A closure was used but didn't implement the expected trait.

Erroneous code example:

```
struct X;

fn foo<T>(_: T) {}
fn bar<T: Fn(u32)>(_: T) {}

fn main() {
    let x = X;
    let closure = |_| foo(x); // error: expected a closure that implements
                             //        the `Fn` trait, but this closure only
                             //        implements `FnOnce`
    bar(closure);
}
```

In the example above, `closure` is an `FnOnce` closure whereas the `bar` function expected an `Fn` closure. In this case, it's simple to fix the issue, you just have to implement `Copy` and `Clone` traits on `struct X` and it'll be ok:

```
#[derive(Clone, Copy)] // We implement `Clone` and `Copy` traits.
struct X;

fn foo<T>(_: T) {}
fn bar<T: Fn(u32)>(_: T) {}

fn main() {
    let x = X;
    let closure = |_| foo(x);
    bar(closure); // ok!
}
```

To better understand how these work in Rust, read the Closures chapter of the Book.