A closure or generator was constructed that references its own type.

Erroneous code example:

```
fn fix<F>(f: &F)
  where F: Fn(&F)
{
    f(&f);
}

fn main() {
    fix(&|y| {
        // Here, when `x` is called, the parameter `y` is equal to `x`.
    });
}
```

Rust does not permit a closure to directly reference its own type, either through an argument (as in the example above) or by capturing itself through its environment. This restriction helps keep closure inference tractable.

The easiest fix is to rewrite your closure into a top-level function, or into a method. In some cases, you may also be able to have your closure call itself by capturing a `&Fn()` object or `fn()` pointer that refers to itself. That is permitting, since the closure would be invoking itself via a virtual call, and hence does not directly reference its own *type*.