KU

BRYN MAWR
COLLEGE

Constrained Type Families

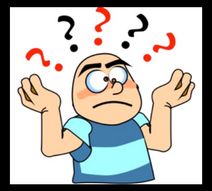
J. Garrett Morris
University of Edinburgh
University of Kansas
garrett@ittc.ku.edu

Richard A. Eisenberg
Bryn Mawr College
rae@cs.brynmawr.edu

Wednesday, 6 September 2017
ICFP
Oxford, UK

Two main contributions:

1. Discovering the problem:



GHC assumes all type families are total.

2. First type safety proof with

non-termination and non-linear patterns.

Simpler metatheory

But first:
An introduction to type families

In Haskell,
type families
=
type functions

results applicable to any language with
partiality and type-level computation

```
class Collects c where
  type Elem c
  empty  :: c
  insert :: Elem c → c → c
```

```
instance Collects [a] where
  type Elem [a] = a
  ...

instance Collects Word where
  type Elem Word = Bool
  ...
```

equality axiom

axiom is independent of class

```
type family Elem c
class Collects c where
  empty  :: c
  insert :: Elem c → c → c
```

```
data Z
data S n

type family Pred n
type instance Pred (S n) = n
type instance Pred Z     = Z
```

Haskellers leave no feature unabused

```
data Z
data S n
```

closed type family

```
type family Pred n where
  Pred (S n) = n
  Pred n     = n
```

Our new old idea:
Constrained Type Families

(originally suggested by
Chakravarty et al., ICFP ’05)

Definitions

A ground type has no type families.

A total type family, when applied to ground types, always equals some ground type.

Constrained Type Families

- All partial type families are associated
- Class constraint necessary to use an associated type family

Example

```
type family F a

thwack :: F a → Maybe a
thwack = ...
```

Example

```
class CF a where
  type F a

thwack :: CF a ⇒ F a → Maybe a
thwack = ...
```

The Totality Trap

Wat #1

```
type family F a

x = fst (5, ⊥ :: F Int)

Ok, modules loaded: Wat.
```

that's not a type!

Wat #1

```
class CF a where
  type F a
```

```
x = fst (5, ⊥ :: F Int)
```

that's not a type!

```
error: No instance for (CF Int)
```

Wat #2

```
type family EqT a b where
   EqT a a = Char
   EqT a b = Bool

f :: a → EqT a (Maybe a)
f _ = False
```

```
Wat.hs: error: ...
```

This has bitten!

In real life!

To people who use Haskell to make $$$!

Wat #2

```
type family EqT a b where
   EqT a a = Char
   EqT a b = Bool

f :: a → EqT a (Maybe a)
f _ = False

Ok, modules loaded: NoWat.
```

Why Wat #2?

```
type family Maybes a
type instance Maybes a =
                    Maybe (Maybes a)

f :: a → EqT a (Maybe a)
```

with $a \mapsto$ Maybes Int,

a = Maybe a!

Wat #3

```
type family Maybes a
type instance Maybes a =
                    Maybe (Maybes a)

justs = Just justs
Wat.hs: error:
• Cannot construct the
  infinite type:
    a ~ Maybe a
```

type inference fail.

Red herring:
“Just ban Maybes!”

Sometimes we
need loopy type families.

Wat #3

```
instance CMaybes a ⇒ CMaybes a where
  type Maybes a = Maybe (Maybes a)
```

```
justs = Just justs
Wat.hs: error:
• Cannot construct the
  infinite type:
    a ~ Maybe a
```

GHC does not infer impossible constraint.

The fundamental problem:

GHC today assumes all
type families are total.

Constrained type families fix this.

Why does this fix the wats?

The class constraint
restricts the type
family domain.

First known proof
of consistency with
non-linear patterns
and
non-termination.

Wrinkle:
Total type families

Total type families need
not be associated.

need better termination checker

Wrinkle: Backward compatibility

- Infer constraints
- New feature: Closed type classes
- Details in paper

Open question:
Forward compatibility

- Dependent types
- Termination checking
- Is Girard's paradox encodable?

Constrained type families:

- let us escape the totality trap
- prevent the usage of bogus types
- make closed type families more powerful
- simplify injective type families
- remove an unnecessary feature
- simplify the metatheory
- allow us to prove type safety

KU

BRYN MAWR
COLLEGE

Constrained Type Families

J. Garrett Morris
University of Edinburgh
University of Kansas
garrett@ittc.ku.edu

Richard A. Eisenberg
Bryn Mawr College
rae@cs.brynmawr.edu

Wednesday, 6 September 2017
ICFP
Oxford, UK