A parameter type is missing an explicit lifetime bound and may not live long enough.

Erroneous code example:

```
// This won't compile because the applicable impl of
// `SomeTrait` (below) requires that `T: 'a`, but the struct does
// not have a matching where-clause.
struct Foo<'a, T> {
    foo: <T as SomeTrait<'a>>::Output,
}

trait SomeTrait<'a> {
    type Output;
}

impl<'a, T> SomeTrait<'a> for T
where
    T: 'a,
{
    type Output = u32;
}
```

The type definition contains some field whose type requires an outlives annotation. Outlives annotations (e.g., `T: 'a`) are used to guarantee that all the data in `T` is valid for at least the lifetime `'a`. This scenario most commonly arises when the type contains an associated type reference like `<T as SomeTrait<'a>>::Output`, as shown in the previous code.

There, the where clause `T: 'a` that appears on the impl is not known to be satisfied on the struct. To make this example compile, you have to add a where-clause like `T: 'a` to the struct definition:

```
struct Foo<'a, T>
where
    T: 'a,
{
    foo: <T as SomeTrait<'a>>::Output
}

trait SomeTrait<'a> {
    type Output;
}

impl<'a, T> SomeTrait<'a> for T
where
    T: 'a,
{
    type Output = u32;
}
```