



Products

How to make RefCell's Ref live long enough

[Log in](#) [sign up](#)

[Ask Question](#)

Asked 4 years, 9 months ago

Active 4 years, 9 months ago

Viewed 157 times



3



I'm working on an interpreter in Rust for a programming language. Everything was going fine until I decided to implement closures, which caused some massive headaches because now each closure value needs to have a mutable reference of the environment it was defined in. I finally got it to mostly work with `RefCell`, but I'm now running into one more error that I can't figure out how to solve.

```
error: `e` does not live long enough
--> src/interpreters.rs:163:23
   |
163 |         let val = e.lookup(&name);
   |                        ^ does not live long enough
...
169 |     }
   |     - borrowed value only lives until here
   |
note: borrowed value must be valid for the lifetime `b` as defined on the body at 147:93...
--> src/interpreters.rs:147:94
   |
147 | fn eval_expr<a, 'b, 'c>(ast: &'a Expr, env: &'b RefEnv<'b>) -> Result<Value<'b>, Error<'c>> {
   |                                ^
   |
error: aborting due to previous error
```

Here is the relevant code:

```
use std::collections::HashMap;
use std::cell::RefCell;
#[derive(Clone, Debug)]
pub enum Value<'a> {
    Number(f64),
    UserFunc(Definition, Environment<'a>),
}

#[derive(Clone, Debug)]
pub struct Definition;

pub enum Expr {
    Name(String),
}

// Nothing to do with the problem
pub enum Error {
    UndefinedName(String),
}

#[derive(Debug, Clone)]
pub struct Environment<'a> {
    current_frame: HashMap<String, Option<Value<'a>>>,
    prev: Option<&'a Environment<'a>>,
}
impl<'a> Environment<'a> {
    pub fn new() -> Environment<'a> {
        Environment {
            current_frame: HashMap::new(),
            prev: None,
        }
    }
    pub fn extend(bindings: Vec<(String, Value<'a>>),
        prev: Option<&'a Environment<'a>>)
        -> Environment<'a> {
    }
}
```

How can I change the code to make it compile?

[reference](#) [rust](#) [borrow-checker](#) [borrowing](#)

Share

Improve this

question

Follow

edited Mar 15 '17 at 14:57



Shepmaster

305k ● 59 ● 824 ● 1083

asked Mar 15 '17 at 14:11



BookOwl

Your privacy 378 ● 3 ● 11

By clicking "Accept all cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our [Cookie Policy](#).

Accept all cookies

Customize settings

Is cloning `v` an option? And do you understand what the issue would be if `v` was returned?

– [Matthieu M.](#)

Mar 15 '17 at 14:14

I tried inserting some clone calls (`let val = e.lookup(&name).clone();` and `Ok(v.clone())`), but I still get the error.

– BookOwl

Mar 15 '17 at 14:19

I know that returning `v` right now would be an error because the `Ref<Environment>` that it points to would be freed before it does.

– BookOwl

Mar 15 '17 at 14:23

Superb. Not quite sure about the issue yet, notably because of the lifetime dance with the environments...

– Matthieu M.

Mar 15 '17 at 14:43

Exactly. The closures need to have a reference to the environment they created them so that lexical scoping can work.

– BookOwl

Mar 15 '17 at 14:45

[Add a comment](#)

1 Answer

[Active](#) [Oldest](#) [Votes](#)



4



This isn't really a problem regarding `RefCell` or `Ref`. You have this method:

```
impl<a> Environment<a> {  
    pub fn lookup(&self, name: &str) -> Option<Option<Value>>;  
}
```

After [lifetime elision](#) is reversed, that is equivalent to:

```
impl<a> Environment<a> {  
    pub fn lookup<b, 'c>(&b self, name: &'c str) -> Option<Option<Value<b>>>;  
}
```

Which means that the `Value` is only guaranteed to contain values that live as long as the `Environment`. What you really want is to tie the `Value` to whatever the `Environment` references:

```
impl<a> Environment<a> {  
    pub fn lookup(&self, name: &str) -> Option<Option<Value<a>>>;  
}
```

This allows your example code to compile.

[Share](#)

[Improve this answer](#)

[Follow](#)

[edited Mar 15 '17 at 15:09](#)

[answered Mar 15 '17 at 15:03](#)



[Shepmaster](#)

305k ● 59 ● 824 ● 1083

Wow, thanks! I never would have guessed that that was the problem.

– BookOwl

Mar 15 '17 at 15:15

[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](#) [borrow-checker](#) [borrowing](#) 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

[What's the difference between the 'ref' and 'out' keywords?](#)

[How do I pass a variable by reference?](#)

[1174](#)

[How to copy a dictionary and only edit the copy](#)

[List changes unexpectedly after assignment. Why is this and how can I prevent it?](#)

[Factory method: instance does not live long enough](#)

[HashMap key does not live long enough](#)

[Value does not live long enough](#)






[borrowed value does not live long enough in loop](#)

[ref to 'static' does not live long enough?](#)

[1](#)

[Nesting Structs: "borrowed value does not live long enough"](#)

Hot Network Questions

-  [Bit Rot within LUKS Encryption](#)
 -  [Send Geometry nodes value into Shading tab](#)
 -  [Does saying "Keep it up" put me in an authoritative position?](#)
 -  [Is Elon Musk really exploiting a loophole to avoid taxes?](#)
 -  [Why is the light source not showing but light is being cast on the object](#)
- [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](#)

site design / logo © 2021 Stack Exchange Inc; user contributions licensed under [cc by-sa](#). rev 2021.12.22.41046