Expected bound lifetime parameter, found concrete lifetime when trying to pass function that returns reference

help

mihaivo December 3, 2019, 7:44am 1

I am trying to sort according to a function that returns a reference, but I am getting a very weird error:

```
error[E0271]: type mismatch resolving `for<'r> <F as std::ops::FnOnce<(&'r T,)>>::Output == _`
 --> src/../...rs:89:19
  |
89 |         self.data.sort_unstable_by_key(f);
  |                   ^^^^^^^^^^^^^^^^^^^ expected bound lifetime parameter, found concrete lifetime

error: aborting due to previous error
```

```
impl<T: Send> Partition<T> {
   pub fn sort_by_light_key<F, K>(&mut self, f: F)
       where F: Copy + Sync + FnMut(&T) -> &K,
             K: Ord
   {
       self.data.sort_unstable_by_key(f);
   }
}
```

I really don't understand what the problem is here so any help would be great...

1 Like

inejge December 3, 2019, 10:12am 2

The signature of `sort_unstable_by_key()` says `F: FnMut(&T) -> K`, whereas you have `... -> &K`. If you remove the `&` the minimal version of the code will compile. With that out of the way, how is `Copy + FnMut` supposed to work?

1 Like

mihaivo December 3, 2019, 11:38am 3

Well isn't the `&K` a restriction to a type `K`. It means it requires a reference of a type which is a type, so it should fit.

The reason why I want `&K` is because somewhere else it the code partitioning is done on the lambda function and I don't want to copy the data, rather just use a reference for performance reasons.

kornel December 3, 2019, 11:59am 4

The type system doesn't have the subtyping flexibility you expect from it. The sort function wants `F: FnMut(&T) -> K, K: Ord`, and only *exactly* this is allowed.

In your case it's (pseudo-syntax):

```
F: FnMut(&T) -> Z, Z: &K, K: Ord
```

which means value returned from the function (`Z`) is not comparable any more (no `Ord` for `Z`). Dereference of that value is comparable, but the code which uses this callback won't dereference it, and therefore has no way of comparing `&K`.

2 Likes

mihaivo December 3, 2019, 2:05pm 5

Thank you for your explanation.

mihaivo December 4, 2019, 7:06pm 6

So the reason why this is not working is:

kornel:

> The type system doesn't have the subtyping flexibility you expect from it. The sort function wants `F: FnMut(&T) -> K, K: Ord` , and only *exactly* this is allowed.
>
> In your case it's (pseudo-syntax):

`F: FnMut(&T) -> Z, Z: &K, K: Ord`

which means value returned from the function (`Z`) is not comparable any more (no `Ord` for `Z`). Dereference of that value is comparable, but the code which uses this callback won't dereference it, and therefore has no way of comparing `&K` .

However I found an alternative. You can use the `sort_unstable_by` function. This idea came after reading this stack overflow post:

```
self.data.sort_unstable_by(|a, b|  f(a).cmp(f(b))  );
```

system Closed March 3, 2020, 7:11pm 7

This topic was automatically closed 90 days after the last reply. New replies are no longer allowed.

Related topics

Topic		Replies	Views	Activity
Sort_by_key, and probably a simple lifetime issue help	5	1511	June 22, 2022	
Why does `slice::sort_by_key` uses `B` and not `&B` as the key? help	8	1851	January 12, 2023	
Can't understand this lifetime help	8	273	September 10, 2024	
Lifetime problem with "sort_unstable_by_key" help	4	1487	January 12, 2023	
Lifetime error among trait and closure help	3	195	June 19, 2024	