A method was called on a raw pointer whose inner type wasn't completely known.

Erroneous code example:

```
# #![deny(warnings)]
# fn main() {
let foo = &1;
let bar = foo as *const _;
if bar.is_null() {
    // ...
}
# }
```

Here, the type of `bar` isn't known; it could be a pointer to anything. Instead, specify a type for the pointer (preferably something that makes sense for the thing you're pointing to):

```
let foo = &1;
let bar = foo as *const i32;
if bar.is_null() {
    // ...
}
```

Even though `is_null()` exists as a method on any raw pointer, Rust shows this error because Rust allows for `self` to have arbitrary types (behind the arbitrary_self_types feature flag).

This means that someone can specify such a function:

```
ignore (cannot-doctest-feature-doesnt-exist-yet) impl Foo {
fn is_null(self: *const Self) -> bool {         // do something
else     } }
```

and now when you call `.is_null()` on a raw pointer to `Foo`, there's ambiguity.

Given that we don't know what type the pointer is, and there's potential ambiguity for some types, we disallow calling methods on raw pointers when the type is unknown.