During a method call, a value is automatically dereferenced as many times as needed to make the value's type match the method's receiver. The catch is that the compiler will only attempt to dereference a number of times up to the recursion limit (which can be set via the `recursion_limit` attribute).

For a somewhat artificial example:

```
#![recursion_limit="4"]

struct Foo;

impl Foo {
    fn foo(&self) {}
}

fn main() {
    let foo = Foo;
    let ref_foo = &&&&&Foo;

    // error, reached the recursion limit while auto-dereferencing `&&&&&Foo`
    ref_foo.foo();
}
```

One fix may be to increase the recursion limit. Note that it is possible to create an infinite recursion of dereferencing, in which case the only fix is to somehow break the recursion.