

# Corten: Refinement Types for Imperative Languages with Ownership

Abschlusspräsentation Masterarbeit

Carsten Csiky | 26th Oktober 2022

## Inhaltsverzeichnis



- 1. Motivation
- 2. Empirical Analysis
- 3. Solution
- 4. Soundness Justification
- 5. Related Work
- 6. Conclusion / Future Work



```
fn max(a: i32, b: i32) {
       if a > b { a } else { b }
```

Motivation •00000

Empirical Analysis

Solution

Soundness Justification

Related Work



```
fn max(a: i32, b: i32) {
       if a > b { a } else { b }
}
```

Return Value  $(v): v \geq a \land v \geq b$ 



```
fn max(a: i32, b: i32) {
       if a > b { a } else { b }
```

Return Value  $(v): v > a \land v > b$ 

Refinement Typesrondon\_liquid\_2008 in Functional Programming Languages



```
//@ max(a: i32, b: i32) -> {v:i32 | v >= a && v >= b }

fn max(a: i32, b: i32) -> i32 {
    if a > b { a } else { b }
}
```

Motivation

Empirical Analysis

Solution

Soundness Justification

Related Work





```
//@ max(a: i32, b: i32) -> \{v:i32 | v >= a && v >= b \}
fn max(a: i32, b: i32) -> i32 {
         if a > b { a } else { b }
}
   let \Gamma = (a : \{v : i32 \mid true\}, b : \{v : i32 \mid true\}) and \tau = \{v : i32 \mid v \ge a \land v \ge b\}
```

$$\Gamma \vdash \text{if } a > b \{a\} \text{ else } \{b\} : \tau$$

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
//@ \max(a: i32, b: i32) -> \{v:i32 \mid v >= a \&\& v >= b \}
fn max(a: i32, b: i32) -> i32 {
          if a > b { a } else { b }
}
   let \Gamma = (a : \{v : i32 \mid true\}, b : \{v : i32 \mid true\}) and \tau = \{v : i32 \mid v \ge a \land v \ge b\}
```

$$\Gamma$$
,  $a > b \vdash a : \tau$ 

$$\Gamma$$
,  $\neg$ ( $a > b$ )  $\vdash b : \tau$ 

$$\Gamma \vdash \text{if } a > b \{a\} \text{ else } \{b\} : \tau$$

Motivation 000000

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
//@ \max(a: i32, b: i32) -> \{v:i32 \mid v >= a \&\& v >= b \}
fn max(a: i32, b: i32) -> i32 {
         if a > b { a } else { b }
}
  let \Gamma = (a : \{v : i32 \mid true\}, b : \{v : i32 \mid true\}) and \tau = \{v : i32 \mid v \ge a \land v \ge b\}
```

$$\frac{\Gamma, a > b \vdash \{v : i32 \mid v \doteq a\} \preceq \tau}{\Gamma, a > b \vdash a : \tau} \qquad \frac{\Gamma, \neg(a > b) \vdash b : \tau}{\Gamma, \neg(a > b) \vdash b : \tau}$$

Motivation 000000

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
//@ \max(a: i32, b: i32) -> \{v:i32 \mid v >= a \&\& v >= b \}
fn max(a: i32, b: i32) -> i32 {
           if a > b { a } else { b }
}
   let \Gamma = (a : \{v : i32 \mid true\}, b : \{v : i32 \mid true\}) and \tau = \{v : i32 \mid v \ge a \land v \ge b\}
                       *
    \Gamma, a > b \vdash a : \{v : i32 \mid v = a\} \Gamma, a > b \vdash \{v : i32 \mid v = a\} \prec \tau
                                         \Gamma. a > b \vdash a : \tau
                                                                                                          \Gamma, \neg (a > b) \vdash b : \tau
```

 $\Gamma \vdash \text{if } a > b \{a\} \text{ else } \{b\} : \tau$ 

Motivation 000000

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
//@ \max(a: i32, b: i32) -> \{v:i32 \mid v >= a \&\& v >= b \}
fn max(a: i32, b: i32) -> i32 {
               if a > b { a } else { b }
}
    let \Gamma = (a : \{v : i32 \mid true\}, b : \{v : i32 \mid true\}) and \tau = \{v : i32 \mid v \ge a \land v \ge b\}
                                                                        \mathsf{SMT-VALID}\left(\begin{array}{l}\mathsf{true} \wedge \mathsf{true} \wedge a > \mathsf{b}\\ \wedge \mathsf{v} \doteq a\\ \Longrightarrow (\mathsf{v} \geq \mathsf{a} \wedge \mathsf{v} \geq \mathsf{b})\end{array}\right)
      \Gamma, a > b \vdash a : \{v : i32 \mid v \doteq a\} \Gamma, a > b \vdash \{v : i32 \mid v \doteq a\} \leq \tau
                                                           \Gamma. a > b \vdash a : \tau
                                                                                                                                                       \Gamma, \neg (a > b) \vdash b : \tau
                                                                   \Gamma \vdash \text{if } a > b \{a\} \text{ else } \{b\} : \tau
```

Motivation 000000

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
//@ \max(a: i32, b: i32) -> \{v:i32 \mid v >= a \&\& v >= b \}
fn max(a: i32, b: i32) -> i32 {
               if a > b { a } else { b }
    let \Gamma = (a : \{v : i32 \mid true\}, b : \{v : i32 \mid true\}) and \tau = \{v : i32 \mid v \ge a \land v \ge b\}
                                                                       \mathsf{SMT-VALID}\left(\begin{array}{l}\mathsf{true} \wedge \mathsf{true} \wedge a > \mathsf{b}\\ \wedge \mathsf{v} \doteq a\\ \Longrightarrow (\mathsf{v} \geq \mathsf{a} \wedge \mathsf{v} \geq \mathsf{b})\end{array}\right)
      \Gamma, a > b \vdash a : \{v : i32 \mid v = a\} \Gamma, a > b \vdash \{v : i32 \mid v = a\} \prec \tau
                                                          \Gamma. a > b \vdash a : \tau
                                                                                                                                                    \Gamma, \neg (a > b) \vdash b : \tau
                                                                  \Gamma \vdash \text{if } a > b \{a\} \text{ else } \{b\} : \tau
```

Motivation 000000

}

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
clamp(a: &mut i32, b: i32) {
   if *a > b { *a = b }
```

Motivation 000000

Empirical Analysis

Solution

Soundness Justification

Related Work



```
clamp(a: &mut i32, b: i32) {
        if *a > b { *a = b }
}
    client(...) {
        . . .
        clamp(&mut x, 5);
        clamp(&mut y, 6);
        print(x);
        . . .
```

Motivation 000000

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
clamp(a: &mut i32, b: i32) {
    if *a > b { *a = b }
client(...) {
    clamp(&mut x, 5);
    clamp(&mut y, 6);
    print(x);
    . . .
```

What does this it print(x) output?

- In most imperative programming languages:
  - Could be: old x or 5

Motivation 000000

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
clamp(a: &mut i32, b: i32) {
        if *a > b { *a = b }
}
    client(...) {
        . . .
        clamp(&mut x, 5);
        clamp(&mut y, 6);
        print(x);
        . . .
```

What does this it print(x) output?

- In most imperative programming languages:
  - Could be: old x or 5
  - But also 6 (if x aliases with y)!

Motivation 000000

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



What does this it print(x) output?

- In most imperative programming languages:
  - Could be: old x or 5
  - But also 6 (if x aliases with y)!
- In Rust:
  - Just old x or 5
  - And nothing else!

Motivation

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



```
clamp(a: &mut i32, b: i32) {
    // borrows a
    // owns b
    if *a > b { *a = b }
    // "returns" the borrow of a
client(...) { // owns x, y
    clamp(&mut x, 5); // lend x mutably
    clamp(&mut y, 6); // lend y mutably
    print(x);
    . . .
```

### Ownership in Rust: Mutability XOR Aliasing

Each lexical scope tracks permissions for visible memory objects. Possible Permission Levels:

- Owner (e.g. b)
  - can: read, write
  - transfer ownership (if no outstanding borrows)
- Mutable Reference (e.g. &mut x)
  - can: read, write
  - guarantee: no aliasing
- Immutable Reference (e.g. &v)
  - can: read, alias
  - guarantee: no mutation

Motivation 000000

**Empirical Analysis** 

Solution

Soundness Justification

Related Work



#### Consequences:

- unique data owner
- no global, mutable state
- no cycles in memory structure

# Ownership in Rust: Mutability XOR Aliasing

Each lexical scope tracks permissions for visible memory objects. Possible Permission Levels:

- Owner (e.g. b)
  - can: read, write
  - transfer ownership (if no outstanding borrows)
- Mutable Reference (e.g. &mut x)
  - can: read, write
    - guarantee: no aliasing
- Immutable Reference (e.g. &v)
  - can: read, alias
  - guarantee: no mutation



#### Consequences:

- unique data owner
- no global, mutable state
- no cycles in memory structure

#### Used for:

- safe non-gc memory management
- safe concurrency
- safe low-level hardware access

## Ownership in Rust: Mutability XOR Aliasing

Each lexical scope tracks permissions for visible memory objects. Possible Permission Levels:

- Owner (e.g. b)
  - can: read, write
  - transfer ownership (if no outstanding borrows)
- Mutable Reference (e.g. &mut x)
  - can: read, write
  - guarantee: no aliasing
- Immutable Reference (e.g. &v)
  - can: read, alias
  - guarantee: no mutation



#### Consequences:

- unique data owner
- no global, mutable state
- no cycles in memory structure

#### Used for:

- safe non-gc memory management
- safe concurrency
- safe low-level hardware access
- ⇒ show: program verification as well

# Ownership in Rust: Mutability XOR Aliasing

Each lexical scope tracks permissions for visible memory objects. Possible Permission Levels:

- Owner (e.g. b)
  - can: read, write
  - transfer ownership (if no outstanding borrows)
- Mutable Reference (e.g. &mut x)
  - can: read, write
  - guarantee: no aliasing
- Immutable Reference (e.g. &v)
  - can: read, alias
  - guarantee: no mutation

# **Extend Refinement Types to Rust**



asd

Motivation

Empirical Analysis

Solution

Soundness Justification

Related Work

#### Literatur



# Backup-Teil

Folien, die nach \beginbackup eingefügt werden, zählen nicht in die Gesamtzahl der Folien.

Zweiter Abschnitt

Farben o

# Blöcke in den KIT-Farben



Greenblock Standard (block)

= exampleblock

Blueblock

Redblock = alertblock

Brownblock

Purpleblock

Cyanblock

Yellowblock

Lightgreenblock

Orangeblock

Grayblock

Contentblock (farblos)

Zweiter Abschnitt

11/8

Farben o

# **Auflistungen**



#### Text

- Auflistung Umbruch
- Auflistung
  - Auflistung
  - Auflistung

Zweiter Abschnitt

Bei Frames ohne Titel wird die Kopfzeile nicht angezeigt, und der freie Platz kann für Inhalte genutzt werden.

Zweiter Abschnitt

●○○

Farben
○

13/8 26.10.2022 Carsten Csiky: Rust & Refinement Types

Department of Informatics – Institute of Information Security and Dependability (KASTEL) Bei Frames mit Option [plain] werden weder Kopf- noch Fußzeile angezeigt.

# Beispielinhalt



Bei Frames mit Option [t] werden die Inhalte nicht vertikal zentriert, sondern an der Oberkante begonnen.

Zweiter Abschnitt 0000

Farben

# Beispielinhalt: Literatur



Zweiter Abschnitt 0000

Farben

# **Farbpalette**



kit-green100	kit-green9	0 kit-green8	30 kit-gree	n70 kit-g	reen60 k	it-green50	kit-gr	een40	kit-green30		it-green25	kit-gree	t-green20 kit-		kit-gı	reen10	kit-greens	<b>i</b>
kit-blue100	kit-blue90 kit-blue80		kit-blue70	kit-blue60	kit-blue5	0 kit-blu	e40 k	it-blue30	kit-bl	lue25	kit-blue20	kit-blue	15 kit-	blue10	kit-blue5			
kit-red100	kit-red90	kit-red80 kit	t-red70 kit	-red60 ki	t-red50 k	it-red40	kit-red3	0 kit-red	d25	kit-red20	kit-red1	5 kit-re	d10 k	it-red5				
kit-gray100	kit-gray90	kit-gray80	kit-gray70	kit-gray60	kit-gray5	0 kit-gra	ay40 F	kit-gray30	kit-g	ray25	kit-gray20	kit-gray	15 kit	-gray10	kit-gray	5		
kit-orange100 kit-orang		e90 kit-orange80 k		orange70	nge70 kit-orange		ange50	50 kit-orange		kit-orange30 kit		-orange2	inge25 kit-orange20		kit-orange15		kit-orange	0 kit-orange
kit-lightgreen100 kit-li		ntgreen90 k	it-lightgreen8	0 kit-ligh	tgreen70	kit-lightgreen60		kit-lightgreen		0 kit-lightgreen40		kit-lightgreen30		kit-lighto	kit-lightgreen25		ntgreen20	kit-lightgreen
kit-lightgreen10 kit-lightgreen5																		
kit-brown100	kit-brown9	00 kit-brown	180 kit-bro	wn70 kit	-brown60	kit-brown5	60 kit-	brown40	n40 kit-brov		kit-brown	25 kit-l	kit-brown20 kit-bro		wn15 kit-brown		n10 kit-brown5	
kit-purple100	kit-purple90 kit-purp		e80 kit-pu	rple70 ki	t-purple60	kit-purple	e50 ki	t-purple40	40 kit-purp		kit-purp	le25 ki	t-purple20 kit-pu		rple15	kit-pur	ole10 kit	-purple5
kit-cyan100	kit-cyan90	kit-cyan80	kit-cyan70	kit-cyan	60 kit-cya	ın50 kit-	cyan40	kit-cyan	an30 kit-cya		5 kit-cya	n20 kit	kit-cyan15 kit-cya		10 kit	-cyan5		

Zweiter Abschnitt

