Get a mutable reference to a struct inside an enum

help

russellw March 10, 2023, 4:37am 1

This is a followup to my post Accessing struct methods from their dyn trait binding . The problem is accessing methods for a specific node type when all you have is **Box<dyn Node>**,

Rather than wrapping the Nodes in Boxes I tried try wrapping in an enum:
enum Node {Chars(CharsNode), And(AndNode), Or(OrNode)... }

I've found I can get a reference to the contents which is in general good enough - the tree is immutable once it is built. During the build phase though there are a few times they may need to be mutable. Is there any way to get a mutable reference to a struct inside an enum? I've tried a bunch of different ways and have not stumbled across one that works.

If nothing else comes up I've implemented clone() on the Node structs so I can get a mutable copy and replace the original in the tree (the mut access only comes at one place, where the parent object is not yet wrapped in the enum). Still, it would be cleaner to modify in place, and I'd like to understand if this is possible.

2e71828 March 10, 2023, 8:26am 2

Something like this?

```
impl Node {
   fn as_mut_or(&mut self)->Option<&mut OrNode> {
       match self {
           Node::Or(or_node) => Some(or_node),
           _ => None
       }
   }
}
```

russellw March 10, 2023, 9:19am 3

2e71828:

```
impl Node {
   fn as_mut_or(&mut self)->Option<&mut OrNode> {
       match self {
           Node::Or(or_node) => Some(or_node),
           _ => None
       }
   }
}
```

Exactly like that, thanks. I really don't have a good feel for where the mut's, &'s, and lifetimes all belong.

steffahn March 10, 2023, 9:35am 4

FYI, it's a shortened syntax called "match ergonomics" that results in

- a mutable reference (`self`) being accepted in the match against `Node::Or`, and
- a mutable reference being produced implicitly from the result of that match

The equivalent explicit way to do the same thing would be either

```
impl Node {
   fn as_mut_or(&mut self)->Option<&mut OrNode> {
       match self {
           &mut Node::Or(ref mut or_node) => Some(or_node),
           _ => None
       }
   }
}
```

or

```
impl Node {
   fn as_mut_or(&mut self)->Option<&mut OrNode> {
       match *self {
           Node::Or(ref mut or_node) => Some(or_node),
           _ => None
       }
   }
}
```

2 Likes

system Closed June 8, 2023, 9:36am 5

This topic was automatically closed 90 days after the last reply. We invite you to open a new topic if you have further questions or comments.

Related topics

Topic	Replies	Views	Activity
How can I modify some struct behind an enum? help	12	3010	March 4, 2023
Rust matching on enum members behind mutable reference help	3	1177	April 18, 2022
Pattern match and type cast of a mut reference help	3	1907	January 12, 2023
Enum implementing function that mutates itself help	2	655	August 17, 2019
Implement generic struct with different method variants help	6	338	September 7, 2024