The built-in function traits are generic over a tuple of the function arguments. If one uses angle-bracket notation (`Fn<(T,), Output=U>`) instead of parentheses (`Fn(T) -> U`) to denote the function trait, the type parameter should be a tuple. Otherwise function call notation cannot be used and the trait will not be implemented by closures.

The most likely source of this error is using angle-bracket notation without wrapping the function argument type into a tuple, for example:

```
#![feature(unboxed_closures)]

fn foo<F: Fn<i32>>(f: F) -> F::Output { f(3) }
```

It can be fixed by adjusting the trait bound like this:

```
#![feature(unboxed_closures)]

fn foo<F: Fn<(i32,)>>(f: F) -> F::Output { f(3) }
```

Note that `(T,)` always denotes the type of a 1-tuple containing an element of type `T`. The comma is necessary for syntactic disambiguation.