Week 2
Type classes in Haskell
(automatic type derivation)
Type classes

Automatic derivation

There are some instances that seem “obvious”, in the sense that we could create them with a simple algorithm.

For example, this sort of equality instance:

```haskell
data Day = Monday | Tuesday | ... | Sunday

instance Eq Day where
  Monday == Monday = True
  Tuesday == Tuesday = True
  ...
  _ == _ = False
```
Type classes

Automatic derivation

Or this sort, for a container, which relies on delegating to another instance

```haskell
data Sequence a = Link a (Sequence a) | Empty

instance (Eq a) => Eq (Sequence a) where
    Empty == Empty = True
    (Link a as) == (Link b bs) = (a == b) && (as == bs)
```
Type classes

Automatic derivation

Or, indeed, an instance of Show that just replicates the names of the constructors:

data Day = Monday | Tuesday | ... | Sunday

instance Show Day where
  show Monday = “Monday”
  ...

Type classes

Automatic derivation

Haskell has a mechanism to automatically create instances for a limited set of the classes declared in the Prelude.

There’s nothing here you couldn’t hand-write, and the automatic instances won’t always be what you want. But for lots of simple cases they work just fine.
Type classes

Automatic derivation

We can write:

```haskell
data Day = Monday | Tuesday | ... | Sunday
deriving (Eq, Show)
```

This works for:

- `Eq ( Monday == Monday )`
- `Ord ( Monday < Tuesday )`
- `Enum ( [Monday .. Friday] )`
- `Bounded ( minBound::Day => Monday )`
- `Read (read “Monday” => Monday)`
- `Show (show Monday => “Monday”)`
Thank you

glenn.Strong@scss.tcd.ie
https://scss.tcd.ie/Glenn.Strong/