Karlsruher Institut für Technologie

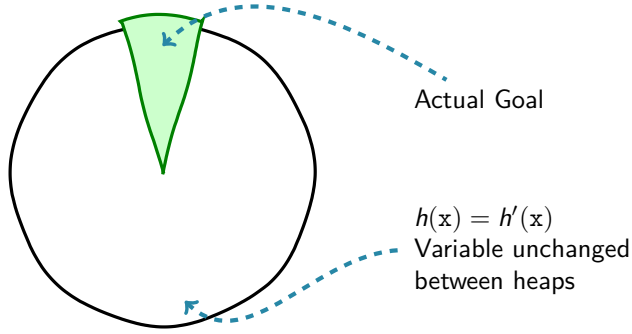
LiquidRust

Embracing Mutability on Refinement Types using Rust's Ownership Model

Carsten Csiky | 24. Februar 2022

Motivation

```
public IntList square(IntList list) {
  return list.map(x -> x*x);
}
```




Mutability is the problem!

Refinement Types

- Extension of the Type-System
- `i32` $\Rightarrow$ `{ v: i32 | v > 0 }`

Type Checking

Given Type Context:

$\Gamma = \{f : \{v : \tau_{param} \mid p(v)\} \rightarrow \{u : \tau_{res} \mid r(u)\}, a : \{w : \tau_{arg} | q(w)\}\}$

Is `f(a)` type correct?

- $\{v : \tau_{arg} \mid p(v)\} \preceq \{w : \tau_{param} \mid q(w)\}$

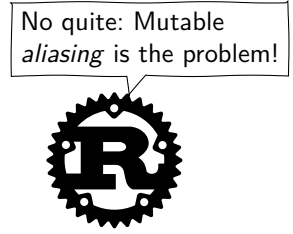
Subtyping judgment $\{\tau | p\} \preceq \{\tau' | q\}$

(I) Base types τ, τ' unify

(II) $\forall v.\ \tau.p(v) \Rightarrow q(v)$

Mutability is the problem!


Mutability is the problem!


No quite: Mutable
aliasing is the problem!

Every lexical scope:

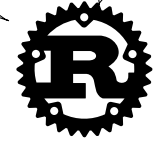
- tracks "Permissions"
- for every lexically visible variable

Possible Variable Permissions:

1. Owner `v`
   - read
   - write
   - can transfer ownership, if no outstanding borrows at time of transfer
2. immutable Reference `&v`
   - read
   - guarantee: no writes
   - possibly other readers
3. mutable Reference `&mut v`
   - read and write
   - no other reader or writer

Mutability is the problem!

No quite: Mutable *aliasing* is the problem!



Rust + Refinement Types?

How can Refinement Types be adapted for mutable Languages leveraging Rust's Ownership Model?

Goal

- design a decidable type system with refinement types for functional verification in Rust.
- implement a proof of concept for automatic verification / type checking.
- backward-compatible to Rust
- limit to safe, (non-higher-order?) Rust
- no liquid type inference

How?

- adapt Refinement Types semantics to Rust, especially for mutability in Rust's Ownership Model
- prototype translation to verification backend (e.g. z3, prusti)
- evaluate the verification and the expressiveness of the refinement language on minimal examples

```
fn push_all(
    a: &mut Vec<i32>,
    b: &Vec<i32>) {
      ..
}

fn client() -> i32 {
  let mut a = vec![1, 2];
  let b = vec![2, 3];
  push_all(a, b);
  return a[5]; // type error!
}
```

Related Work

- Property Types in Java[1]
  - Adaptation of Refinement Types for Java
  - Limited to immutable (`final`) subset
- Prusti[2]
  - Heavy-Weight functional verification for rust with semantics for `unsafe`.
  - Seperation Logic
- Async Liquid Types[3]
  - for Refinement Types + Mutability in OCaml
  - cannot take advantage of Rust's Ownership model
- RustHorn[4]
  - constrained horn clauses based verification
  - `&mut` handling interesting and should be adapted for LiquidRust

Literatur

[1] Florian Lanzinger. „Property Types in Java: Combining Type Systems and Deductive Verification“. Master Thesis. Karlsruher Institut für Technologie, Feb. 2021.

[2] Vytautas Astrauskas u. a. „Leveraging rust types for modular specification and verification“. In: *Proceedings of the ACM on Programming Languages* 3 (OOPSLA 10. Okt. 2019), S. 1–30. ISSN: 2475-1421. DOI: 10.1145/3360573. URL: https://dl.acm.org/doi/10.1145/3360573 (besucht am 23. 02. 2022).

[3] Johannes Kloos, Rupak Majumdar und Viktor Vafeiadis. „Asynchronous Liquid Separation Types“. In: (2015). Unter Mitarb. von Marc Herbstritt. Artwork Size: 25 pages Medium: application/pdf Publisher: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 25 pages. DOI: 10.4230/LIPICS.ECOOP.2015.396. URL: http://drops.dagstuhl.de/opus/volltexte/2015/5223/ (besucht am 27. 01. 2022).

[4] Yusuke Matsushita, Takeshi Tsukada und Naoki Kobayashi. „RustHorn: CHC-based verification for Rust programs“. In: *European Symposium on Programming*. Springer, Cham, 2020, S. 484–514.

Example Rust

```rust
/// Add `b` to end of Vec `as`
fn push(as: &mut Vec<i32>, b: i32) {..}
/// Get highest value. Returns `None` for empty Vec
fn maximum(as: &Vec<i32>) -> Option<i32> {..}

fn update_exam_tries(tries: &mut Vec<i32>, new_try: i32) {
  push(&mut tries, new_try);

  match maximum(ties) {
    None => panic!("impossible"), // `tries` must at least contain `new_try`
    Some(max_mark) if max_mark < 4    => println!("puh"),
    _              if tries.len() > 3 => println!("Härtefallantrag?"),
    _                                 => println!("versuch's nochmal"),
  }
}
```

Example Rust

```
/// Add `b` to end of Vec `as`
fn push(as: &mut ty!(
  { before : Vec<i32> | before.len() == l} ~~> { after: Vec<i32> | after.len() == l + 1 })
  , b: i32) {..}
/// Get highest value. Returns `None` for empty Vec
fn maximum(as: &Vec<i32>) -> { r: Option<i32> | r.is_none() <=> as.len() == 0 } {..}

fn update_exam_tries(tries: &mut Vec<i32>, new_try: i32) {
  push(&mut tries, new_try);

  match maximum(ties) {
    None => proof_unrechable!(),
    Some(max_mark) if max_mark < 4    => println!("puh"),
    _              if tries.len() > 3 => println!("Härtefallantrag?"),
    _                                 => println!("versuch's nochmal"),
  }
}
```

Blöcke
in den KIT-Farben

Greenblock
Standard (`block`)

Blueblock
= `exampleblock`

Redblock
= `alertblock`

Brownblock

Purpleblock

Cyanblock

Yellowblock

Lightgreenblock

Orangeblock

Grayblock

Contentblock
(farblos)

Auflistungen

Text

- Auflistung
  Umbruch
- Auflistung
  - Auflistung
  - Auflistung

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

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.

Literatur

Backup-Teil

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

[1] Florian Lanzinger. „Property Types in Java: Combining Type Systems and Deductive Verification“. Master Thesis. Karlsruher Institut für Technologie, Feb. 2021.

[2] Vytautas Astrauskas u. a. „Leveraging rust types for modular specification and verification“. In: *Proceedings of the ACM on Programming Languages* 3 (OOPSLA 10. Okt. 2019), S. 1–30. ISSN: 2475-1421. DOI: 10.1145/3360573. URL: https://dl.acm.org/doi/10.1145/3360573 (besucht am 23. 02. 2022).

[3] Johannes Kloos, Rupak Majumdar und Viktor Vafeiadis. „Asynchronous Liquid Separation Types“. In: (2015). Unter Mitarb. von Marc Herbstritt. Artwork Size: 25 pages Medium: application/pdf Publisher: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 25 pages. DOI: 10.4230/LIPICS.ECOOP.2015.396. URL: http://drops.dagstuhl.de/opus/volltexte/2015/5223/ (besucht am 27. 01. 2022).

[4] Yusuke Matsushita, Takeshi Tsukada und Naoki Kobayashi. „RustHorn: CHC-based verification for Rust programs“. In: *European Symposium on Programming*. Springer, Cham, 2020, S. 484–514.

Farbpalette

kit-green100 kit-green90 kit-green80 kit-green70 kit-green60 kit-green50 kit-green40 kit-green30 kit-green25 kit-green20 kit-green15 kit-green10 kit-green5

kit-blue100 kit-blue90 kit-blue80 kit-blue70 kit-blue60 kit-blue50 kit-blue40 kit-blue30 kit-blue25 kit-blue20 kit-blue15 kit-blue10 kit-blue5

kit-red100 kit-red90 kit-red80 kit-red70 kit-red60 kit-red50 kit-red40 kit-red30 kit-red25 kit-red20 kit-red15 kit-red10 kit-red5

kit-gray100 kit-gray90 kit-gray80 kit-gray70 kit-gray60 kit-gray50 kit-gray40 kit-gray30 kit-gray25 kit-gray20 kit-gray15 kit-gray10 kit-gray5

kit-orange100 kit-orange90 kit-orange80 kit-orange70 kit-orange60 kit-orange50 kit-orange40 kit-orange30 kit-orange25 kit-orange20 kit-orange15 kit-orange10 kit-orange5

kit-lightgreen100 kit-lightgreen90 kit-lightgreen80 kit-lightgreen70 kit-lightgreen60 kit-lightgreen50 kit-lightgreen40 kit-lightgreen30 kit-lightgreen25 kit-lightgreen20 kit-lightgreen15 kit-lightgreen10 kit-lightgreen5

kit-brown100 kit-brown90 kit-brown80 kit-brown70 kit-brown60 kit-brown50 kit-brown40 kit-brown30 kit-brown25 kit-brown20 kit-brown15 kit-brown10 kit-brown5

kit-purple100 kit-purple90 kit-purple80 kit-purple70 kit-purple60 kit-purple50 kit-purple40 kit-purple30 kit-purple25 kit-purple20 kit-purple15 kit-purple10 kit-purple5

kit-cyan100 kit-cyan90 kit-cyan80 kit-cyan70 kit-cyan60 kit-cyan50 kit-cyan40 kit-cyan30 kit-cyan25 kit-cyan20 kit-cyan15 kit-cyan10 kit-cyan5