Nightly Night: Generators

By Jonathan Louie

Topics

What are generators?

Why generators?

Challenges stabilizing generators

What Are Generators?

What are generators?

- **Suspendable and resumable functions**
- **Suspended using “yield” keyword**
- **Any closure containing the “yield” keyword becomes a generator**

Basic Example

https://play.rust-lang.org/?version=nightly&mode=debug&edition=2021&gist=73ccdb809a3c9f133d3f6ece487c6d4e

Generator Trait

```
pub trait Generator<R = ()> {
    type Yield;
    type Return;
    fn resume(
        self: Pin<&mut Self>,
        resume: R
    ) -> GeneratorState<Self::Yield, Self::Return>;
}

pub enum GeneratorState<Y, R> {
    Yielded(Y),
    Complete(R),
}
```

Generators vs. Coroutines

Generators

- Can return values when paused
- Suitable for iteration
- Implemented using Generator trait

Coroutines

- Can return values when paused and can receive values when resumed
- Not strictly suitable for iteration
- Also implemented using Generator trait, which is somewhat confusing (subject to change)

Why Generators?

Why Generators?

- Mostly meant to be used through async/await notation
- More convenient to write than an Iterator impl
- No implicit memory allocation
- Generators/Coroutines are translated to state machines internally by the compiler (“Stackless Coroutines”)

Current State of Generators

Why Are Generators Unstable?

- There are several unanswered design questions still
  - Syntax questions
  - Semantics questions

Syntax Issues

- Functions and blocks cannot currently be generators
  - Only closures containing yield keyword can be generators
  - This requires a keyword like async to support
  - Because a keyword for generators was not reserved in edition 2018, it will likely not be able to land until at least 2024
- What should the return type be?
  - Currently, only the type yielded by the state machine is exposed in the syntax
  - This is in line with async
  - This may be re-litigated, due to some people having doubts about this choice

Semantics Issues

- How to deal with “return” expressions and “?” in generators?
- Big problem: self-references
  - The Iterator trait is stable and does not support self-references
  - This cannot be fixed, even with Editions

References

https://doc.rust-lang.org/beta/unstable-book/language-features/generators.html

https://without.boats/blog/generators/

https://github.com/rust-lang/rfcs/blob/master/text/2033-experimental-coroutines.md