A trait type has been dereferenced.

Erroneous code example:

```
# trait SomeTrait { fn method_one(&self){} fn method_two(&self){} }
# impl<T> SomeTrait for T {}
let trait_obj: &SomeTrait = &"some_value";

// This tries to implicitly dereference to create an unsized local variable.
let &invalid = trait_obj;

// You can call methods without binding to the value being pointed at.
trait_obj.method_one();
trait_obj.method_two();
```

A pointer to a trait type cannot be implicitly dereferenced by a pattern. Every trait defines a type, but because the size of trait implementers isn't fixed, this type has no compile-time size. Therefore, all accesses to trait types must be through pointers. If you encounter this error you should try to avoid dereferencing the pointer.

You can read more about trait objects in the Trait Objects section of the Reference.