Type Class: The Ultimate Ad Hoc

George Wilson

Data61/CSIRO

george.wilson@data61.csiro.au

July 31, 2017

Type classes are a language feature

- Haskell
- Eta
- Clean

Type classes are a language feature

- Haskell
- Eta
- Clean

or sometimes a design pattern

- Scala
- OCaml

Let's go beyond the basics of type classes

Polymorphism

Polymorphism is good

- greater reuse
- less repetition
- fewer names need inventing
- fewer possible implementations

Broadly speaking there are two major forms of polymorphism:

- *parametric* polymorphism
- *ad-hoc* polymorphism

Parametric polymorphism (sometimes called *generics*)

A function is parametricly polymorphic iff it can be called at different types, and always do the same thing.

```
reverse :: [a] -> [a]
```

Parametric polymorphism (sometimes called *generics*)

A function is parametricly polymorphic iff it can be called at different types, and always do the same thing.

```
reverse :: [a] -> [a]
```

Parametric polymorpism is great, but it's not the focus of this talk

Ad-hoc polymorphism