A Introduction to an Algebra of labelled Graphs

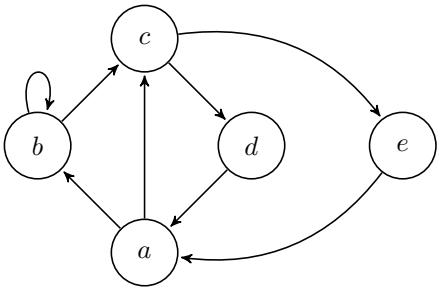
Anton Lorenzen

March 2018

The algebraic-graphs package, or just alga, is a library for constructing graphs in Haskell using a functional interface. This is a ground up introduction to alga. You should definitely read it from the beginning on though, even if you have already read the original functional pearl[1] since some definitions are different.

A Introduction to Algebraic Graphs

Think of any (finite) graph. As you probably know a graph can be represented as a matrix. Let's say we have the vertices $V = (a, b, c, d, e)$:



This is equivalent to the following matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We can decompose A into 25 matrices each containing one of A's elements and zeros everywhere else:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \dots$$

TODO: (also note that $A + A = A$, since 1 is the maximum element in each cell).

We can write each of these matrices as $FromVertex \rightarrow ToVertex$ in the case of a 1 and ε in the case of a zero. Therefore we have:

$$A = \varepsilon + a \rightarrow b + \dots$$

[1] https://github.com/snowleopard/alga-paper

We can therefore represent every matrix using the following data type:

```
newtype Vertex a = Vertex a

data Graph a = Empty
  | Connect Vertex Vertex
  | Overlay (Graph a) (Graph a)
```

Here `Connect` corresponds to $\rightarrow$, `Empty` to ε and `Overlay` to $+$. We require that `Overlay` is associative, commutative, has `Empty` as an identity element and is idempotent ($a \rightarrow b + a \rightarrow b = a \rightarrow b$) just like our matrix addition above.

Now, this isn't too revolutionary since we are basically just building an unbalanced binary tree of edges. But it gets more interesting if we allow `Connect` to connect more than simple vertices. For example, maybe we could define the following:

```
data Graph a = ε
  | Vertex a
  | (Graph a) → (Graph a)
  | (Graph a) + (Graph a)

[x,y,z] = map Vertex ["x", "y", "z"]
```

With the law:

```
x → (y + z) == (x → y) + (x → z)
(x + y) → z == (x → z) + (y → z)
```

That opens up the question what

```
x → y == x → (y + ε)
      == (x → y) + (x → ε)
```

means for the $x \rightarrow \varepsilon$ part? Obviously, it needs to be ε, you might say and you would end up with a $\rightarrow$ that looks pretty much like matrix multiplication. But alga took a different route and added another law, which gives it a different spin:

```
x → ε == ε → x == x
x → (y → z) == (x → (y + z)) + (y → z)
(x → y) → z == ((x + y) → z) + (x → y)
```

This law is what makes alga different from existing approaches, because it does it away with the difference between vertices and edges: The $\rightarrow$ constructor of the graph preserves the vertices contained in it. We can even derive this property from the laws:

$$\begin{aligned} x \to y &= (x \to y) \to \varepsilon \\ &= ((x + y) \to \varepsilon) + (x \to y) \\ &= x + y + (x \to y) \end{aligned}$$

Convince yourself that we can see x and y as arbitrary graphs from now on and not just as vertices. Also we will assume that $\to$ binds stronger than $+$.

So, what is a good intuition for $\to$? To come back to your A matrix above, we can write

$$(a + b) \to (d + e) = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Exercises:

- Show that the laws imply that $\to$ is associative.
- Show that the idempotence of $+$ and ε being the identity of $+$ follow from the other laws.
- Show that $x \to x \to x = x \to x$.
- Show we can derive the decomposition law from the original paper:

$$a \to b \to c = a \to b + a \to c + b \to c$$

About intuition

Now that you have worked through the exercises, you will probably agree that we can characterize a graph by the following laws:

- Addition: $+$ is associative and commutative.
- Identity: ε is the identity of $\to$.
- Distribution:

$$\begin{aligned} x \to (y + z) &= (x \to y) + (x \to z) \\ (x + y) \to z &= (x \to z) + (y \to z) \end{aligned}$$

- Extraction:

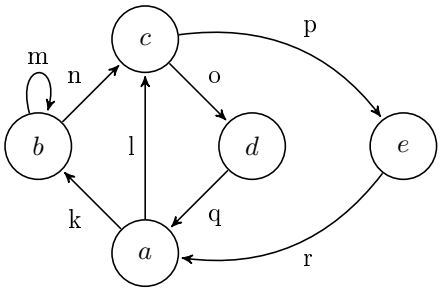
$$\begin{aligned} x \to (y \to z) &= (x \to (y + z)) + (y \to z) \\ (x \to y) \to z &= ((x + y) \to z) + (x \to y) \end{aligned}$$

The intuition I gave for these laws above might help you to understand what these laws mean in the contexts of graphs and indeed all implementations of these laws as adjacency maps or sets follow this intuition. However it falls short, if we go outside the context of graphs, so I would like to show you some statements you can't derive from the laws:

- *$+$ and $\to$ are distinct.* More specifically, every associative, commutative, idempotent operation with an identity fulfills the laws.
- *ε denotes the empty matrix.* Consider for example $a + b = a \to b = \min(a, b)$ with the identity being ∞.

Labelled Graphs

Let's go back to our graph from the beginning and add labels to the edges.



This is equivalent to the following matrix:

$$A = \begin{bmatrix} 0 & k & l & 0 & 0 \\ 0 & m & n & 0 & 0 \\ 0 & 0 & 0 & o & p \\ q & 0 & 0 & 0 & 0 \\ r & 0 & 0 & 0 & 0 \end{bmatrix}$$

Here you can think of $k, l, m, \ldots$ as variable names standing for strings, numbers or anything else. We will come back to which elements exactly we can use here in a minute.

Again we can decompose A into 25 matrices each containing one of A's elements and zeros everywhere else:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & k & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \ldots$$

And write each of these matrices as $FromVertex \xrightarrow{edge} ToVertex$. Therefore we have:

$$A = a \xrightarrow{0} a + a \xrightarrow{k} b + \ldots$$

We can therefore represent every labelled graph as the following data type:

```
data Graph l a = ε
  | Vertex a
  | (Graph a) -l-> (Graph a)
  | (Graph a) + (Graph a)

[x,y,z] = map Vertex ["x", "y", "z"]
[k,l,m,n] = ["k", "l", "m", "n"]
```

With the new laws:

- Addition: $+$ is associative and commutative.
- Identity: ε is the identity of $\xrightarrow{l}$.
- Distribution:

$$\begin{aligned} x \xrightarrow{k} (y + z) &= (x \xrightarrow{k} y) + (x \xrightarrow{k} z) \\ (x + y) \xrightarrow{k} z &= (x \xrightarrow{k} z) + (y \xrightarrow{k} z) \end{aligned}$$

- Extraction:

$$x \xrightarrow{k} (y \xrightarrow{l} z) = (x \xrightarrow{k} (y + z)) + (y \xrightarrow{l} z)$$
$$(x \xrightarrow{k} y) \xrightarrow{l} z = ((x + y) \xrightarrow{k} z) + (x \xrightarrow{l} y)$$

- Absorption:

$$x \xrightarrow{k} y + x \xrightarrow{l} y = x \xrightarrow{k+l} y$$

Exercise: Under which circumstances is the new $\xrightarrow{l}$ still associative?

Semirings

You have probably noticed that we used a new + operation on our labels above and as you might have guessed there are laws for it too, since we might run into contradictions otherwise:

- + is commutative, associative and idempotent.
- There is an element called 0 which acts as an identity for +.