A trait was implemented on another which already automatically implemented it.

Erroneous code examples:

```
trait Foo { fn foo(&self) { } }
trait Bar: Foo { }
trait Baz: Bar { }

impl Bar for Baz { } // error, `Baz` implements `Bar` by definition
impl Foo for Baz { } // error, `Baz` implements `Bar` which implements `Foo`
impl Baz for Baz { } // error, `Baz` (trivially) implements `Baz`
impl Baz for Bar { } // Note: This is OK
```

When `Trait2` is a subtrait of `Trait1` (for example, when `Trait2` has a definition like `trait Trait2: Trait1 { ... }`), it is not allowed to implement `Trait1` for `Trait2`. This is because `Trait2` already implements `Trait1` by definition, so it is not useful to do this.