An “or” pattern was used where the variable bindings are not consistently bound across patterns.

Erroneous code example:

```
let x = (0, 2);
match x {
    (0, ref y) | (y, 0) => { /* use y */} // error: variable `y` is bound with
                                          //        different mode in pattern #2
                                          //        than in pattern #1
    _ => ()
}
```

Here, `y` is bound by-value in one case and by-reference in the other.

To fix this error, just use the same mode in both cases. Generally using `ref` or `ref mut` where not already used will fix this:

```
let x = (0, 2);
match x {
    (0, ref y) | (ref y, 0) => { /* use y */}
    _ => ()
}
```

Alternatively, split the pattern:

```
let x = (0, 2);
match x {
    (y, 0) => { /* use y */ }
    (0, ref y) => { /* use y */}
    _ => ()
}
```