An associated function for a trait was defined to be static, but an implementation of the trait declared the same function to be a method (i.e., to take a `self` parameter).

Erroneous code example:

```
trait Foo {
    fn foo();
}

struct Bar;

impl Foo for Bar {
    // error, method `foo` has a `&self` declaration in the impl, but not in
    // the trait
    fn foo(&self) {}
}
```

When a type implements a trait's associated function, it has to use the same signature. So in this case, since `Foo::foo` does not take any argument and does not return anything, its implementation on `Bar` should be the same:

```
trait Foo {
    fn foo();
}

struct Bar;

impl Foo for Bar {
    fn foo() {} // ok!
}
```