Shallow Typed Racket
Shallow Typed Racket

"same types, but weaker"
same static types ...

Untyped

#lang racket

(define (add2 n)
  (+ n 2))

Deep TR

#lang typed/rkt

(: add2 (-> Natural Natural))

(define (add2 n)
  (+ n 2))
same static types ...

Untyped

#lang racket
(define (add2 n)
 (+ n 2))

Deep TR
#lang typed/rkt
(: add2 (–> Natural Natural))
(define (add2 n)
 (+ n 2))

Shallow TR
#lang shallow
(: add2 (–> Natural Natural))
(define (add2 n)
 (+ n 2))
... 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)
(define (make-random-list) : (Listof String) ....)

> (make-random-list)
'()

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

> (make-random-list)
contract error
... but weaker

Shallow TR

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

(defun (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))
'(())
```
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"

No Chaperones!
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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Worst Deep</th>
<th>Worst Shallow</th>
</tr>
</thead>
<tbody>
<tr>
<td>sieve</td>
<td>10x</td>
<td>2x</td>
</tr>
<tr>
<td>fsmoo</td>
<td>451x</td>
<td>4x</td>
</tr>
<tr>
<td>dungeon</td>
<td>14000x</td>
<td>5x</td>
</tr>
<tr>
<td>mbta</td>
<td>2x</td>
<td>2x</td>
</tr>
<tr>
<td>tetris</td>
<td>12x</td>
<td>8x</td>
</tr>
<tr>
<td>synth</td>
<td>49x</td>
<td>4x</td>
</tr>
</tbody>
</table>
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

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

```racket
#lang typed/rkt
(provide b)
(: b Any)
(define b (box 42))
```

U-main.rkt

```racket
#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

```racket
#lang racket
(index-of (list 'a 'b) 'a)
```

0
Deep types can be weird

D-main.rkt

```scheme
#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

```racket
#lang shallow
(require/typed racket/list
  [index-of
   (All (T)
     (-> (Listof T) T
       (U #f Natural))))])
(index-of (list 'a 'b) 'a)
```

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

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Deep/Untyped</th>
<th>Shallow/Untyped</th>
</tr>
</thead>
<tbody>
<tr>
<td>sieve</td>
<td>1x</td>
<td>2x</td>
</tr>
<tr>
<td>fsmoo</td>
<td>0.9x</td>
<td>4x</td>
</tr>
<tr>
<td>dungeon</td>
<td>1x</td>
<td>5x</td>
</tr>
<tr>
<td>mbta</td>
<td>2x</td>
<td>2x</td>
</tr>
<tr>
<td>tetris</td>
<td>0.9x</td>
<td>8x</td>
</tr>
<tr>
<td>synth</td>
<td>0.9x</td>
<td>4x</td>
</tr>
</tbody>
</table>
Shallow types have limited scope

Shallow TR

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

repl

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

S-repl

```
> ((make-random-fn))
contract error
```
Shallow Typed Racket

"same types, but weaker"

No Chaperones!

+ fast boundaries
+ more expressive
+ simple
Shallow Typed Racket
"same types, but weaker"

+ fast boundaries
+ more expressive
+ simple

- slow @ fully-typed
- temporary
Recommendations
Recommendations

Fix perf.

Avoid error
Recommendations

- Untyped
  - Fix perf.
  - Avoid error

- Shallow

- Deep
  - Guarantees
  - Fully-typed
Recommendations

Untyped \rightarrow \text{Avoid error} \rightarrow \text{Fix perf.} \rightarrow \text{Guarantees} \rightarrow \text{Fully-typed} \rightarrow \text{Deep}

migrate: first Shallow, then Deep
Shallow Typed Racket
"same types, but weaker"

No Chaperones!

+ fast boundaries
+ more expressive
+ simple

- slow @ fully-typed
- temporary

~~ coming soon ~~

RFC typed-racket/pull/952
PR typed-racket/pull/948