An existential-aware `DeriveFoldable`

Ryan Scott

✉ rgscott@indiana.edu

github.com/RyanGlScott

August 30, 2015

The current situation (GHC 7.10.2)

```
data Plain a = Plain Int a [a]
  deriving (Functor, Foldable, Traversable) ✓
```

```
data Expr a where
    EInt  :: Int -> Expr Int
    EAdd  :: Expr Int -> Expr Int -> Expr Int
    EBool :: Bool -> Expr Bool
    EIf   :: Expr Bool
              -> Expr a
              -> Expr a
              -> Expr a

deriving instance Functor     Expr ✗
deriving instance Foldable    Expr ✗
deriving instance Traversable Expr ✗
```

Why can't we derive `Foldable`?

```
class Functor f where
    fmap :: (a -> b) -> f a -> f b
```

```
instance Functor Expr where
    fmap f (EInt i) = EInt (f a) -- Can't conclude
                                 -- EInt b ~ EInt Int!
```

✗

```
class Foldable t where
    foldMap :: Monoid m => (a -> m) -> t a -> m
    foldr   :: (a -> b -> b) -> b -> t a -> b
    ...
```

```
instance Foldable Expr where
    foldMap f (EInt i) = f i -- This typechecks. Hm...
```

?

Actually, we can!

```haskell
data Expr a where
  EInt  :: Int -> Expr Int
  EAdd  :: Expr Int -> Expr Int -> Expr Int
  EBool :: Bool -> Expr Bool
  EIf   :: Expr Bool -> Expr a -> Expr a -> Expr a

instance Foldable Expr where
  foldMap f (EInt i)     = f i
  foldMap f (EAdd e1 e2) = foldMap f e1 <> foldMap f e2
  foldMap f (EBool b)    = f b
  foldMap f (EIf c t f') = foldMap f c <> foldMap f t <> foldMap f f'
```

But...

```haskell
data G a where
  G1 ::               Int -> G Int
  G2 :: a ~ Int => Int -> G a
  G3 :: b ~ Int => b   -> G Int
  G4 :: a ~ Int => a   -> G a
```

Compromise

- We only fold over a constructor argument if it *syntactically* mentions the last type parameter.

```
data G a where
  G1 ::            Int -> G Int
  G2 :: a ~ Int => Int -> G a
  G3 :: b ~ Int => b   -> G Int
  G4 :: a ~ Int => a   -> G a

instance Foldable G where
  foldMap _ G1{}   = mempty
  foldMap _ G2{}   = mempty
  foldMap _ G3{}   = mempty
  foldMap f (G4 i) = f i
```

- Slated to land in GHC 7.12 (8.0?)

https://ghc.haskell.org/trac/ghc/ticket/10447