Abusing reborrowing for fun, profit, and a safepoint garbage collector

Aapo Alasuutari, FOSDEM 2025

About me

- Work at Valmet Automation
  - Software architect for a browser based automation UI platform
  - TypeScript developer by day
- Avid choir singer
- OpenSource enthusiast, contributor, albatross
- Data-oriented design zealot
- Developing Nova JavaScript engine
  - Rust developer by night!

Quick refresh on lifetimes and borrowing in Rust

- Lifetime is a period of time / section of code
  - References are ways to access a data
- Three types of lifetimes:
  - Static
  - Owned
  - Generics
- Shared (&’a T): I am observing T for the duration of ‘a. No mutating!
- Exclusive: (&’mut T): I am mutating T for the duration of ‘a. No observing!

Reborrowing

- Every use of a reference is formally a reborrow for a shorter lifetime
  - Call of fn with **&'a mut T** reborrows for **&'b** where **'a: 'b**
  - If 'b escapes the call, 'a is used to determine if the escape is acceptable
- An exclusive borrow can reborrowed as shared
  - Call with **&mut T** a method that takes **&T** is perfectly okay
  - Reborrow for shorter lifetime means that if 'b doesn't escape call, **&mut T** can be used as exclusive again after the call
  - If 'b does escape the call, **&mut T** is "in use" for the escaped lifetime
- Reborrow "signature":
  - **fn reborrow(&'short (mut) T: 'long) -> T: 'short**

Challenge: A safe exact tracing safepoint garbage collector with unrooted values using lifetimes

- “Garbage collector”?
  - System to automatically determine which memory is unused and release it.
- “Tracing”?
  - Memory usage is determined by following references (“tracing”) and comparing traced objects to all objects. Unreachable objects can be released.
- “Exact”?
  - Tracing starts from static places and follows static references.
- “Safepoint”?
  - Garbage collection happens only at defined “safe points” in the program.
  - Exact safepoint collector: At safepoint, all values must be in statical places.
  - Unrooted values: Values on stack must be rooted before safepoint!
    - Lifetimes please!

Let’s look at code!

Can this be made nicer?

- **.reborrow(), .shared()** calls can be eliminated
  - “autoreborrow “ traits
- **.unbind()** calls for method parameters requires a new feature
  - “Interprocedural reborrowing”
- **.unbind().bind(shared)** calls for return values requires a new feature
  - “Safe to downgrade to shared exclusive references”
  - Ugh

Q&A

https://github.com/aapoalas/abusing-reborrowing

Nova JavaScript Engine

https://trynova.dev/

https://github.com/trynova/nova/