The Algebra of Algebraic Data Types

Chris Taylor
November 2012

Abusing the algebra of algebraic data types - why does this work?

86

52

The 'algebraic' expression for algebraic data types looks very suggestive to someone with a background in mathematics. Let me try to explain what I mean.

Having defined the basic types

- Product `•`
- Union `+`
- Singleton `X`
- Unit `1`

and using the shorthand X^2 for `X•X` and `2X` for `X+X` et cetera, we can then define algebraic expressions for e.g. linked lists

`data List a = Nil | Cons a (List a)` ↔ `L = 1 + X • L`

and binary trees:

`date Tree a = Nil | Branch (Tree a) (Tree a)` ↔ $T = 1 + X \cdot T^2$

Now, my first instinct as a mathematician is to go nuts with these expressions, and try to solve for `L` and `T`. I could do this through repeated substitution, but it seems much easier to abuse the notation horrifically and pretend I can rearrange it at will. For example, for a linked list:

“Haskell's algebraic data types are named such since they correspond to an ***initial algebra*** *in* ***category theory****, giving us some laws, some operations and some symbols to manipulate.”*

– Don Stewart

*“Haskell's algebraic data types are named such since they correspond to an initial algebra in category theory, giving us some **laws**, some **operations** and some **symbols** to manipulate.”*

– Don Stewart

Algebra

Symbols

Operations

Laws

Algebra

Symbols	0, 1, 2, x, y, z, ...
Operations	+, –, x, ÷, ...
Laws	$0 + x = x$, ...

Algebra

Symbols	Types (`()`, `Int`, `Bool`, `...`)
Operations	Type constructors (`Maybe`, `Either`)
Laws	?

Algebra

Symbols	*Things*
Operations	*Ways to make new things*
Laws	*Rules the things follow*

Prelude

```
{-# LANGUAGE EmptyDataDecls
           , TypeOperators #-}
```

One

```
data Unit = Unit
```

One

```
data Unit = Unit
```

```
data () = ()
```

Addition

```
data a :+ b = AddL a | AddR b
```

Addition

```
data a :+ b = AddL a | AddR b

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

Multiplication

```
data a :* b = Mul a b
```

Multiplication

```
data a :* b = Mul a b
```

```
data (a,b) = (a,b)
```

Zero

"In general, an algebraic type specifies a sum of one or more alternatives, where each alternative is a product of zero or more fields.

It might have been useful to permit a sum of zero alternatives, which would be a completely empty type, but at the time the value of such a type was not appreciated."

– Hudak, Hughes, Peyton Jones, Wadler

Zero

```
data Void
```

Two

```
type Two = Unit :+ Unit
```

Two

```
type Two = Unit :+ Unit

data Bool = False | True
```

Notation

`Void`	=>	0
`Unit`	=>	1
`Bool`	=>	2
`Addition`	=>	$a + b$
`Multiplication`	=>	$a \cdot b$

Laws

$0 + x = x$

```
Either Void x ≅ x
```

Laws

$$0 \cdot x = 0$$

`(Void,x) ≅ Void`

Laws

$$1 \cdot x = x$$

`((),x) ≅ x`

Laws

$$x + y = y + x$$

`Either x y ≅ Either y x`

Laws

$$x \bullet y = y \bullet x$$

`(x,y) ≅ (y,x)`

Functions

```
data a -> b = ?
```

Functions

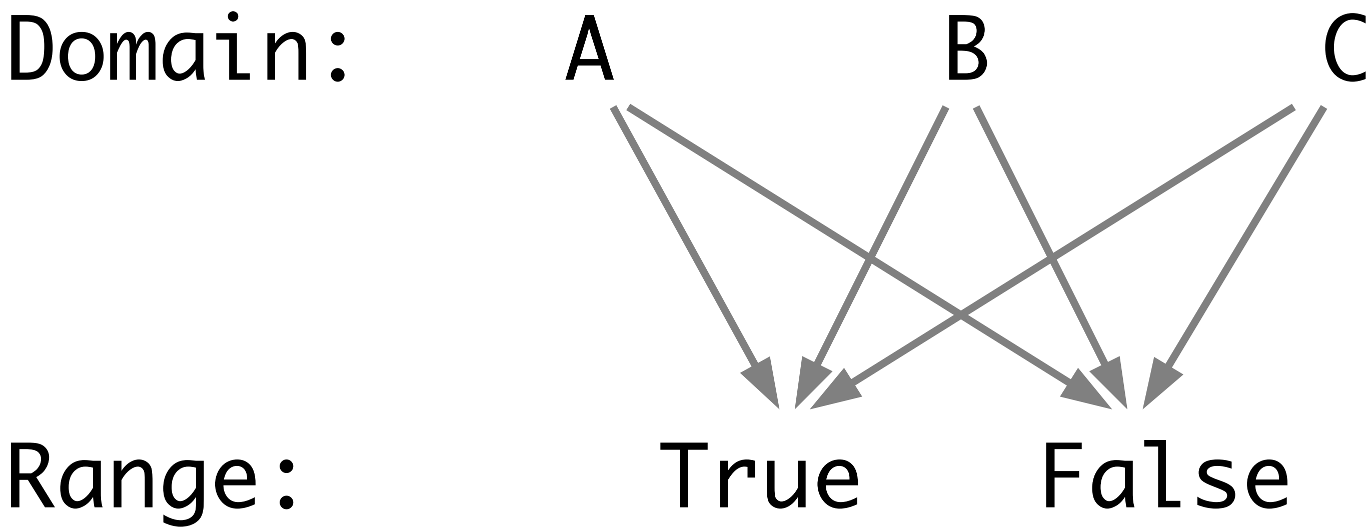
Domain:	True	False
Range:	True	False

Functions

```
Domain:        True   False

Range:         True   False
```

Functions



Functions

$a \ \text{->}\ b \qquad \text{<=>} \qquad b^a$

Laws

$$1^a = 1$$

```
a -> () ≅ ()
```

Laws

$$a^1 = a$$

$$() \to a \cong a$$

Laws

$$(b \cdot c)^a = b^a \cdot c^a$$

```
a -> (b,c) ≅ (a -> b,a -> c)
```

Laws

$$c^{ba} = (c^b)^a$$

```
(a,b) -> c ≅ a -> b -> c
```

Lists

```
data List x = Nil | Cons x (List x)
```

Lists

```
data List x = Nil | Cons x (List x)
```

$$L(x) = 1 + x \cdot L(x)$$

Lists

```
data List x = Nil | Cons x (List x)
```

```
L = 1 + x L
```

Lists

```
data List x = Nil | Cons x (List x)
```

```
L = 1 + x L
L = 1 + x (1 + x L)
```

Lists

```
data List x = Nil | Cons x (List x)
```

$$L = 1 + x\,L$$
$$L = 1 + x\,(1 + x\,L)$$
$$L = 1 + x + x^2\,(1 + x\,L)$$

Lists

```
data List x = Nil | Cons x (List x)
```

$$L = 1 + x\,L$$

$$L = 1 + x\,(1 + x\,L)$$

$$L = 1 + x + x^2\,(1 + x\,L)$$

$$L = 1 + x + x^2 + x^3 + x^4 + \ldots$$

Lists

```
data List x = Nil | Cons x (List x)
```

```
L = 1 + x L
```

Lists

```
data List x = Nil | Cons x (List x)
```

```
L = 1 + x L
L (1 - x) = 1
```

Lists

```
data List x = Nil | Cons x (List x)
```

```
L = 1 + x L
L (1 - x) = 1
L = 1 / (1 - x)
```

Lists

```
data List x = Nil | Cons x (List x)
```

$$L = 1 + x\,L$$

$$L\,(1 - x) = 1$$

$$L = 1 / (1 - x)$$

$$L = 1 + x + x^2 + x^3 + x^4 + \ldots$$

Trees

```
data Tree x = Tip | Node (Tree x) x (Tree x)
```

Trees

```
data Tree x = Tip | Node (Tree x) x (Tree x)
```

$$T = 1 + x\,T^2$$

Trees

```
data Tree x = Tip | Node (Tree x) x (Tree x)
```

$$T = 1 + x\,T^2$$

$$x\,T^2 - T + 1 = 0$$

Quadratic Formula (Interlude)

$$ax^2 + bx + c = 0$$

Quadratic Formula (Interlude)

$$ax^2 + bx + c = 0$$

$$x = (-b \pm \sqrt{(b^2 - 4ac)}) / 2a$$

Trees

```
data Tree x = Tip | Node (Tree x) x (Tree x)
```

$$T = 1 + x\,T^2$$

$$x\,T^2 - T + 1 = 0$$

Trees

```
data Tree x = Tip | Node (Tree x) x (Tree x)
```

$$T = 1 + x\,T^2$$

$$x\,T^2 - T + 1 = 0$$

$$T = (1 - \sqrt{(1 - 4x)}) / 2x$$

Trees

```
data Tree x = Tip | Node (Tree x) x (Tree x)
```

$$T = 1 + x\, T^2$$

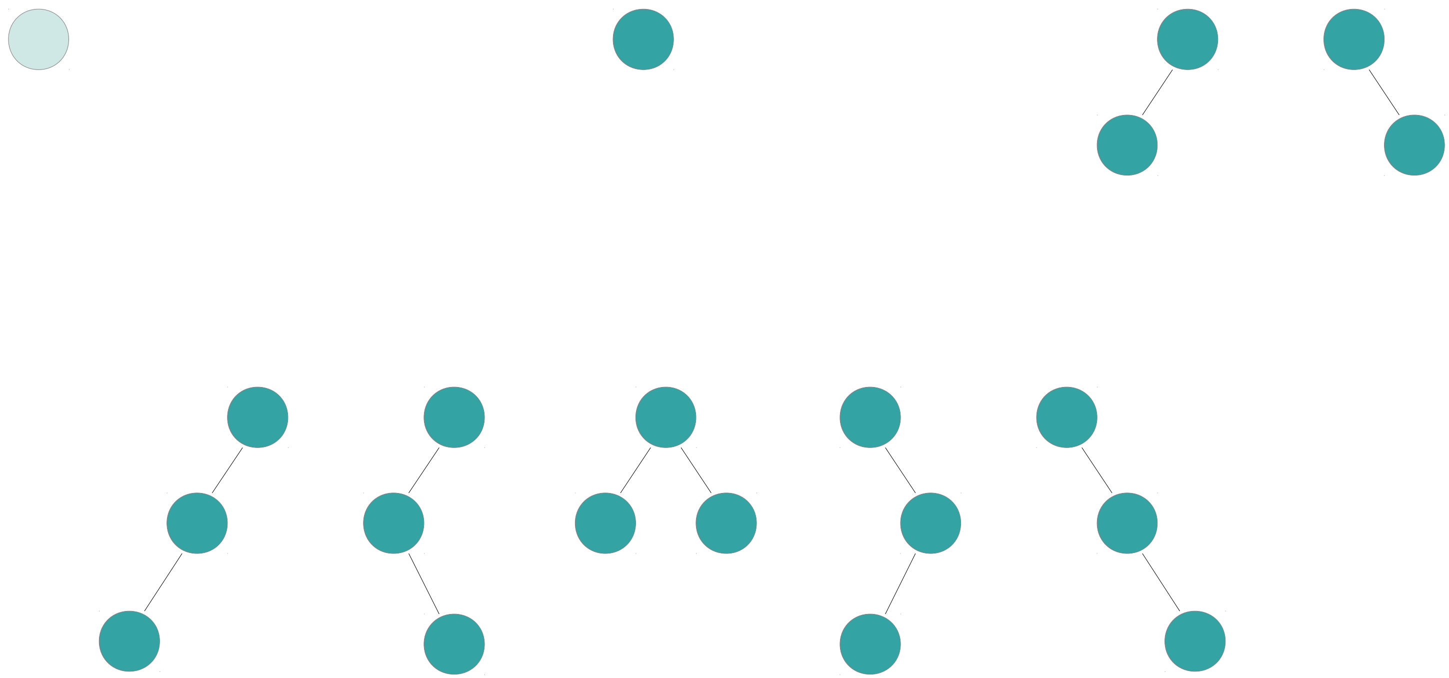
$$x\, T^2 - T + 1 = 0$$

$$T = (1 - \sqrt{(1 - 4x)}) / 2x$$

$$T = 1 + x + 2x^2 + 5x^3 + 14x^4 + \ldots$$

Trees

$$T = 1 + x + 2x^2 + 5x^3 + 14x^4 + \ldots$$



Zippers

Problem: Navigate and modify a data structure (e.g. a list) efficiently

```
> let x = [1, 2, 3, 4, 5, 6]
```

Zippers

Problem: Navigate and modify a data structure (e.g. a list) efficiently

```
> let x = [1, 2, 3, 4, 5, 6]
```

Solution: The zipper

```
> data Zip a = Zip [a] a [a]
```

Zippers

Problem: Navigate and modify a data structure (e.g. a list) efficiently

```
> let x = [1, 2, 3, 4, 5, 6]
```

Solution: The zipper

```
> let z =      Zip [2, 1] 3 [4, 5, 6]
> right z -- Zip [3, 2, 1] 4 [5, 6]
> right z -- Zip [4, 3, 2, 1] 5 [6]
```

One-Hole Contexts

One-hole context: Data structure with a hole

```
[1,2,3] * [5,6]
```

One-Hole Contexts

One-hole context: Data structure with a hole

```
[1,2,3] * [5,6]
```

Zipper: One-hole context paired with data

```
(4, [1,2,3] * [5,6])
```

One-Hole Contexts

x x

One-Hole Contexts

x	x
*	1

One-Hole Contexts

`(x,x)` x^2

One-Hole Contexts

`(x,x)` x^2

`(*,x) + (x,*)` $2x$

One-Hole Contexts

`(x,x,x)` x^3

One-Hole Contexts

`(x,x,x)` x^3

```
(*,x,x) + (x,*,x)
    + (x,x,*)
```

$3x^2$

One-Hole Contexts

x	=>	1
x^2	=>	$2x$
x^3	=>	$3x^2$

One-Hole Contexts

“The Derivative of a Regular Type is
its Type of One-Hole Contexts”

– Conor McBride

Type Calculus

∂ = “Take the derivative with respect to x”

Type Calculus

$$\partial 1 = 0$$

Type Calculus

$$\partial 1 = 0$$

$$\partial x = 1$$

Type Calculus

```
∂1          = 0

∂x          = 1

∂(f + g) = ∂f + ∂g
```

Type Calculus

$$\partial 1 = 0$$

$$\partial x = 1$$

$$\partial(f + g) = \partial f + \partial g$$

$$\partial(f \cdot g) = \partial f \cdot g + f \cdot \partial g$$

Type Calculus

$$
\begin{aligned}
\partial 1 &= 0 \\
\partial x &= 1 \\
\partial (f + g) &= \partial f + \partial g \\
\partial (f \cdot g) &= \partial f \cdot g + f \cdot \partial g \\
\partial (f(g)) &= \partial f(g) \cdot \partial g
\end{aligned}
$$

Type Calculus

```
L(x)  = 1 / (1 - x)
```

Type Calculus

$$L(x) = 1 / (1 - x)$$

$$\partial L(x) = 1 / (1 - x)^2$$

Type Calculus

$$L(x) = 1 / (1 - x)$$

$$\partial L(x) = 1 / (1 - x)^2$$

$$= L(x)^2$$

Type Calculus

$$T = 1 + x\,T^2$$

Type Calculus

$$T = 1 + x\ T^2$$

$$\partial T = T^2 + 2xT\ \partial T$$

Type Calculus

$$T = 1 + x\,T^2$$

$$\partial T = T^2 + 2xT\,\partial T$$

$$\partial T = T^2 / (1 - 2xT)$$

Type Calculus

$$T = 1 + x\,T^2$$

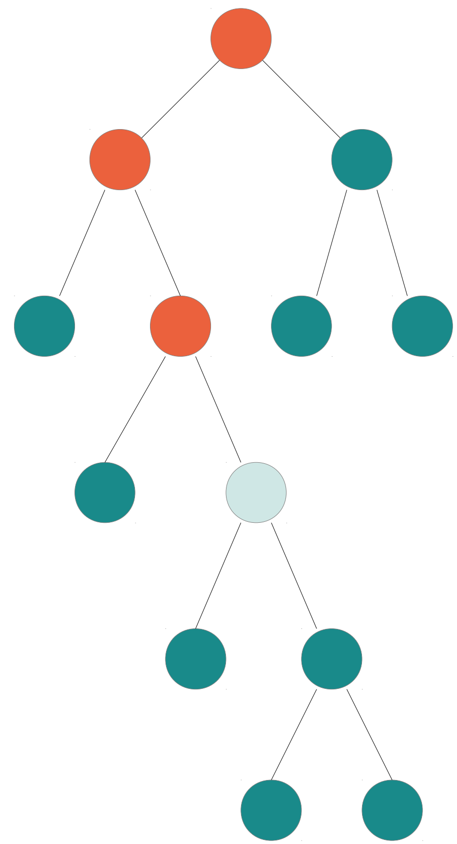
$$\partial T = T^2 + 2xT\,\partial T$$

$$\partial T = T^2 / (1 - 2xT)$$

$$\partial T = T^2 \bullet L(2xT)$$

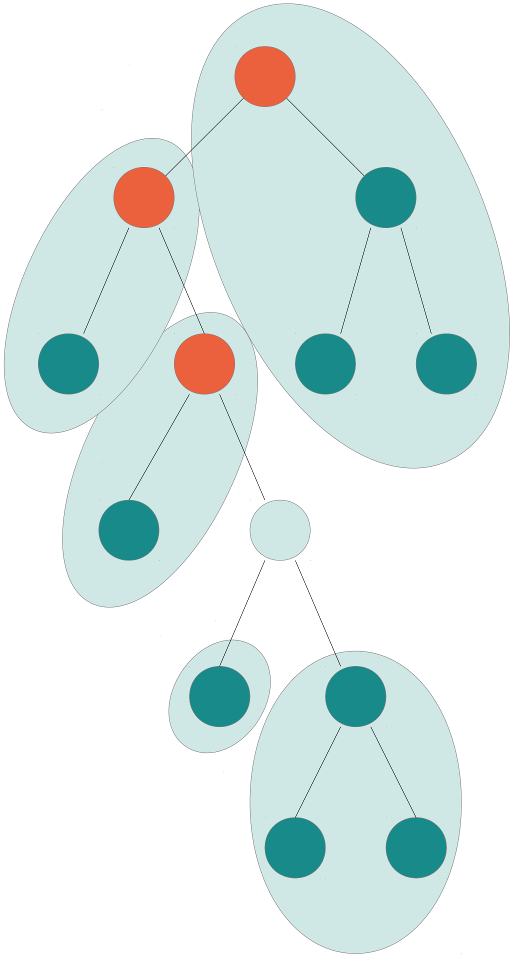
Type Calculus

$$\partial T = T^2 \cdot L(2xT)$$



Type Calculus

$$\partial T = T^2 \cdot L(2 \times T)$$



Non-Regular Types

Bags: No ordering

ULists: Unique elements

Sets: Unique elements, no ordering

Cyclic lists, Deques, etc.

Non-Regular Types

Set_n = "Sets of size n"

$Set_0(x) = 1$
$Set_1(x) = x$
$Set_2(x) = x\ (x-1)\ /\ 2$

Non-Regular Types

$$\mathrm{Set}_n(x) = x\ (x-1)\ \ldots\ (x-n+1)\ /\ n!$$

$$= x^{\underline{n}}\ /\ n!$$

Non-Regular Types

$$\text{Set}(x) = 1 + x + x^{\underline{2}}/2! + x^{\underline{3}}/3! + \ldots$$

Non-Regular Types

$$\mathrm{Set}(x) = 1 + x + x^2/2! + x^3/3! + \ldots$$

$$\Delta f(x) = f(x+1) - f(x)$$

Non-Regular Types

$$\mathrm{Set}(x) = 1 + x + x^{\underline{2}}/2! + x^{\underline{3}}/3! + \ldots$$

$$\Delta f(x) = f(x+1) - f(x)$$

$$\Delta \mathrm{Set}(x) = \mathrm{Set}(x)$$

Non-Regular Types

```
ΔSet(x)                    = Set(x)
```

Non-Regular Types

$$\Delta Set(x) = Set(x)$$

$$\Rightarrow Set(x+1) - Set(x) = Set(x)$$

Non-Regular Types

$$
\begin{aligned}
&\Delta \mathrm{Set}(x) &&= \mathrm{Set}(x) \\
\Rightarrow\ &\mathrm{Set}(x+1) - \mathrm{Set}(x) &&= \mathrm{Set}(x) \\
\Rightarrow\ &\mathrm{Set}(x+1) &&= 2\ \mathrm{Set}(x)
\end{aligned}
$$

Non-Regular Types

$$
\begin{aligned}
& \Delta \mathrm{Set}(x) && = \mathrm{Set}(x) \\
\Rightarrow\ & \mathrm{Set}(x+1) - \mathrm{Set}(x) && = \mathrm{Set}(x) \\
\Rightarrow\ & \mathrm{Set}(x+1) && = 2\ \mathrm{Set}(x) \\
\Rightarrow\ & \mathrm{Set}(x) && = 2^x
\end{aligned}
$$

Non-Regular Types

$$Set(x) = 2^x$$

```
Set x ≅ x -> Bool
```

More Type Algebra

github.com/chris-taylor/LondonHUG

More Type Algebra

Combinatorial Species

Andre Joyal, Brent Yorgey

Calculus of Types

Conor McBride, Dan Piponi (sigfpe)