The Unsize trait should not be implemented directly. All implementations of Unsize are provided automatically by the compiler.

Erroneous code example:

```
#![feature(unsize)]

use std::marker::Unsize;

pub struct MyType;

impl<T> Unsize<T> for MyType {}
```

If you are defining your own smart pointer type and would like to enable conversion from a sized to an unsized type with the DST coercion system, use `CoerceUnsized` instead.

```
#![feature(coerce_unsized)]

use std::ops::CoerceUnsized;

pub struct MyType<T: ?Sized> {
    field_with_unsized_type: T,
}

impl<T, U> CoerceUnsized<MyType<U>> for MyType<T>
    where T: CoerceUnsized<U> {}
```