



Products

What is the difference between `Some(&a) => a` and `Some(a) => *a` when matching an Option?

Log in Sign up

Ask Question

Asked 3 years, 6 months ago

Active 3 years, 6 months ago

Viewed 120 times



4



Why does this pass:

```
fn f(v: Vec<isize>)->(Vec<isize>, isize) {  
    match v.get(0) {  
        Some(&a) => (v, a),  
        _ => (v, 0)  
    }  
}
```

[Playground](#)

but this doesn't?:

```
fn f(v: Vec<isize>)->(Vec<isize>, isize) {  
    match v.get(0) {  
        Some(a) => (v, *a),  
        _ => (v, 0)  
    }  
}
```

[Playground](#)

```
error[E0505]: cannot move out of `v` because it is borrowed  
--> src/main.rs:7:21  
|  
6 |   match v.get(0) {  
|     - borrow of `v` occurs here  
7 |     Some(a) => (v, *a),  
|               ^ move out of `v` occurs here
```

[rust](#) [borrowing](#)

Share

Improve this question

Follow

edited Jun 26 '18 at 14:10



Tim Diekmann

6,286 • 10 • 33 • 57

asked Jun 26 '18 at 13:51



Yuki Ito

162 • 8

[Add a comment](#)

2 Answers

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



1



[v.get\(0\)](#) returns a reference to the element in the vector, so you are matching `&isize`. The `Vec` is now borrowed in the match arm.

In the first code snippet, you copy the `isize`, so the `Vec` isn't borrowed here. In the second snippet, the `Vec` is still borrowed, so you cannot move it out of scope.

However, you should consider to use `if let` or `unwrap_or`:

```
fn f(v: Vec<isize>)->(Vec<isize>, isize) {  
    let a = v.get(0).cloned();  
    (v, a.unwrap_or(0))  
}
```

Your privacy

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](#) [Playground](#) [Customize settings](#)

```
fn f(v: Vec<isize>)->(Vec<isize>, isize) {
  if let Some(&a) = v.get(0) {
    (v, a)
  } else {
    (v, 0)
  }
}
```

[Playground](#)

See also:

- [How do I not borrow an Option when matching?](#)
- [How do I borrow a reference to what is inside an Option<T>?](#)
- [Cannot move out of borrowed content](#)

Share

Improve this answer

Follow

edited Jun 26 '18 at 14:22

answered Jun 26 '18 at 14:15



Tim Diekmann

6,286 ● 10 ● 33 ● 57

[Add a comment](#)



0



In the first snippet, when you type `Some(&a)`, you do not borrow `v` because `a` is copied.

In the second case, `Some(a)` is of type `Option<isize>` so it holds a reference to `v`. When you try to move it, it triggers an error. If you copy it first, and then you return the pair, it works (you need the [NLL feature](#) however):

```
#![feature(nll)]

fn main() {
  println!("{}", f(vec![1]))
}

fn f(v: Vec<isize>)->(Vec<isize>, isize) {
  match v.get(0) {
    Some(a)=> {
      let a = *a; // v is no more borrowed
      (v, a)
    },
    _ => (v, 0)
  }
}
```

[Playground](#)

The borrow checker cannot be perfect, so you will often encounter some slightly inconsistent stuff.

Share

Improve this answer

Follow

edited Jun 26 '18 at 15:49



Tim Diekmann

6,286 ● 10 ● 33 ● 57

answered Jun 26 '18 at 14:09



Boethios

28.9k ● 10 ● 104 ● 147

[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 [rust](#) [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

Linked

[Cannot move out of borrowed content / cannot move out of behind a shared reference](#)

[How do I borrow a reference to what is inside an Option<T>?](#)

2

[How do I not borrow an Option when matching?](#)

Related

[What are the differences between Rust's 'String' and 'str'?](#)

[How to match a String against string literals?](#)

[Cannot move out of borrowed content / cannot move out of behind a shared reference](#)

[Cannot move out of borrowed content when borrowing a generic type](#)

[Variable binding: moving a &mut or borrowing the referent?](#)

[What is the difference between iter and into_iter?](#)

1

[Multiple borrows error when fetching and setting a value in Struct](#)

[why is rust 'pub fn func\(&'a mut self\)' considered "mutably borrowed" after run?](#)

Hot Network Questions

 [What was the plutonium for, that was stolen at the start of The Amazing Spider-Man 2?](#)

 [What does "Graecōs Argōs" in this sentence mean? \(LLpsI\)](#)

 [Is it possible to convert a taproot address into a native segwit address?](#)

 [How do I get the http endpoint to work for cardano-wallet?](#)

 [Are they already planning a successor to the JWST?](#)

[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