A non-default implementation was already made on this type so it cannot be specialized further.

Erroneous code example:

```
#![feature(specialization)]

trait SpaceLlama {
    fn fly(&self);
}

// applies to all T
impl<T> SpaceLlama for T {
    default fn fly(&self) {}
}

// non-default impl
// applies to all `Clone` T and overrides the previous impl
impl<T: Clone> SpaceLlama for T {
    fn fly(&self) {}
}

// since `i32` is clone, this conflicts with the previous implementation
impl SpaceLlama for i32 {
    default fn fly(&self) {}
    // error: item `fly` is provided by an `impl` that specializes
    //        another, but the item in the parent `impl` is not marked
    //        `default` and so it cannot be specialized.
}
```

Specialization only allows you to override `default` functions in implementations.

To fix this error, you need to mark all the parent implementations as default. Example:

```
#![feature(specialization)]

trait SpaceLlama {
    fn fly(&self);
}

// applies to all T
impl<T> SpaceLlama for T {
    default fn fly(&self) {} // This is a parent implementation.
}

// applies to all `Clone` T; overrides the previous impl
impl<T: Clone> SpaceLlama for T {
    default fn fly(&self) {} // This is a parent implementation but was
                             // previously not a default one, causing the error
}

// applies to i32, overrides the previous two impls
impl SpaceLlama for i32 {
```

```rust
    fn fly(&self) {} // And now that's ok!
}
```