(/)

CODE WORDS – ISSUE THREE (/ISSUES/THREE)

The algebra (and calculus!) of algebraic data types

Joel Burget (/about#Joel Burget)

Note: This article assumes some introductory Haskell knowledge.

Introduction

Just as algebra is fundamental to the whole of mathematics, algebraic data types (ADTs) are fundamental to many common functional programming languages. They're the primitives upon which all of our richer data structures are built, including everything from sets, maps, and queues, to bloom filters (http://book.realworldhaskell.org /read/advanced-library-design-building-a-bloom-filter.html) and neural networks (http://hackage.haskell.org/package/hnn).

Algebraic data types and mathematical algebra have some similar looking operations. In fact, it's common in type theory to use the algebraic notation to define algebraic data types, rather than the Haskell-style data type declarations:

Haskell	**Math**	**Notes**