A mutable borrow inside immutable borrow

help

tastelessjolt December 30, 2017, 9:36am 1

Btw, correct me if I'm wrong in saying "A mutable borrow inside immutable borrow".
I have this code.

```
use std::collections::HashMap;

#[cfg(test)]
mod test {
   use super::Cacher;

   #[test]
   fn normal_usage() {
       use std::thread;
       use std::time::Duration;

       let mut cached_caller = Cacher::new (| arg : &str | {
           println!("calculating slowly...");
           thread::sleep(Duration::from_secs(2));
           String::from(arg).len()
       });

       assert_eq!(cached_caller.value("32"), 2);
       assert_eq!(cached_caller.value("agath"), 5);
       assert_eq!(cached_caller.value("32"), 2);
       assert_eq!(cached_caller.value("agath"), 5);
   }
}

pub struct Cacher<T, G, H>
   where H: Clone, T: Fn(G) -> H,
{
   mapping: HashMap<G, H>,
   calculation: T,
}

impl<T, G, H> Cacher<T, G, H>
   where H: Clone, T: Fn(G) -> H, G: std::cmp::Eq, G: std::hash::Hash, G: Clone {
   pub fn new(calculation: T) -> Cacher<T, G, H> {
       Cacher {
           mapping: HashMap::new(),
           calculation: calculation,
       }
   }

   pub fn value(&mut self, arg: G) -> H {
       let res = match self.mapping.get(&arg) {
           Some(v) => Some(v.clone()),
           None => None,
       };

       match res {
            Some(v) => v,
            None => {
               let arg_clone = arg.clone();
               let v = (self.calculation)(arg);
               let clone = v.clone();
               self.mapping.insert(arg_clone, clone);
               v
           }
       }
   }
}
```

Can I not do something like this directly?

```
    pub fn value(&mut self, arg: G) -> H {
       // let res = match self.mapping.get(&arg) {
       //     Some(v) => Some(v.clone()),
       //     None => None,
       // };

       match self.mapping.get(&arg) {
            Some(v) => v.clone(),
            None => {
               let arg_clone = arg.clone();
               let v = (self.calculation)(arg);
               let clone = v.clone();
               self.mapping.insert(arg_clone, clone);
               v
           }
       }
   }
```

This gives the error:

```
   Compiling closures v0.1.0 (file:///home/harshithg/dev/rust/closures)
error[E0502]: cannot borrow `self.mapping` as mutable because it is also borrowed as immutable
 --> src/lib.rs:53:17
  |
47 |         match self.mapping.get(&arg) {
  |              ------------ immutable borrow occurs here
...
53 |                 self.mapping.insert(arg_clone, clone);
  |                ^^^^^^^^^^^^ mutable borrow occurs here
...
56 |         }
  |        - immutable borrow ends here

error: aborting due to previous error

error: Could not compile `closures`.
```

Thank you

bjorn3 December 30, 2017, 1:13pm 2

This is a limitation of the current borrow checker. self.mapping is borrowed for the duration of the match. There is work done to make this possible (non lexical lifetimes (nll)).

2 Likes

jonh December 30, 2017, 2:37pm 3

You can

```
match self.mapping.get(&arg).clone()
```

emoon December 30, 2017, 2:53pm 4

A (not very nice) work-around is to do this also

```
let mut should_insert = false;

match self.mapping.get(&arg) {
       should_insert = true;
}

if should_insert {
       self.mapping.insert(arg_clone, clone);
}
```

tastelessjolt December 30, 2017, 3:07pm 5

This didn't work for me. Did it work for you?

This is what I did.

play.rust-lang.org

Rust Playground

A browser interface to the Rust compiler to experiment with the language

tastelessjolt December 30, 2017, 3:08pm 6

Yeah, this is a workaround, but I wanted to do it the Rust way.
I looks un-clean.

jonh December 30, 2017, 5:02pm 7

There is entry as alternative.

1 Like

vitalyd December 30, 2017, 5:08pm 8

In fact, you can see that it works with NLL: Rust Playground

Related topics

Topic	Replies	Views	Activity
Immutable borrow when using .get on hashmap help	5	3802	November 25, 2020
Need help with mutable and immutable use of *self help	3	258	October 20, 2023
Immutable borrow from mutable self prevents further immutable borrow of self help	5	1759	April 21, 2020
Lifetime is not dropped after `if let (x) { return x }` help	3	494	August 17, 2020
Dereferencing a borrow makes mutable variable immutable? help	4	438	January 12, 2023