cannot infer an appropriate lifetime for borrow expression due to conflicting requirements

Ask Question

Asked 2 years, 3 months ago

Active 2 years, 3 months ago

Viewed 524 times

1

```
pub struct FooStruct<'a> {
  pub bars: Vec<&'a str>,
}

pub trait FooTrait<'a> {
  fn getBars(&self) -> &'a Vec<&'a str>;
}

impl<'a> FooTrait<'a> for FooStruct<'a> {
  fn getBars(&self) -> &'a Vec<&'a str> {
    &self.bars // cannot infer an appropriate lifetime for borrow expression due to conflicting requirements
  }
}
```

Run it: https://play.rust-lang.org/?version=stable&mode=debug&edition=2018&gist=3211c32dd5b9244ff91777f1820ffed5

I do not understand where the requirement conflict comes from. Afaik there is no conflict since everything lives as long as the FooStruct lives.

reference rust lifetime

Share

Improve this question

Follow

asked Sep 29 '19 at 17:40

OoDeLally

534 ● 1 ● 3 ● 21

'a might be *longer* than the lifetime of the FooStruct . Then it would be unsound to have a &'a Vec<_> because the Vec would not last for the entire 'a .
– trent formerly cl
Sep 29 '19 at 17:47

Add a comment

1 Answer

Active Oldest Votes

3

✓

Let's take it apart:

```
pub struct FooStruct<'a> {
  pub bars: Vec<&'a str>,
}
```

FooStruct holds a container containing string slices with lifetime 'a . The container's lifetime corresponds to the lifetime of FooStruct .

```
pub trait FooTrait<'a> {
  fn getBars(&self) -> &'a Vec<&'a str>;
}
```

FooTrait wants getBars to return a reference to a container holding string slices with lifetime 'a . The lifetime of the returned reference should be 'a , too.

```
impl<'a> FooTrait<'a> for FooStruct<'a> {
  fn getBars(&self) -> &'a Vec<&'a str> {
    &self.bars
  }
}
```

Here, getBars returns a reference to self.bars which is a container of string slices with lifetime 'a . So far, so good.

- However, what is the lifetime of &self.bars ? It corresponds to the lifetime of self (i.e. the respective FooStruct).
- What is the lifetime of self ? It is 'self (an implicit lifetime).

However, the FooTrait requires that the returned reference lifetime is 'a , so that does not match FooTrait 's declaration.

One solution is to separate the lifetimes in FooTrait :

```
pub trait FooTrait<'a> {
  fn getBars<'s>(&'s self) -> &'s Vec<&'a str>;
}

impl<'a> FooTrait<'a> for FooStruct<'a> {
  fn getBars<'s>(&'s self) -> &'s Vec<&'a str> {
    &self.bars
  }
}
```

Share
Improve this answer
Follow

answered Sep 29 '19 at 17:51

phimuemue
31.8k ● 9 ● 76 ● 110

Thank you. My mistake was the wrong assumption that FooStruct<'a> or FooTrait<'a> implicitly binds 'a to self . Now I understand that it merely informs of the existence of such lifetimes. Making the lifetime of &self explicit indeed solved the problem.
– OoDeLally
Sep 29 '19 at 18:56

Add a comment

Your Answer

Post Your Answer

By clicking “Post Your Answer”, you agree to our terms of service, privacy policy and cookie policy

Not the answer you're looking for? Browse other questions tagged reference rust lifetime or ask your own question.

The Overflow Blog

- Sequencing your DNA with a USB dongle and open source code
- Don’t push that button: Exploring the software that flies SpaceX rockets and...

Featured on Meta

- Providing a JavaScript API for userscripts
- Congratulations to the 59 sites that just left Beta

Related

Passing local lifetime to satisfy trait

Cannot infer lifetime for autoref when calling mutable method from closure

Conflicting lifetime requirements when using a reference inside of a closure

Rust mutable iterator problem - how do I fix the 'cannot infer an appropriate lifetime' error?

Can I define a trait that has a generic function returning a trait object?

Hot Network Questions

How do I get the http endpoint to work for cardano-wallet?
Vizier of the Menagerie and cost reducing
Naruto fighting game with Hulk and Homer Simpson?
Without passport stamps, can my country's authorities still know what countries I've visited?
How to salvage bitter homemade mustard?
more hot questions

Question feed

STACK OVERFLOW

Questions
Jobs
Developer Jobs Directory
Salary Calculator
Help
Mobile

PRODUCTS

Teams
Talent
Advertising
Enterprise

COMPANY

About
Press
Work Here
Legal
Privacy Policy
Terms of Service
Contact Us
Cookie Settings
Cookie Policy

STACK EXCHANGE NETWORK

Technology
Culture & recreation
Life & arts
Science
Professional
Business
API
Data

Blog
Facebook
Twitter
LinkedIn
Instagram