

Deriving-via

BALDUR BLÖNDAL,
ANDRES LÖH, Well-Typed LLP
RYAN SCOTT, Indiana University

We present a new Haskell language extension that miraculously solves all problems in generic programming that ever existed.

ACM Reference Format:

Baldur Blöndal, Andres LöH, and Ryan Scott. 2017. Deriving-via. 1, 1 (November 2017), 2 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

"These types we write down they're not just names for data representations in memory, they're tags that queue in mathematical structures that we exploit."¹

1 INTRODUCTION

It is common folklore that `Monoids` can be lifted over `Applicatives`,

```
instance (Applicative f, Monoid a) => Monoid (f a) where
  empty  :: f a
  mempty = pure empty
  mappend :: f a -> f a -> f a
  mappend = liftA2 mappend
```

Conor McBride calls this "routine programming" using `Monoid` and `Applicative` as building blocks.²

But this instance is undesirable for multiple reasons (TODO: more reasons, rewrite)

- It overlaps with every `Monoid` instance over an applied type.
- "Structure of the `f` is often considered more significant than that of `x`."³
- It may not be the desired `Monoid`: Some constructors have an 'inherent monoidal structural', most notably the *free monoid* (lists: `[a]`) where we prioritize the list structure and not that of the elements.

Lists are in fact an instance of a wholly separate way of defining `Monoids` based on `Alternative`

```
instance Alternative f => Monoid (f a) where
  empty  :: f a
  mempty = empty
```

¹Taken from unknown position: <https://www.youtube.com/watch?v=3U3lV5VPmOU>

²<http://strictlypositive.org/Idiom.pdf>

³Much of this is stolen from Conor: <https://personal.cis.strath.ac.uk/conor.mcbride/so-pigworker.pdf>

Authors' addresses: Baldur Blöndal; Andres LöH, Well-Typed LLP; Ryan Scott, Indiana University.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted by ACM, Inc., provided that the copies are not made for general distribution, for advertising or promotional purposes, for creating new collective works, or for resale. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2017 Copyright held by the owner/author(s).

XXXX-XXXX/2017/11-ART

<https://doi.org/10.1145/nnnnnnn.nnnnnnn>

```
mappend :: f a -> f a -> f a  
mappend = (<|>)
```

2 EXAMPLES

3 FORMALISM

4 ADVANCED USES

4.1 Generalized GeneralizedNewtypeDeriving

4.2 DeriveAnyClass

5 LIMITATIONS, CONCLUSIONS AND FUTURE WORK

6 RELATED WORK