

Sorting with Bialgebras MGS 2015 #3

Type K of keys, ordered.

$$L \times X = 1 + K \times X \quad L = L, \text{ but "ordered"}$$

$$\mu F \xrightarrow{\text{in}} F \mu F \quad \nu G \xleftarrow{\text{out}} G \nu G$$

Sorting has type $\mu L \rightarrow \nu L$.

Functions of type $\mu F \rightarrow \nu G$ include

unfold (fold $(f : FG, \mu F \rightarrow G, \mu F)$)

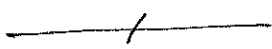
ad fold (unfold $(g : F \nu G \rightarrow G, F \nu G)$)

Lambek: $G \mu F \approx G F \mu F$ G in $^\circ$ of $FG, \mu F \rightarrow G F \mu F$

$F \nu G \approx F G \nu G$ $g \circ F$ out $^\circ$ of $FG, \nu G \rightarrow G F \nu G$

Apparently, we want distributive laws

$$FG \dashv\dashv GF$$



data $Lx = Nil | Cons Kx$

Consider

data $Lx = Nil | Cons Kx$

lsort = fold insert

insert = unfold ins

$ins :: L(\nu L) \rightarrow L(L(\nu L))$

$ins Nil = Nil$

$ins (Cons a (Out^\circ Nil)) = Cons a Nil$

$ins (Cons a (Out^\circ (Cons b x)))$

$\quad | a \leq b = Cons a (Cons b x)$

$\quad | otherwise = Cons b (Cons a x)$

lsort
then
lsort

Dually

$\text{bSort} = \text{unfold bubble}$

$\text{bubble} = \text{fold bub}$

$\text{bub} :: L(L(\mu L)) \rightarrow L(\mu L)$

$\text{bub Nil} = \underline{\text{Nil}}$

$\text{bub} (\text{Cons } a \text{ Nil}) = \text{Cons } a (\text{In Nil})$

$\text{bub} (\text{Cons } a (\text{Cons } b x))$

| $a \leq b$ = $\text{Cons } a (\text{In}(\text{Cons } b x))$

| otherwise = $\text{Cons } b (\text{In}(\text{Cons } a x))$

Note similarities between ins & bub .

$\text{swap} :: L(L x) \rightarrow L(L x)$

$\text{swap Nil} = \underline{\text{Nil}}$

$\text{swap} (\text{Cons } a \text{ Nil}) = \text{Cons } a \text{ Nil}$

$\text{swap} (\text{Cons } a (\text{Cons } b x))$

| $a \leq b$ = $\text{Cons } a (\text{Cons } b x)$

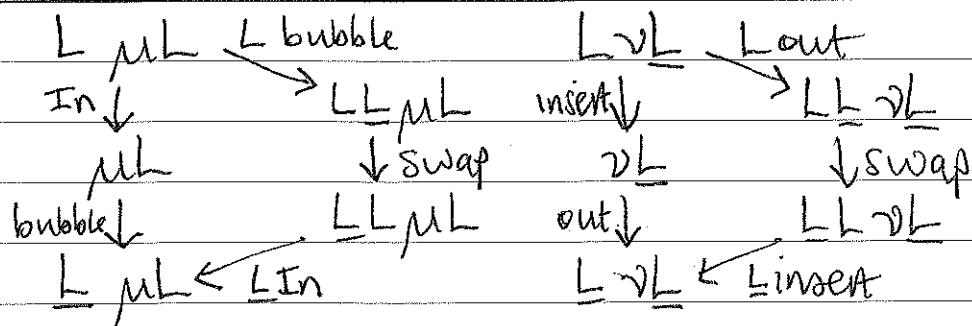
| otherwise = $\text{Cons } b (\text{Cons } a x)$

so

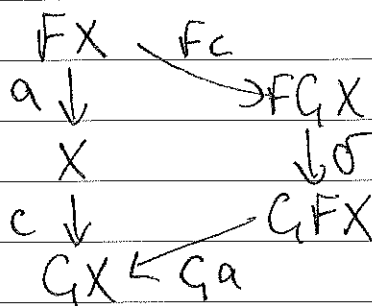
$\text{ins} = \text{swap} \circ L \text{ out}$

$\text{bub} = L \text{ In} \circ \text{swap}$

Note also, new polymorphic.

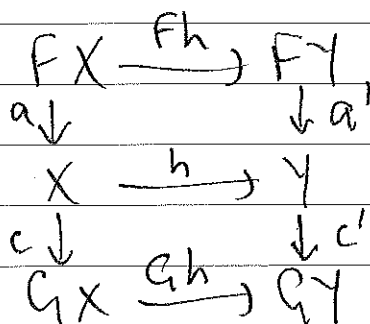


For $\sigma: FG \rightarrow GF$, a
 σ -bialgebra (X, a, c)
 is F -algebra (X, a)
 and G -coalgebra (X, c)
 such that:

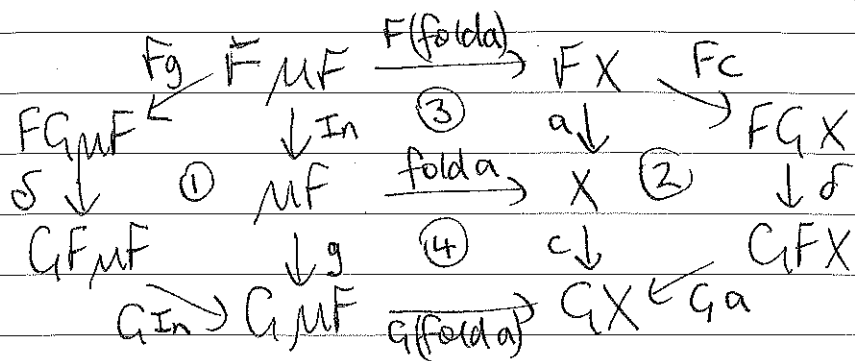


Homomorphism $h: (X, a, c) \rightarrow (Y, a', c')$ between
 σ -bialgebras is $h: X \rightarrow Y$ such that
 $h: (X, a) \rightarrow (Y, a')$ and $h: (X, c) \rightarrow (Y, c')$ homs.

Identity is a
 hom, and they
 compose; so they
 form a category.



Initial algebras lift to initial bialgebras



ie fold a is unique hom $(MF, In, g) \rightarrow (X, a, c)$
 $\textcircled{1}, \textcircled{2}$ commute by defn.

fold a is unique arrow to make $\textcircled{3}$ commute
 Need to check $\textcircled{4}$. Note $\textcircled{1}$ implies

$g: (MF, In) \rightarrow (GMF, GIn \circ \delta)$, so $g = \text{fold}(GIn \circ \delta)$
 Similarly $c: (X, a) \rightarrow (GFX, Ga \circ \delta)$
 so $c \circ \text{fold } a = \text{fold}(Ga \circ \delta)$.

Then $\textcircled{4}$ follows by fusion:

$G(\text{fold } a) \circ GIn \circ \delta = Ga \circ \delta \circ FG(\text{fold } a)$
 ie $(MF, In, \text{fold}(GIn \circ \delta))$ initial δ -bialgebra

Dually, unfold c is unique hom from (X, a, c)
 so $(\forall G, \text{unfold}(\delta \circ F \circ \text{out}), \text{out})$ is
 final δ -bialgebra.

Specifically, $F ::= L$, $G ::= L$, $\delta ::= \text{swap}$
 Then $(\underline{L}, \text{In}, \text{fold } \text{bub})$ initial bi-algebra
 $(\underline{L}, \text{unfold } \text{ins}, \text{out})$ final
 By initiality, $\text{fold}(\text{unfold } \text{ins})$ is
 unique hom between them. But by
 finality, so is $\text{unfold}(\text{fold } \text{bub})$.
 Hence they're equal — not just
 extensionally ("both sort") but
 intensionally ("same algorithm") too.
 cf diagrams on p138.

Also works for apo and para.

$\text{apo} ::= \text{functor } f \Rightarrow (a \rightarrow f(bf + a)) \rightarrow a \rightarrow \forall f$
 $\text{insert} = \text{apo } \text{ins}$

$\text{ins} ::= L(\underline{L}) \rightarrow L(\underline{L} + L \underline{L})$

$\text{ins } \text{Nil} = \text{Nil}$

$\text{ins } (\text{Cons } a (\text{In } \text{Nil})) = \text{Cons } a (\text{Lft } (\text{In } \text{Nil}))$

$\text{ins } (\text{Cons } a (\text{In } (\text{Cons } b \text{ xc})))$

$\quad | \text{as } b = \text{Cons } a (\text{Lft } (\text{In } (\text{Cons } b \text{ xc})))$

$\quad | \text{ow} = \text{Cons } b (\text{Rgt } (\text{Cons } a \text{ xc})))$

and dually...

para :: functor f => (f (mf x a) -> a) -> mf -> a
 select = para sel

sel :: L (ML x L ML) -> L ML

sel Nil = Nil

sel (Cons a (x, Nil)) = Cons a x

sel (Cons a (x, Cons b x'))

| a < b = Cons a x

| a > b = Cons b (In (Cons a x'))

Unify Make polymorphic: "swap & stop"

ins = swap o L out

$F_+ A = A + FA$

swap :: L (L X) -> L (L_+ X)

$F_x A = A x FA$

and unify

swap₂ :: L (L_x X) -> L (L_+ X)

swap₂ Nil = Nil

swap₂ (Cons a (x, Nil)) = Cons a (L_F (In Nil))

swap₂ (Cons a (x, Cons b x'))

| a < b = Cons a (L_F (In (Cons b x')))

| a > b = Cons b (Rgt (Cons a x'))

ins = swap₂ o L (id Δ out) ← invariant!

sel = L (id ∇ In) o swap₂ so replace by x

with more work, swap₂ :: L_+ L_x -> L_x L_+.