Rust & Typestates

José Duarte

December 29, 2020

Introduction

What are typestates?

In a nutshell, typestates can be thought of a way to constrain APIs as the program state evolves. More formally, typestates belong to the behavioral types category and are built on the idea of lifting state to the type level, since state becomes part of the type system, the compiler will be able to reason about state, effectively helping the developer track state and validate certain assumptions.

Why are typestates useful?

Diving deeper on why do typestates help the developer, I provide a simple yet classic example. Consider a stream, whether it be a file or a socket, to be read, the stream must first be open, only then can it be read and finally it must be closed.

```
1 Scanner s = new Scanner(System.in); // open the stream
2 s.nextLine();                       // read
3 s.close();                          // close the stream
4 s.nextLine();                       // IllegalStateException
```

In the example above the developer tries to read a line after closing the stream, this yields a `IllegalStateException` since you cannot read from a closed source.

The fact that this code compiles without warnings (even when `-Xlint:all` is used) is problematic, since the error can only be caught at runtime. While the presented example is simple, code running in production is not, and code paths that raise runtime errors may be untested until it reaches the hands of the user.

Using typestates solves the above problem by establishing a distinction between the open and closed `Scanner`, consider the following example:

```
1 Scanner[Open] s = new Scanner(System.in);   // open the stream
2 s.nextLine();                               // read
3 Scanner[Closed] s = s.close();              // close the stream
4 s.nextLine();                               // compile-time error
```

The compiler is now able to provide the developer with an error at compile-time since it now knows that the `Scanner` is closed and thus, it does not have a function `nextLine`.