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 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
We’d like to blame

Expected: Number
Found: nil
(defmacro when [test & body] `(if ~test
   (do ~@body)
   nil))

We'd like to blame

(when (number? a) a)

expands to

(if (number? a) a nil)

Expected: Number
Found: nil

But, we actually blame!
(defmacro when [test & body]
  `(if ~test
     (do ~@body)
     nil))

Solution: Custom Blame Forms
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))

expands to

(...
  (inc a)
  ...
)

Expected Type:
(Seq Sym)
expands to

\[(\text{for } [a \leftarrow \text{Int}, \, '(1 \ 2 \ 3)] \ (\text{inc } a))\]

Expected Type:
\[(\text{Seq } \text{Sym})\]

We'd like to propagate expected type

\[(\text{inc } a)\]

We'd like to propagate expected type
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))

Solution: Simplified Expansion
Providing an extensible interface to Typed Clojure’s internals helps it be more expressive and usable.
<table>
<thead>
<tr>
<th>The pitch</th>
<th>The evidence</th>
</tr>
</thead>
</table>
| Providing an **extensible interface** to Typed Clojure’s internals helps it be more **expressive and usable**. | 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!