Skip to main content

Stack Overflow

1. About
2. Products
3. For Teams

1. Stack Internal Implement a knowledge platform layer to power your enterprise and AI tools.
2. Stack Data Licensing Get access to top-class technical expertise with trusted & attributed content.
3. Stack Ads Connect your brand to the world’s most trusted technologist communities.
4. Releases Keep up-to-date on features we add to Stack Overflow and Stack Internal.
5. About the company Visit the blog

Loading…

current community

Stack Overflow
help chat
- Meta Stack Overflow

your communities

Sign up or log in to customize your list.

more stack exchange communities

company blog

3. Log in
4. Sign up

AI Assist is now on Stack Overflow. Start a chat to get instant answers from across the network. Sign up to save and share your chats.

Home
Questions
AI Assist
Tags
Challenges
7. Chat
Articles
Users
Companies
Collectives

Communities for your favorite technologies. Explore all Collectives

How are lifetimes of struct arguments treated in calls to functions with lifetime parameters?

Ask Question

Asked 10 years, 6 months ago

Modified 10 years, 6 months ago

Viewed 489 times

5

This code passes the compiler (for clarification lifetimes are not elided):

```
struct Foo<'a> {
   _field: &'a i32,
}

fn test<'a, 'b, 'c>(_x: &'a mut Foo<'c>, _y: &'b bool) {  // case 1
}

fn main() {
   let f = &mut Foo { _field: &0 };
   {
       let p = false;
       test(f, &p);
   }
}
```

If I use `'b` instead of `'c` in `test`'s definition like so:

```
fn test<'a, 'b>(_x: &'a mut Foo<'b>, _y: &'b bool) {  // case 2
}
```

the code fails to compile ("p does not live long enough")!

What I would expect to happen at the call of `test` in case 2 is:

- `'a` is set to the actual lifetime of `f`,
- `'b` is set to the intersection of the `Foo`'s actual lifetime and &p's actual lifetime which is &p's lifetime,

and everything should be fine, as in case 1.

Instead, what actually seems to happen in case 2 is that `'b` is forced to become the lifetime of the `Foo` which is too big for &p's lifetime, hence the compiler error 'p does not live long enough'. True?

Even stranger (case 3): this only fails if `test` takes a &mut. If I leave the `<'b>` in, but remove the `mut` like so:

```
fn test<'a, 'b>(_x: &'a Foo<'b>, _y: &'b bool) {  // case 3
}
```

the code passes again.

Anyone to shed light on this?

Cheers.

- rust

Share

Improve this question

Follow

asked May 28, 2015 at 23:20

dacker

49722 silver badges1010 bronze badges

1

This was cross-posted to the Rust Users Forum

Shepmaster – Shepmaster

2015-05-29 11:48:01 +00:00

Commented May 29, 2015 at 11:48

Add a comment |

1 Answer

Sorted by: Reset to default

Highest score (default) Trending (recent votes count more) Date modified (newest first) Date created (oldest first)

4

Noting the difference with `mut` was a key observation. I think that it will make more sense if you change the type of the second argument and give one possible implementation:

```
fn test<'a, 'b>(_x: &'a mut Foo<'b>, _y: &'b i32) {
   _x._field = _y;
}
```

This function has the ability to mutate `_x`. That mutation also includes *storing a new reference* in `_field`. However, if we were able to store a reference that had a **shorter** lifetime (the intersection you mentioned), as soon as the inner block ended, the reference in the `Foo` would become invalid and we would have violated Rust's memory safety guarantees!

When you use an immutable reference, you don't have this danger, so the compiler allows it.

You have discovered an important thing - Rust doesn't always care what you do in the function. When checking if a function call is valid, only the type signature of the function is used.

I'm sure there's a fancy way of saying this using the proper terms like *contravariance* and *covariance*, but I don't know those well enough to use them properly! ^_^

Share

Improve this answer

Follow

edited May 29, 2015 at 11:49

answered May 29, 2015 at 3:59

Shepmaster

439k116116 gold badges1.3k1.3k silver badges1.5k1.5k bronze badges

Sign up to request clarification or add additional context in comments.

2 Comments

Add a comment

dacker

dacker Over a year ago

Thx a lot shepmaster! That really pins it down. I just received a similar hint from mkrasnenkov at the Rust user forum (not as detailed as yours) and was able to give a small summary of all 3 cases there.

2015-05-29T06:13:11.947Z+00:00

0

Reply

- Copy link

dacker

dacker Over a year ago

In case 1, because of the &mut, 'c gets Foo's original lifetime. 'b is independent and does no harm. In case 2, again because of the &mut, 'b gets Foo's original lifetime. This time, the &p is "too short" to fit into the &'b bool, hence "p does not live long enough". In case 3, 'b is allowed to be shortened to &p's lifetime because there is no danger of test mutating _field to a reference that is "shorter" than the original Foo. Hope, I understood correctly.

2015-05-29T06:18:49.91Z+00:00

2

Reply

- Copy link

Your Answer

Thanks for contributing an answer to Stack Overflow!

- Please be sure to *answer the question*. Provide details and share your research!

But *avoid* …

- Asking for help, clarification, or responding to other answers.
- Making statements based on opinion; back them up with references or personal experience.

To learn more, see our tips on writing great answers.

Sign up or log in

Sign up using Google

Sign up using Email and Password

Submit

Post as a guest

Name

Email

Required, but never shown

Post as a guest

Name

Email

Required, but never shown

Post Your Answer Discard

By clicking “Post Your Answer”, you agree to our terms of service and acknowledge you have read our privacy policy.

Start asking to get answers

Find the answer to your question by asking.

Ask question

Explore related questions

- rust

See similar questions with these tags.

- The Overflow Blog
  Introducing Stack Overflow AI Assist—a tool for the modern developer
  Treating your agents like microservices
- Featured on Meta
  Chat room owners can now establish room guidelines
  AI Assist is now available on Stack Overflow
  Policy: Generative AI (e.g., ChatGPT) is banned

Related

65

What is the correct way to use lifetimes with a struct in Rust?

7

How do lifetime bounds on structs work in Rust?

3

type parameter for function vs struct (lifetime issue)

2

Understanding lifetimes for parameterized structs in Rust

1

How to use lifetimes correctly in Rust structs

6

Lifetime error on struct referring to a parametrized function

2

Rust lifetime scoping in structs

0

Passing lifetime to a struct

0

Not clear how to correctly define lifetime for struct

2

Confusion around lifetimes and struct definitions

Hot Network Questions

- Who is Patrick and why is he referred to at anti-AfD demonstrations?
- How do we define time-ordering operations in QFT?
- Is there any way to restore hearts without consuming anything, without hot springs, and without shrines?
- What does Felix mean by telling Bond to pick a contact point while "standing up" in Diamonds Are Forever?
- A hilarious parody of Western movies where an entire town collapses into tunnels
- Explicit examples separating Wasserstein distances from Jensen–Shannon divergence?
- Why does black allow white to have a protected passed central pawn here?
- How to add blur inside subtracted element in Figma?
- Storm Sphere and Obstruction
- I can't add a loop cut on one face
- Does short range teleporting end floating disk?
- "Save as webpage, HTML Only" Fail while offiline, Firefox
- When did "doubt" meaning question or point of uncertainty fall out of use outside of Indian English?
- Calculating the Resistance in an Induction Heater
- Trying to understand UK tax brackets for salary above 150k
- is it possible to keep the space when using url in latex
- Ping sweep of IP subnets
- Swords and sandals movie where the ancient Greek hero fights in space
- How to prevent a batch file call from escaping a caret character?
- Is "We will review application starting from..." a deadline?
- Is Brian Ripley’s Pattern Recognition and NN book still an insightful read?
- Misunderstanding the lorentz transformation
- Correct website for Thailand ETA application form
- What's the point of swapping polarity between mod slots?

more hot questions

Question feed

Subscribe to RSS

Question feed

To subscribe to this RSS feed, copy and paste this URL into your RSS reader.

Stack Overflow

- Questions
- Help
- Chat

Business

- Stack Internal