An attempt was made to retrieve an associated type, but the type was ambiguous.

Erroneous code example:

```
trait MyTrait {type X; }

fn main() {
    let foo: MyTrait::X;
}
```

The problem here is that we're attempting to take the type of X from MyTrait. Unfortunately, the type of X is not defined, because it's only made concrete in implementations of the trait. A working version of this code might look like:

```
trait MyTrait {type X; }
struct MyStruct;

impl MyTrait for MyStruct {
    type X = u32;
}

fn main() {
    let foo: <MyStruct as MyTrait>::X;
}
```

This syntax specifies that we want the X type from MyTrait, as made concrete in MyStruct. The reason that we cannot simply use `MyStruct::X` is that MyStruct might implement two different traits with identically-named associated types. This syntax allows disambiguation between the two.