Nightly Night: GATs

By Jonathan Louie

Motivation

- **Much excitement:**

  https://www.reddit.com/r/rust/comments/k4vzvp/gats_on_nightly/

Why?

Breakdown

What are Associated Types?

Why Associated Types?

What are GATs?

Why GATs?

Limitations

Closing Thoughts

What are Associated Types?

Associated Types

- Types associated with a trait
- Restricts traits to a single implementation for a given type
- Example:
  https://play.rust-lang.org/?version=stable&mode=debug&edition=2018&gist=fd4e7385f2abce99411d5f8bc956a1ce
- More info:
  https://doc.rust-lang.org/book/ch19-03-advanced-traits.html#specifying-placeholder-types-in-trait-definitions-with-associated-types

Why Associated Types?

Why Associated Types?

https://play.rust-lang.org/?version=stable&mode=debug&edition=2018&gist=c761ec7cb3bc2704e52693d18e256679

What are GATs?

Generic Associated Types

- **Enabled using #![feature(generic_associated_types)]**
- **Associated types that are generic!**
- **https://github.com/rust-lang/rfcs/blob/master/text/1598-generic_associated_types.md**

```
trait Foo {
    type Bar<T>;

    fn baz<T>() -> Self::Bar<T>;
}
```

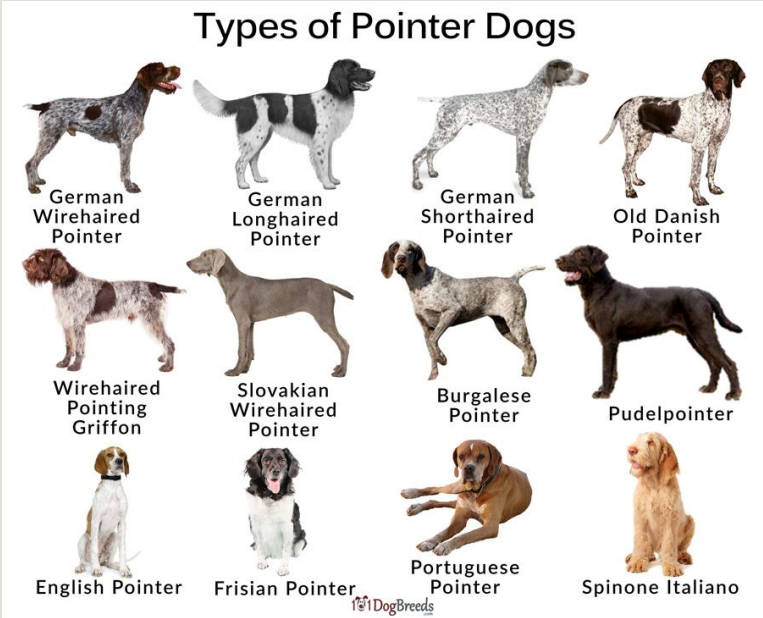
Why GATs?

Streaming Iterators

- **Return a reference to data owned by an iterator**
- **Can avoid hardcoding type of reference**
- **Example:**
  https://play.rust-lang.org/?version=nightly&mode=debug&edition=2018&gist=05d1e9bcb46b14f1657499186afe43ab
- **Workarounds:**
  http://lukaskalbertodt.github.io/2018/08/03/solving-the-generalized-streaming-iterator-problem-without-gats.html#the-problem

Abstracting Over Pointers

https://play.rust-lang.org/?version=nightly&mode=debug&edition=2018&gist=a95a593e222a709205b9541e75f9c0e2



Limitations

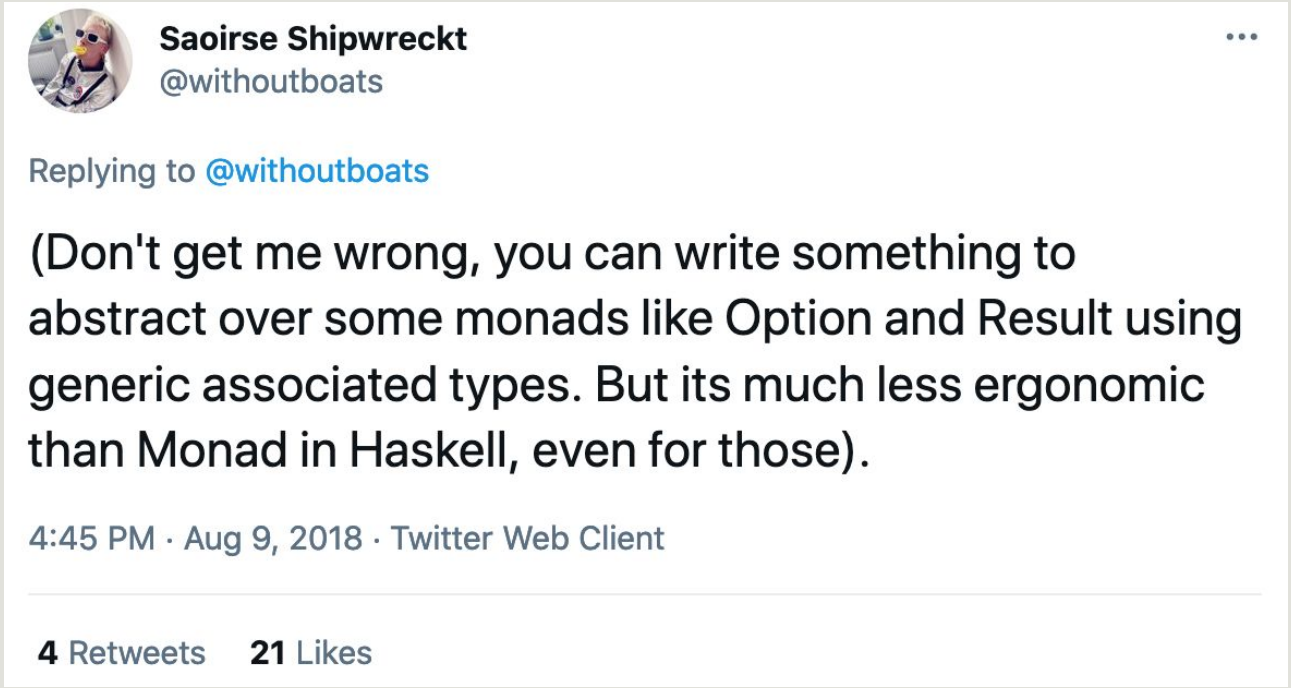
Functional Programming Abstractions

- **Relevant article:**
  https://www.fpcomplete.com/blog/monads-gats-nightly-rust/
- **ICE when implementing join/flatten**
- **mapM/traverse too difficult to implement**
- **Compiler can't know that there's a connection between Self and Wrapped<B>**

```
impl<A> Functor for MyOption<A> {
    type Unwrapped = A;
    type Wrapped<B> = Result<B, String>; // wut?
    // ...
}
```

Why Rust avoids having a Monad trait

https://twitter.com/withoutboats/status/1027702531361857536



Closing Thoughts

- Associated Types are useful for limiting user from implementing a trait for a type more than once
- GATs are just associated types that are generic
- GATs can make code more performant without requiring specialized hacky code
- GATs allow abstracting over generic types
- GATs won’t turn Rust into Haskell