fn))
```

... but weaker

Deep TR

```
(define (make-random-fn)
  : (-> String)
  ....)
```

repl

```
> (make-random-fn)
#<procedure>
> ((make-random-fn))

> ((make-random-fn))
```

... but weaker

Deep TR

```
(define (make-random-fn)
  : (-> String)
  ....)
```

repl

```
> (make-random-fn)
#<procedure>
> ((make-random-fn))
"hello"
> ((make-random-fn))
contract error
```

... but weaker

Shallow TR

```
(define (make-random-fn)
  : (-> String)
  ....)
```

repl

```
> ((make-random-fn))

> ((make-random-fn))

> ((make-random-fn))
```

... but weaker

Shallow TR

```
(define (make-random-fn)
  : (-> String)
  ....)
```

repl

```
> ((make-random-fn))
"hello"
> ((make-random-fn))

> ((make-random-fn))
```

... but weaker

Shallow TR

```
(define (make-random-fn)
  : (-> String)
  ....)
```

repl

```
> ((make-random-fn))
"hello"
> ((make-random-fn))
42
> ((make-random-fn))
'()
```

... but weaker

Shallow TR

```
(define (make-random-fn)
  : (-> String)
  ....)
```

repl

```
> ((make-random-fn))
"hello"
> ((make-random-fn))
42
> ((make-random-fn))
'()
```

S-repl

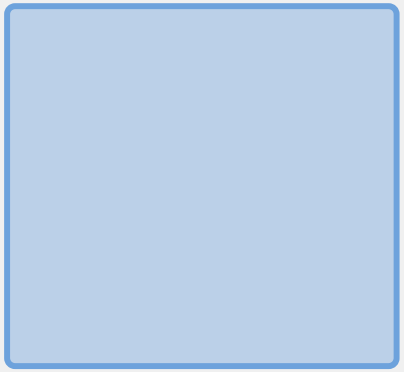
```
> ((make-random-fn))
contract error
```

... but weaker

Deep TR

guarantees full types
everywhere

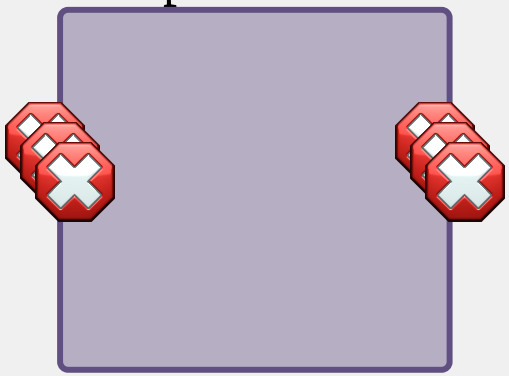
Shallow TR



guarantees type-shapes
only in typed code

Different guarantees => diff. methods

Deep TR

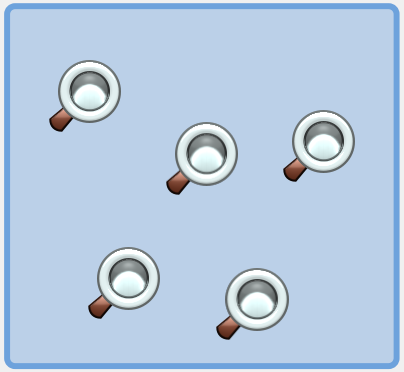


=> enforce full types

with contracts

at boundaries to non-deep code

Shallow TR



=> check type-shapes

with asserts

on every line of shallow code

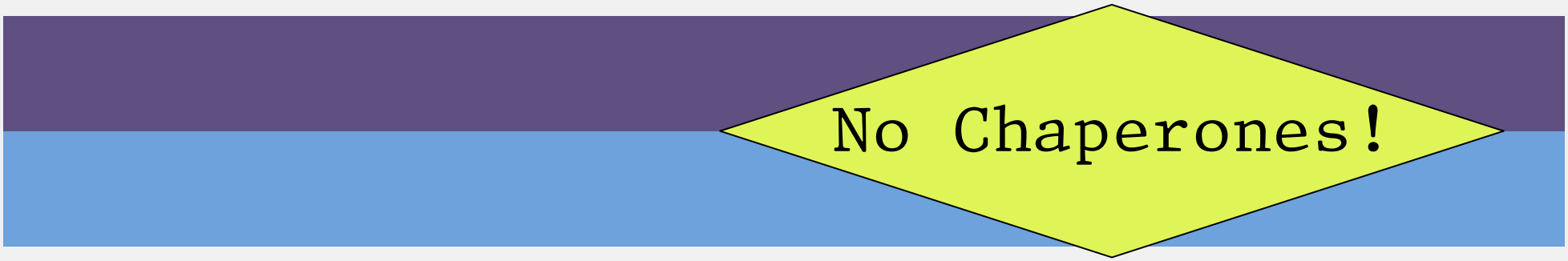
[Thanks Sam Tobin-Hochstadt & Michael M. Vitousek]

Shallow Typed Racket

"same types, but weaker"

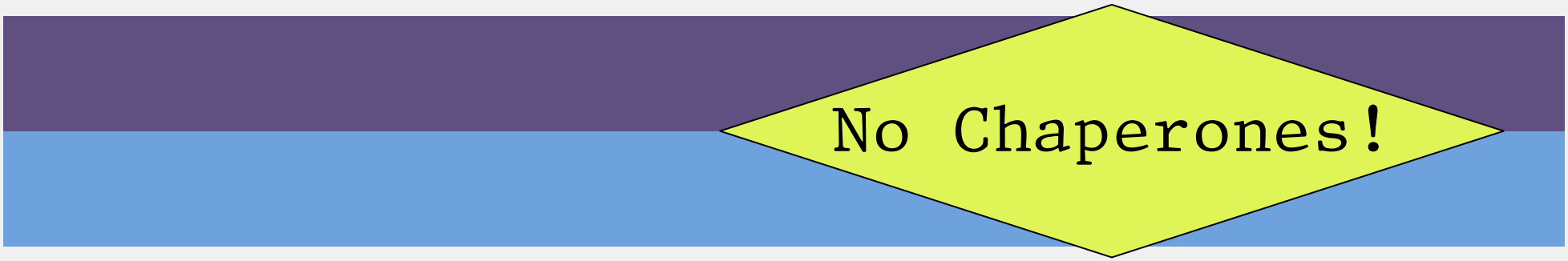
Shallow Typed Racket

"same types, but weaker"



Shallow Typed Racket

"same types, but weaker"



+ fast boundaries
+ more expressive
+ simple

Deep types can be slow

U-streams.rkt

```
#lang racket
....
```

U-main.rkt

```
#lang racket
....
(nth-prime 6667)
```

~2 sec.

U-streams.rkt

```
#lang racket
....
```

D-main.rkt

```
#lang typed/rkt
....
(nth-prime 6667)
```

~13 sec.

Shallow types can be faster

U-streams.rkt

```
#lang racket
....
```

D-main.rkt

```
#lang typed/rkt
....
(nth-prime 6667)
```

~13 sec.

U-streams.rkt

```
#lang racket
....
```

S-main.rkt

```
#lang shallow
....
(nth-prime 6667)
```

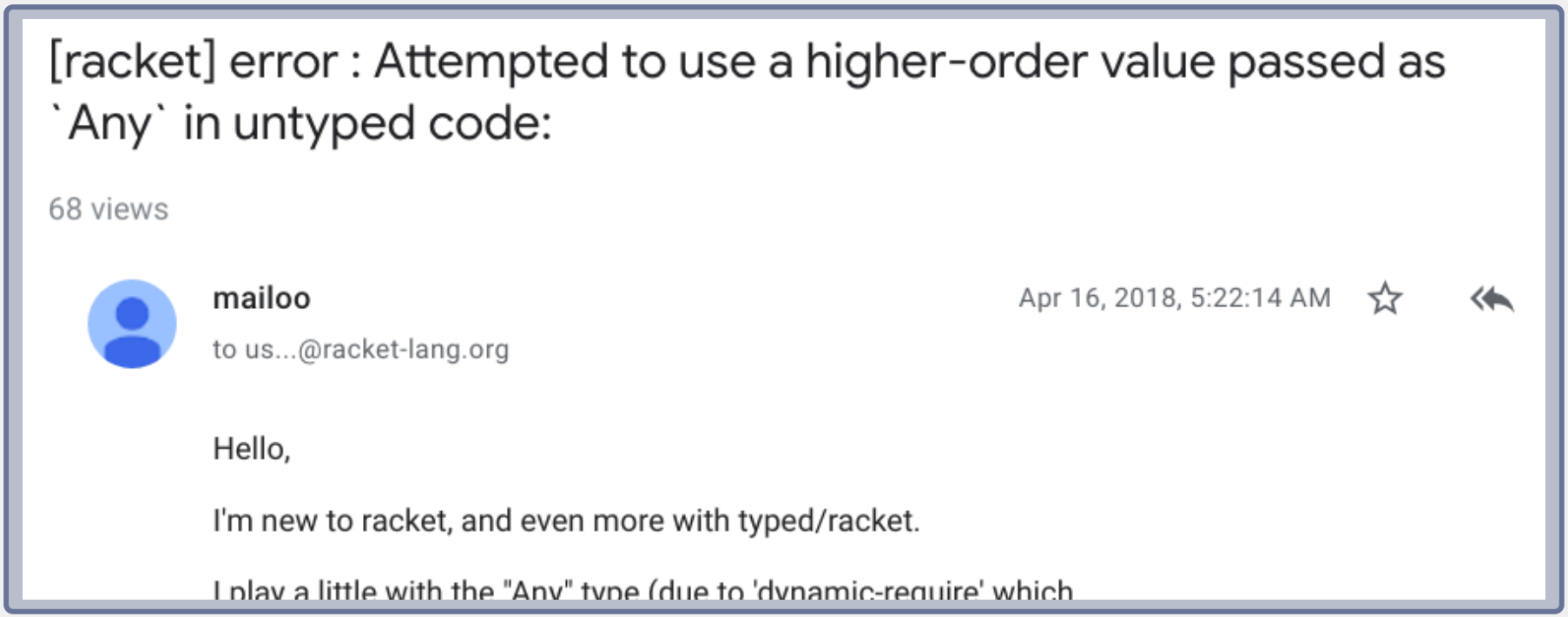
~4 sec.

Deep vs. Shallow, worst-case overhead

Benchmark	Worst Deep	Worst Shallow
sieve	10x	2x
fsmoo	451x	4x
dungeon	14000x	5x
mbta	2x	2x
tetris	12x	8x
synth	49x	4x

Deep types can be strict

Deep types can be strict



[racket] error : Attempted to use a higher-order value passed as `Any` in untyped code:

68 views

mailoo

to us...@racket-lang.org

Apr 16, 2018, 5:22:14 AM

Hello,

I'm new to racket, and even more with typed/racket.

I play a little with the "Any" type (due to 'dynamic-require' which

Deep types can be strict

D-box.rkt

```
#lang typed/rkt
(provide b)

(: b Any)
(define b (box 42))
```

U-main.rkt

```
#lang racket
(require "U-box.rkt")

(set-box! b 0)
```

Error

attempted to use higher-order value passed as Any

Shallow types are more permissive

S-box.rkt

```
#lang shallow
(provide b)

(: b Any)
(define b (box 42))
```

U-main.rkt

```
#lang racket
(require "U-box.rkt")

(set-box! b 0)
```

(void)

OK to use higher-order value passed as Any

Deep types can be weird

U-main.rkt

```
#lang racket


(index-of (list 'a 'b) 'a)
```

0

Deep types can be weird

D-main.rkt

```
#lang typed/rkt
(require/typed racket/list
  [index-of
   (All (T)
    (-> (Listof T) T
        (U #f Natural)))])

(index-of (list 'a 'b) 'a)
```

#f

because (not (equal? 'a #<A4>)) ... of course

Shallow types are more permissive

S-main.rkt

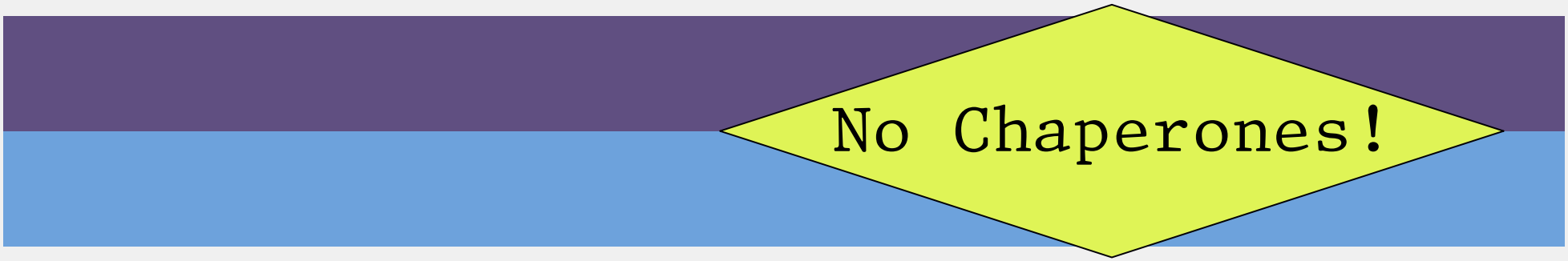
```
#lang shallow
(require/typed racket/list
  [index-of
   (All (T)
    (-> (Listof T) T
        (U #f Natural)))])

(index-of (list 'a 'b) 'a)
```

0

because (equal? 'a 'a)

Shallow Typed Racket

"same types, but weaker"



+ fast boundaries
+ more expressive
+ simple

Hang on ...

Shallow types can be slow!

D-streams.rkt

```
#lang typed/rkt
....
```

D-main.rkt

```
#lang typed/rkt
....
(nth-prime 6667)
```

<2 sec.

S-streams.rkt

```
#lang shallow
....
```

S-main.rkt

```
#lang shallow
....
(nth-prime 6667)
```

~5 sec.

Deep vs. Shallow, fully-typed

Benchmark	Deep/Untyped	Shallow/Untyped
sieve	1x	2x
fsmoo	0.9x	4x
dungeon	1x	5x
mbta	2x	2x
tetris	0.9x	8x
synth	0.9x	4x

Shallow types have limited scope

Shallow TR

```
(define (make-random-fn)
  : (-> String)
  ....)
```

repl

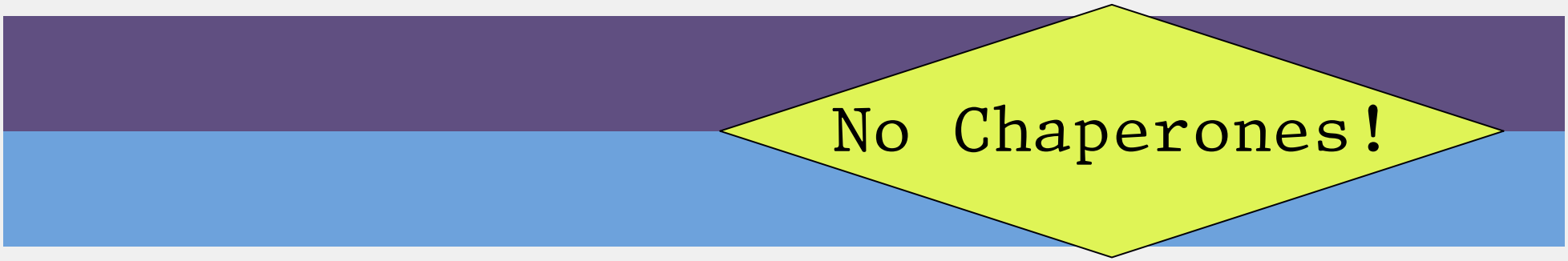
```
> ((make-random-fn))
42
```

S-repl

```
> ((make-random-fn))
contract error
```

Shallow Typed Racket

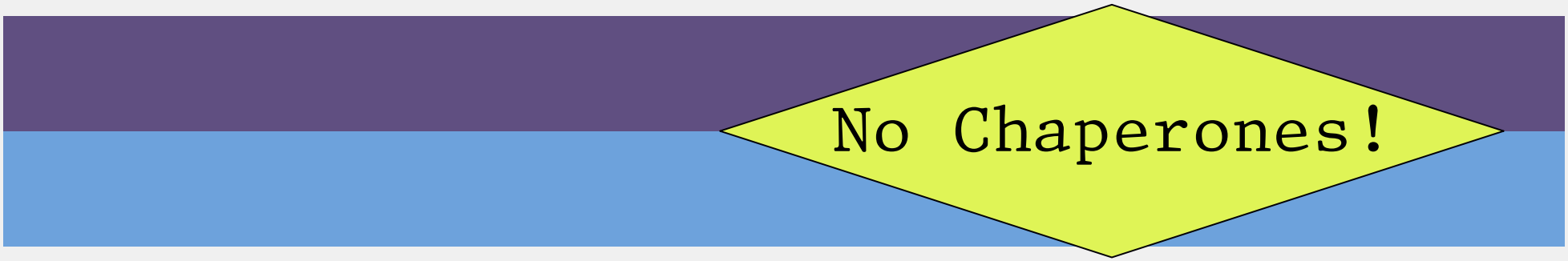
"same types, but weaker"



+ fast boundaries
+ more expressive
+ simple

Shallow Typed Racket

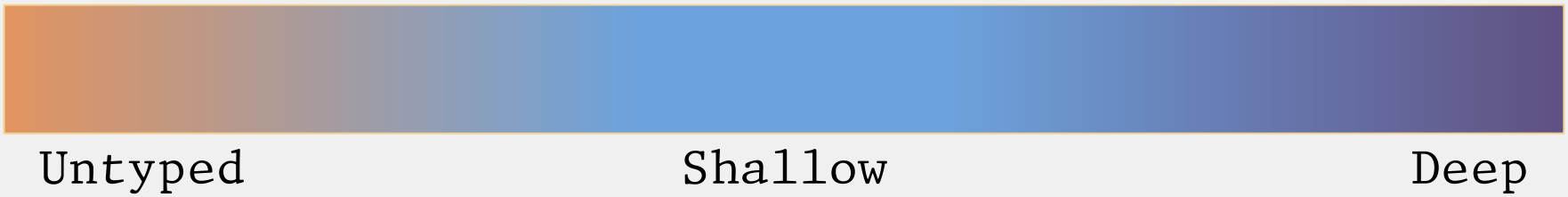
"same types, but weaker"



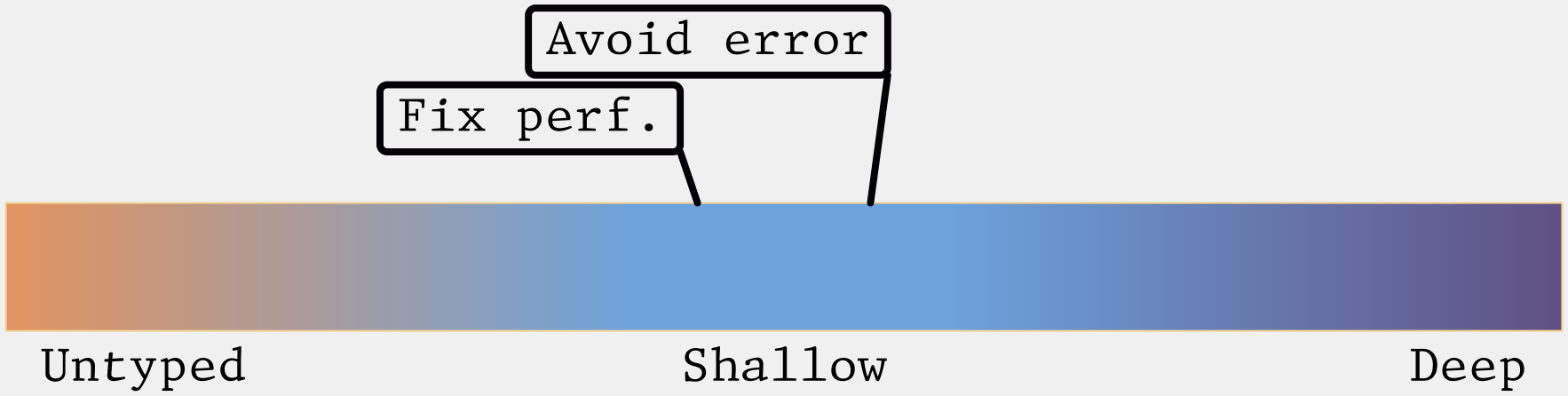
+ fast boundaries
+ more expressive
+ simple

- slow @ fully-typed
- temporary

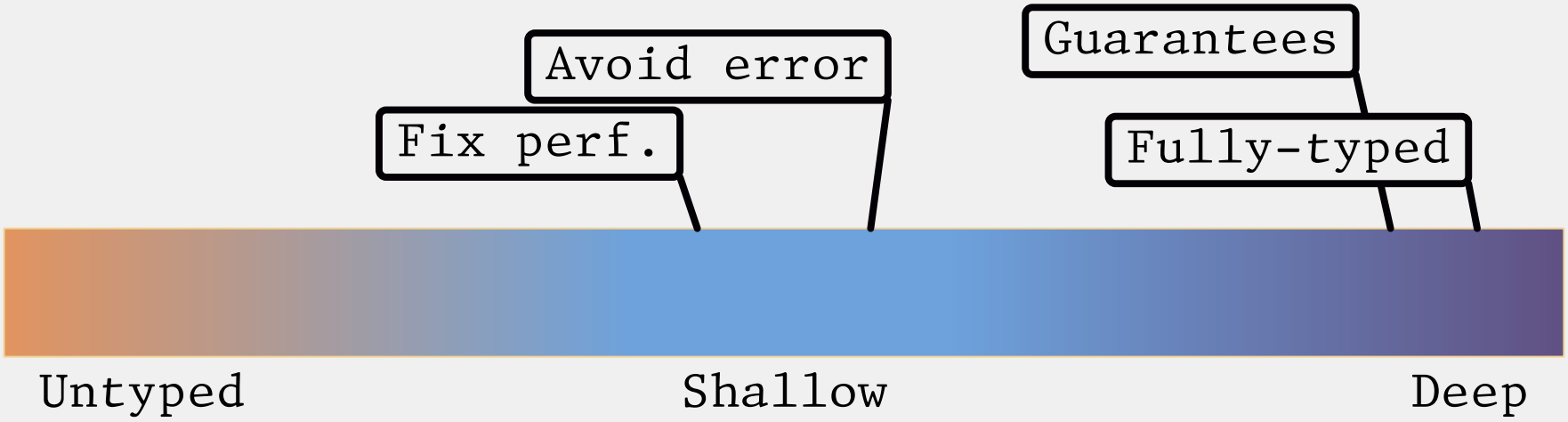
Recommendations



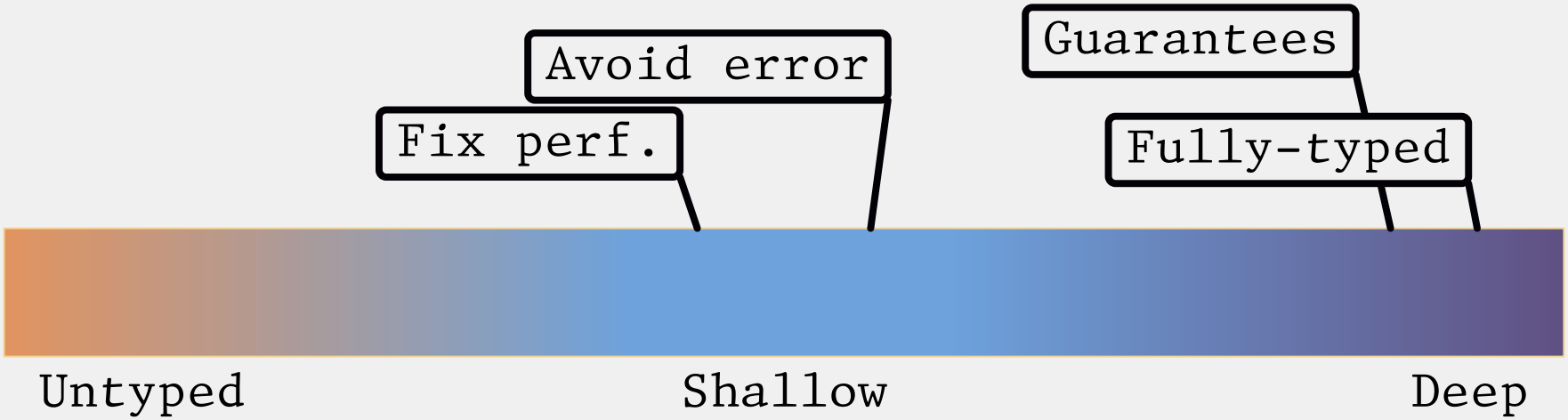
Recommendations

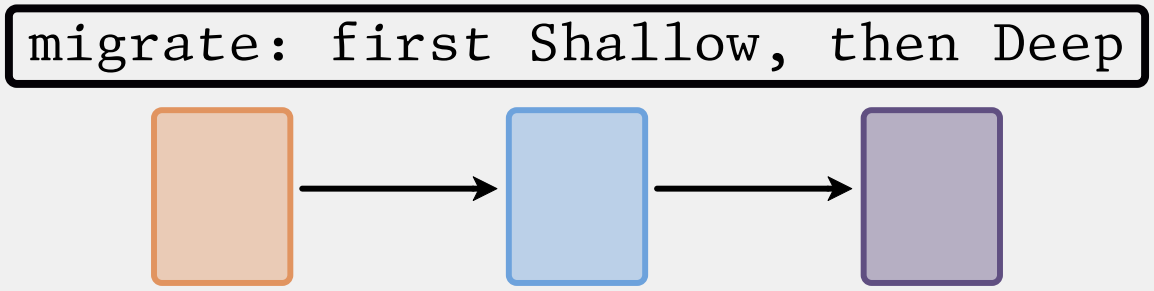


Recommendations



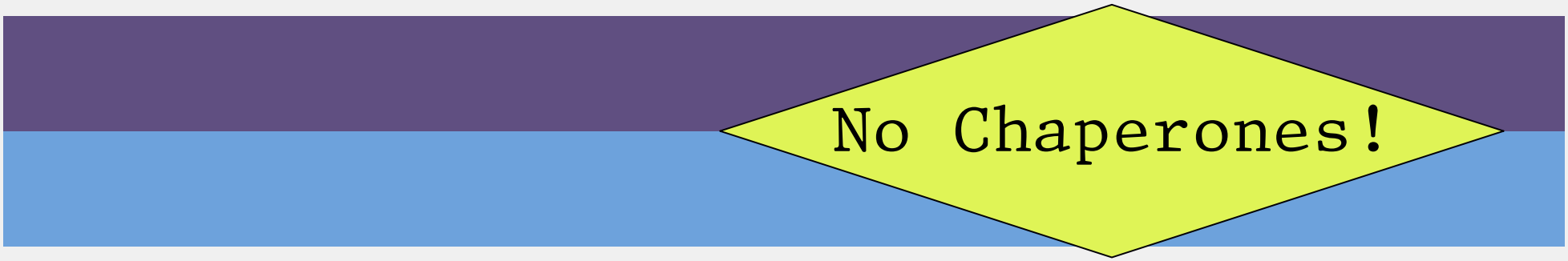
Recommendations





Shallow Typed Racket

"same types, but weaker"



+ fast boundaries
+ more expressive
+ simple

- slow @ fully-typed
- temporary

~~ coming soon ~~

RFC typed-racket/pull/952

PR typed-racket/pull/948