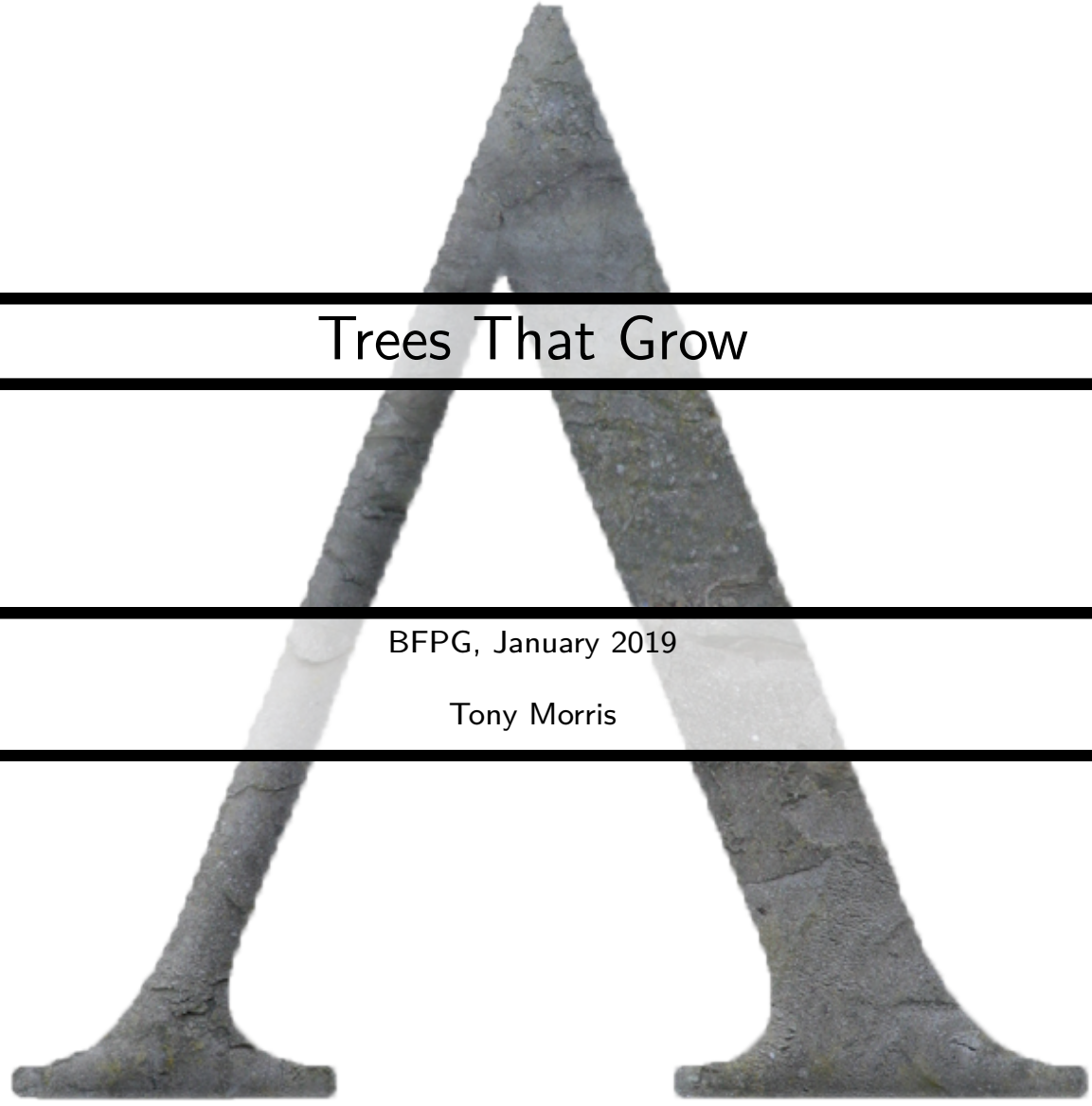


Trees That Grow

BFPG, January 2019

Tony Morris

Have you ever had this problem?

You are writing a data type, such as `data Aircraft`

So you decide that an `Aircraft` is the product of

- `Manufacturer`
- `Designation`
- `Registration`
- `Category`

and a `Category` is the sum of

- `Aeroplane`
- `Helicopter`
- `Gyroplane`
- `Airship`
- `...`

and on the `Aeroplane` constructor you have the product of

- `NonEmpty Propulsion`
- `...`

and a `Propulsion` is the product of

- `Engine`
- `MountPosition`

and an `Engine` is the product of

- `Manufacturer`
- `Designation`
- `EngineType`

and an `EngineType` is the sum of

- `ICE`
- `Jet`
- `Electric`
- `Rocket`

and an `ICE` is the product of

- `AirInduction`
- `FuelInduction`
- `Ignition`
- `ICEType`

and etc etc

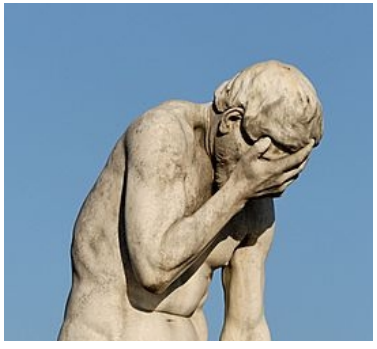
You write all your code in terms of this `Aircraft` data type

Then you create a database schema and store aircraft in it

. . . and then . . .

Project Manager: “can we just add an image to internal combustion engines?”

oh no



Your data type tree needs to grow

Trees That Grow is an approach to this extensibility

Trees That Grow

```
data ICE =
  ICE
    AirInduction
    FuelInduction
    Ignition
    ICEType
```

Trees That Grow

but we came prepared with Trees That (will probably) Grow

Trees That Grow

```
type family XICE x

data ICE_ x =
  ICE_
    AirInduction
    FuelInduction
    Ignition
    ICEType
    (XICE x)

type instance XICE () = ()
type ICE = ICE_ ()
```

Trees That Grow

pattern-matching becomes a pain in the undecorated case:

```
case ice of
  ICE_ air fuel ign typ () ->
```

Trees That Grow

so we write a pattern-synonym:

```
pattern ICE air fuel ign typ <-
          ICE_ air fuel ign typ _
  where ICE air fuel ign typ =
          ICE_ air fuel ign typ ()
```

Trees That Grow

using the pattern-synonym:

```
case ice of
  ICE air fuel ign typ ->
```

Trees That Grow

We can add an image to data ICE_

```
type instance XICE Image = Image
type ICE_Image = ICE Image
```

Trees That Grow

We can also add fields to sum types:

```
data Either a b =
  Left a
  | Right b
```

Trees That Grow

Either that grows

```
type family XEither x

data Either_ x a b =
  Left_ a (XEither x)
  | Right_ b (XEither x)

type instance XEither () = ()
type Either = Either_ ()
```

Trees That Grow

and add pattern-synonyms:

```
pattern Left a <- Left_ a _
  where Left a = Left_ a ()

pattern Right a <- Right_ a _
  where Right a = Right_ a ()
```

Trees That Grow

Note that there exists

```
xeither :: Lens (Either_ x a b) (XEither x)

data Either_ x a b =
  Left_ a (XEither x)
  | Right_ b (XEither x)
```

Trees That Grow

We can add constructors to sum types:

```
data These a b =
  This a
  | That b
  | Both a b
```

Trees That Grow

Theses that grow

```
type family XThese x

data These_ x a b =
  This_ a
  | That_ b
  | Both_ a b
  | XThese_ (XThese x)

type instance XThese Void = Void
type These = These_ Void
```

Trees That Grow

We could add pattern-synonyms, like before

Trees That Grow

We could also write the prisms

```
_This    :: Prism (These_ x a b) a
_That    :: Prism (These_ x a b) b
_Both    :: Prism (These_ x a b) (a, b)
_XThese  :: Prism (These_ x a b) (XThese x)
```

Trees That Grow extends to:[NPJ16]

- existential types
- GADTs

Analysis

This is my opinion of Trees That Grow

Alternatives

We already know that classy lenses (and prisms) work toward resolving this issue

Alternatives

```
class HasImage a where
  image :: Lens a Image

instance HasImage ICE_Image where
```

Alternatives

This requires creating a new data type:

```
data ICE_Image =
  ICE_Image
    ICE
    Image
```

One problem

One problem that I found with TTG is that the type-variable bubbles up the data type tree

One problem

My `Propulsion` data type has 18 type-variables

One problem

I start running out of names at 26

I didn't get to `Aircraft` (which is not at the top of the tree)

Therefore

I have come to prefer classy lenses and prisms.

Interesting Note

GHC plans to utilise TTG for its syntax tree, to achieve extensibility.

?

DATA 61

CSIRO

References

Shayan Najd and Simon Peyton-Jones, *Trees that grow. jucs (2016)*, 2016.