There are conflicting trait implementations for the same type.

Erroneous code example:

```
trait MyTrait {
    fn get(&self) -> usize;
}

impl<T> MyTrait for T {
    fn get(&self) -> usize { 0 }
}

struct Foo {
    value: usize
}

impl MyTrait for Foo { // error: conflicting implementations of trait
                       //        `MyTrait` for type `Foo`
    fn get(&self) -> usize { self.value }
}
```

When looking for the implementation for the trait, the compiler finds both the `impl<T> MyTrait for T` where T is all types and the `impl MyTrait for Foo`. Since a trait cannot be implemented multiple times, this is an error. So, when you write:

```
trait MyTrait {
    fn get(&self) -> usize;
}

impl<T> MyTrait for T {
    fn get(&self) -> usize { 0 }
}
```

This makes the trait implemented on all types in the scope. So if you try to implement it on another one after that, the implementations will conflict. Example:

```
trait MyTrait {
    fn get(&self) -> usize;
}

impl<T> MyTrait for T {
    fn get(&self) -> usize { 0 }
}

struct Foo;

fn main() {
    let f = Foo;

    f.get(); // the trait is implemented so we can use it
}
```