A mutable variable is used but it is already captured by a closure.

Erroneous code example:

```
fn inside_closure(x: &mut i32) {
    // Actions which require unique access
}

fn outside_closure(x: &mut i32) {
    // Actions which require unique access
}

fn foo(a: &mut i32) {
    let mut bar = || {
        inside_closure(a)
    };
    outside_closure(a); // error: cannot borrow `*a` as mutable because previous
                        //        closure requires unique access.
    bar();
}
```

This error indicates that a mutable variable is used while it is still captured by a closure. Because the closure has borrowed the variable, it is not available until the closure goes out of scope.

Note that a capture will either move or borrow a variable, but in this situation, the closure is borrowing the variable. Take a look at the chapter on Capturing in Rust By Example for more information.

To fix this error, you can finish using the closure before using the captured variable:

```
fn inside_closure(x: &mut i32) {}
fn outside_closure(x: &mut i32) {}

fn foo(a: &mut i32) {
    let mut bar = || {
        inside_closure(a)
    };
    bar();
    // borrow on `a` ends.
    outside_closure(a); // ok!
}
```

Or you can pass the variable as a parameter to the closure:

```
fn inside_closure(x: &mut i32) {}
fn outside_closure(x: &mut i32) {}

fn foo(a: &mut i32) {
    let mut bar = |s: &mut i32| {
        inside_closure(s)
    };
    outside_closure(a);
```

```
    bar(a);
}
```

It may be possible to define the closure later:

```
fn inside_closure(x: &mut i32) {}
fn outside_closure(x: &mut i32) {}

fn foo(a: &mut i32) {
    outside_closure(a);
    let mut bar = || {
        inside_closure(a)
    };
    bar();
}
```