The lifetime bound for this object type cannot be deduced from context and must be specified.

Erroneous code example:

```
trait Trait { }

struct TwoBounds<'a, 'b, T: Sized + 'a + 'b> {
    x: &'a i32,
    y: &'b i32,
    z: T,
}

type Foo<'a, 'b> = TwoBounds<'a, 'b, dyn Trait>;
```

When a trait object is used as a type argument of a generic type, Rust will try to infer its lifetime if unspecified. However, this isn't possible when the containing type has more than one lifetime bound.

The above example can be resolved by either reducing the number of lifetime bounds to one or by making the trait object lifetime explicit, like so:

```
trait Trait { }

struct TwoBounds<'a, 'b, T: Sized + 'a + 'b> {
    x: &'a i32,
    y: &'b i32,
    z: T,
}

type Foo<'a, 'b> = TwoBounds<'a, 'b, dyn Trait + 'b>;
```

For more information, see RFC 599 and its amendment RFC 1156.