Typed Clojure

A Typed-Macro Writer's Toolkit

Ambrose Bonnaire-Sergeant

I have no idea how your macro works! Help!!

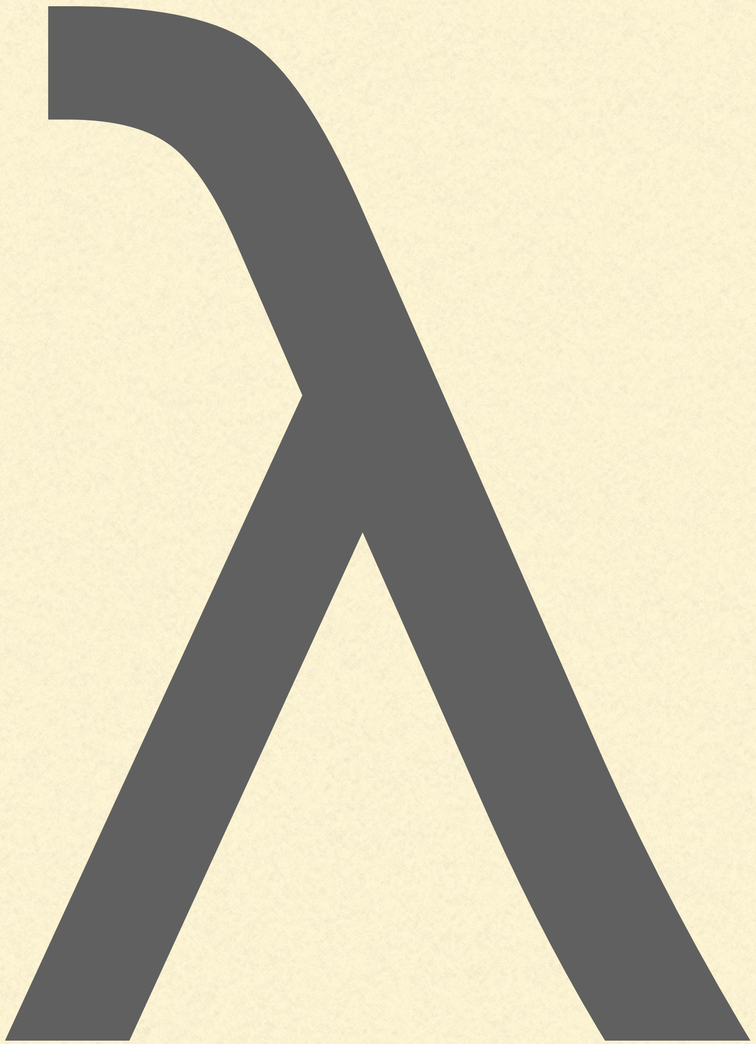
–Type System

I ***want*** to teach the type system how my macro works!

–Macro Author

The pitch

Providing an
extensible interface
to Typed Clojure's
internals helps it be more
expressive and usable.



The pitch

Providing an ***extensible interface*** to Typed Clojure's internals helps it be more ***expressive and usable.***

The evidence

1. Better errors
2. Less annotations
3. Simpler checks

The pitch

Providing an ***extensible interface*** to Typed Clojure's internals helps it be more ***expressive and usable.***

The evidence

1. Better errors
2. Less annotations
3. Simpler checks

```
(when (number? a)
  a)
```

expands to

```
(if (number? a)
  a
  nil)
```

```
(when (number? a)
  a)
```

expands to

```
(if (number? a)
  a
  nil)
```

Expected: Number
Found: nil

```
(when (number? a)
  a)
```

expands to

```
(if (number? a)
  a
  nil)
```

We'd like to blame

Expected: Number
Found: nil

```
(defmacro when [test & body]
  `(if ~test
     (do ~@body)
     nil))
```

But, we actually blame!

expands to

```
(when (number? a)
  a)
```

```
(if (number? a)
  a
  nil)
```

We'd like to blame

Expected: Number
Found: nil

```
(defmacro when [test & body]
  `(if ~test
     (do ~@body)
     nil))
```

Blame:
~@body

Blame:
(when ~test ~@body)

Solution: Custom Blame Forms

The pitch

Providing an ***extensible interface*** to Typed Clojure's internals helps it be more ***expressive and usable.***

The evidence

1. Better errors
2. Less annotations
3. Simpler checks

```
(for [a :- Int, '(1 2 3)] :- Int
  (inc a))
;=> (2 3 4)
```

```
(for [a :- Int, '(1 2 3)] :- Int
  (inc a))
;=> (2 3 4)
```

How to eliminate annotation?

```
(for [a :- Int, '(1 2 3)]
  (inc a))
```

Expected Type:
(Seq Sym)

```
(for [a :- Int, '(1 2 3)]
  (inc a))
```

Expected Type:
(Seq Sym)

expands to

```
(…
  (inc a)
  …)
```

```
(for [a :- Int, '(1 2 3)]
  (inc a))
```

Expected Type:
(Seq Sym)

expands to

```
(…
  (inc a)
  …)
```

We'd like to propagate expected type

```
(for [a :- Int, '(1 2 3)]
  (inc a))
```

Expected Type:
(Seq Sym)

expands to

```
(…
  (inc a)
  …)
```

Expected Type:
Sym

Solution: Custom "Expected Type" Propagation

The pitch

Providing an ***extensible interface*** to Typed Clojure's internals helps it be more ***expressive and usable.***

The evidence

1. Better errors
2. Less annotations
3. Simpler checks

```
(for [a '(1 2 3)]
  (inc a))
```

```
(for [a '(1 2 3)]
  (inc a))
```

expands to

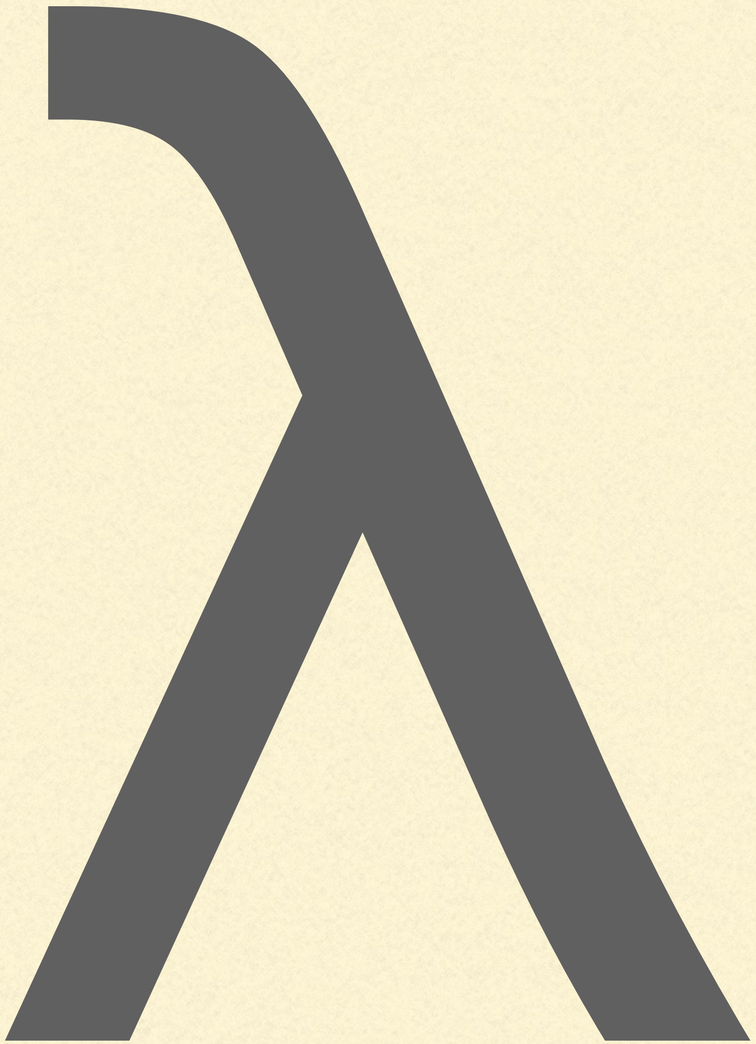
```
(map inc '(1 2 3))
```

<u>Blame</u>:
(inc a)

Solution: Simplified Expansion

The pitch

Providing an
extensible interface
to Typed Clojure's
internals helps it be more
expressive and usable.



The pitch

Providing an ***extensible interface*** to Typed Clojure's internals helps it be more ***expressive and usable.***

The evidence

1. Better errors
2. Less annotations
3. Simpler checks

The pitch

Providing an ***extensible interface*** to Typed Clojure's internals helps it be more ***expressive and usable.***

The evidence

1. Better errors
2. Less annotations
3. Simpler checks

Thanks!