Chapter 6

Categorical equivalence of ornaments and relational algebras

Consider the AlgList datatype in **??** again. The way it is refined relative to List looks canonical, in the sense that any variation of List can be programmed as a special case of AlgList: we can choose whatever index set we want by setting the carrier of the algebra R, and by carefully programming R, we can insert fields into the list datatype that add more information or put restriction on fields and indices. For example, if we want some new information in the nil case, we can program R such that $R\ (\text{`nil}\ ,\ \blacksquare)\ x$ contains a field requesting that information; if, in the cons case, we need the targeted index $x$, the head element $a$, and the index $x'$ of the recursive position to be related in some way, we can program $R$ such that $R\ (\text{`cons}\ ,\ a\ ,\ x'\ ,\ \blacksquare)\ x$ expresses that relationship.

The above observation leads to the following general theorem: Let $O : \mathsf{Orn}\ e\ D\ E$ be an ornament from $D : \mathsf{Desc}\ I$ to $E : \mathsf{Desc}\ J$. Then there is a **classifying algebra** for O

$$clsAlg\ O : \mathbb{F}\ D\ (\mathsf{Inv}\ e) \rightsquigarrow \mathsf{Inv}\ e$$

such that there are isomorphisms

$$(j : J) \to \mu\ \lfloor algOD\ D\ (clsAlg\ O) \rfloor\ (e\ j\ ,\ \mathsf{ok}\ j) \cong \mu\ E\ j$$

That is, the algebraic ornamentation of D using the classifying algebra derived

from O produces a datatype isomorphic to $\mu\ E$, so intuitively the algebraic ornamentation has the same content as O. We may interpret this theorem as saying that algebraic ornamentation is "complete" for the ornament language: any relationship between datatypes that can be described by an ornament can be described up to isomorphism by an algebraic ornamentation. The name "classifying algebra" is explained after the examples below.

Examples (*classifying algebras for two ornaments*)**.** The following relational algebra R serves as a classifying algebra for the ornament $\lceil\ \mathit{OrdListOD}\ \rceil$ from lists to ordered lists:

$$
\begin{aligned}
&R\ :\ \mathbb{F}\ (\mathit{ListD}\ \mathit{Val})\ (\mathsf{Inv}\ !) \rightsquigarrow \mathsf{Inv}\ !\quad \text{-- } !\ :\ \mathit{Val} \to \top \\
&R\ (\text{`}\mathsf{nil}\quad ,\qquad\qquad \blacksquare)\ (\mathsf{ok}\ b)\ =\ \top \\
&R\ (\text{`}\mathsf{cons}\ ,\ x\ ,\ \mathsf{ok}\ b'\ ,\ \blacksquare)\ (\mathsf{ok}\ b)\ =\ (b \leqslant x) \times (b' \equiv x)
\end{aligned}
$$

The nil case says that the empty list can be mapped to any $\mathsf{ok}\ b$ (since any $b\ :\ \mathit{Val}$ is a lower bound of the empty list); for the cons case, where $x\ :\ \mathit{Val}$ is the head element of a non-empty list and $\mathsf{ok}\ b'$ is a possible result of folding the tail (i.e., $b'\ :\ \mathit{Val}$ is a lower bound of the tail), the list can be mapped to $\mathsf{ok}\ b$ if $b\ :\ \mathit{Val}$ is a lower bound of x and x is exactly b'. The type of proofs that a list xs folds to some $\mathsf{ok}\ b$ with R is isomorphic to $\mathsf{Ordered}\ b\ xs$, and hence

$$
\begin{aligned}
\mathsf{AlgList}\ \mathit{Val}\ R\ (\mathsf{ok}\ b)\ &\cong\ \Sigma[\ xs : \mathsf{List}\ \mathit{Val}\ ]\ (\![\ R\ ]\!)\ xs\ (\mathsf{ok}\ b) \\
&\cong\ \Sigma[\ xs : \mathsf{List}\ \mathit{Val}\ ]\ \mathsf{Ordered}\ b\ xs \quad \cong\ \mathsf{OrdList}\ b
\end{aligned}
$$

For another example, consider the ornament *NatD-ListD A*, for which we can use the following classifying algebra S:

$$
\begin{aligned}
&S\ :\ \mathbb{F}\ \mathit{NatD}\ (\mathsf{Inv}\ !) \rightsquigarrow \mathsf{Inv}\ !\quad \text{-- where } !\ :\ \top \to \top \\
&S\ (\text{`}\mathsf{nil}\quad ,\qquad \blacksquare)\ (\mathsf{ok}\ \blacksquare)\ =\ \top \\
&S\ (\text{`}\mathsf{cons}\ ,\ \mathsf{ok}\ \blacksquare\ ,\ \blacksquare)\ (\mathsf{ok}\ \blacksquare)\ =\ A
\end{aligned}
$$

The result of folding a natural number n with S is uninteresting, as it can only be $\mathsf{ok}\ \blacksquare$. The fold, however, requires an element of type A for each successor node it encounters, so a proof that n goes through the fold consists of n elements of type A and amounts to an inhabitant of $\mathsf{Vec}\ A\ n$. Thus

$$
\mu\ \lfloor \mathit{algOD}\ \mathit{NatD}\ S \rfloor\ (\blacksquare\ ,\ \mathsf{ok}\ \blacksquare)\ \cong\ \Sigma[\ n : \mathsf{Nat}\ ]\ (\![\ S\ ]\!)\ n\ (\mathsf{ok}\ \blacksquare)
$$

$$\cong \Sigma[\, n : \mathsf{Nat} \,]\ \mathsf{Vec}\ A\ n \qquad \cong \mathsf{List}\ A$$

This example helps to emphasise proof-relevance of our relational language, deviating from Bird and de Moor [1997], which (implicitly) adopts the traditional proof-irrelevant semantics of relations. □

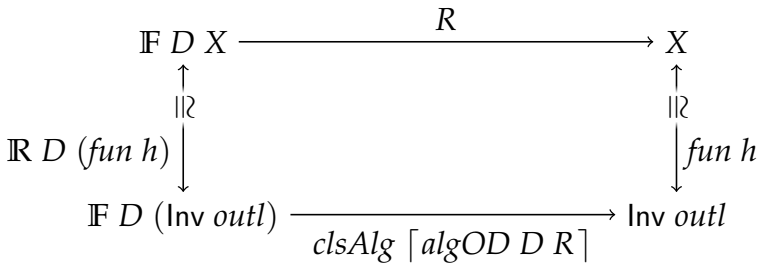
The completeness theorem brings up a nice algebraic intuition about inductive families. Consider the ornament from lists to vectors, for example. This ornament specifies that the type $\mathsf{List}\ A$ is refined by the collection of types $\mathsf{Vec}\ A\ n$ for all $n : \mathsf{Nat}$. A list, say $a :: b :: [] : \mathsf{List}\ A$, can be reconstructed as a vector by starting in the type $\mathsf{Vec}\ A\ \mathsf{zero}$ as $[]$, jumping to the next type $\mathsf{Vec}\ A\ (\mathsf{suc}\ \mathsf{zero})$ as $b :: []$, and finally landing in $\mathsf{Vec}\ A\ (\mathsf{suc}\ (\mathsf{suc}\ \mathsf{zero}))$ as $a :: b :: []$. The list is thus classified as having length 2, as computed by the fold function *length*, and the resulting vector is a fused representation of the list and the classification proof. In the case of vectors, this classification is total and deterministic: every list is classified under one and only one index. But in general, classifications can be partial and nondeterministic. For example, promoting a list to an ordered list is classifying the list under an index that is a lower bound of the list. The classification process checks at each jump whether the list is still ordered; this check can fail, so an unordered list would "disappear" midway through the classification. Also there can be more than one lower bound for an ordered list, so the list can end up being classified under any one of them. Compared with McBride's original functional algebraic ornamentation [2011], which can only capture part of this intuition about classification (namely those classifications that are total and deterministic), relational algebraic ornamentation allows partiality and nondeterminacy and thus captures the idea about classification in its entirety — a classification is just a relational fold computing the index that classifies an inhabitant. All ornaments specify classifications, and thus can be transformed into algebraic ornamentations.

There is a dual to the completeness theorem: every relational algebra is isomorphic to the classifying algebra for the algebraic ornament using the algebra. More precisely: Let $D : \mathsf{Desc}\ I$ be a description and $R : \mathbb{F}\ D\ X \rightsquigarrow X$

an algebra (where $X : I \to \mathsf{Set}$). There is a family of isomorphisms between $X\ i$ and $\mathsf{Inv}\ outl\ i$ for every $i : I$ (where $outl : \Sigma\ I\ X \to I$); call the forward direction of this family of isomorphisms $h : X \rightrightarrows \mathsf{Inv}\ outl$. Then we have

$$fun\ h \bullet R \simeq clsAlg\ \lceil algOD\ D\ R \rceil \bullet \mathbb{R}\ D\ (fun\ h)$$

or diagrammatically:



Together with the completeness theorem, we see that *algOD* and *clsAlg* are, in some sense, inverse to each other up to isomorphism. This suggests that we can seek to construct an **equivalence** between *SliceCategory* Orn $(I\ ,\ D)$ and a suitable category of D-algebras and homomorphisms by extending *algOD* and *clsAlg* to functors between the two categories and proving that the two functors are inverse up to (natural) isomorphism. This we do in Section 6.2, after we look at ornaments from a more semantic perspective in Section 6.1, where we also deal with some unresolved issues in **??**. The equivalence shows that ornaments and relational algebras are essentially the same entities, which lets us associate some ornamental and relational algebraic constructions in Section 6.3. Section 6.4 concludes with some discussion.

see if the above can be revised to be slightly more specific

6.1 Ornaments and horizontal transformations

In this section we aim to find a semantic perspective of ornaments so we can step away from the syntactic detail and handle ornaments more easily, in preparation for Section 6.2. Consider the equivalence on ornaments that was left undefined in **??**, whose type is

$$\begin{aligned} \mathit{OrnEq} \;:\; &\{I\ J \;:\; \mathsf{Set}\}\ \{e\ f \;:\; J \to I\}\ \{D \;:\; \mathsf{Desc}\ I\}\ \{E \;:\; \mathsf{Desc}\ J\} \to \\ &\mathsf{Orn}\ e\ D\ E \to \mathsf{Orn}\ f\ D\ E \to \mathsf{Set} \end{aligned}$$

Here we aim for an extensional equality — for example, if the only difference between two ornaments is either

- that one uses $\sigma\ S$ and the other uses $\Delta[\,s : S\,]\ \nabla[\,s\,]$, both expressing copying, or
- that one uses

  $$\Delta[\,s : S\,]\ \Delta[\,t : T\,]\ \nabla[\,f\ s\,]\ \nabla[\,g\ t\,]$$

  and the other uses

  $$\Delta[\,s : S\,]\ \nabla[\,f\ s\,]\ \Delta[\,t : T\,]\ \nabla[\,g\ t\,]$$

  the latter swapping the order of the two independent markings $\Delta[\,t : T\,]$ and $\nabla[\,f\ s\,]$,

then it seems pointless to distinguish the two ornaments. Since the direct semantics of ornaments is decoded componentwise by *erase*, we might define *OrnEq* as pointwise equality of *erase*, which requires a rather tricky type to express. However, it turns out that we can focus on only one aspect of *erase*: fixing two response descriptions related by at least one response ornament, the two response descriptions necessarily have the same pattern of recursive positions and *erase* always copies the values at the positions, so the only behaviour of *erase* that can vary with response ornaments is how the values of the fields are transformed. This suggests that, in an inhabitant of $[\![\ D\ ]\!]\ X$ where $D : \mathsf{RDesc}\ I$ and $X : I \to \mathsf{Set}$, we can separate the values of the fields from the values at the recursive positions, and focus on how *erase* acts on the first part.

Here **indexed containers** [Morris, 2007, Chapter 8] can provide a helpful perspective: An I-indexed container is a set S of **shapes**, a shape-indexed family of sets $P : S \to \mathsf{Set}$ of **positions**, and a function $\mathit{next} : (s : S) \to P\ s \to I$ associating each position with an index of type I. (The terminology slightly deviates from that of Morris, whose indexed containers refer to indexed families of the above indexed containers, directly comparable with the two-level structure of descriptions.) The container is interpreted as the type

$$\lambda\, X \mapsto \Sigma[\, s : S \,]\ ((p\ :\ P\ s) \to X\ (\mathit{next}\ s\ p))\ :\ (I \to \mathsf{Set}) \to \mathsf{Set}$$

That is, given a type family $X\ :\ I \to \mathsf{Set}$, an inhabitant of the container type starts with a specific shape, from which we derive a set of positions and associated indices, and contains an element of type $X\ i$ for each of the positions where i is the index associated with the position. (For example, take S to be the set of natural numbers, and let the set of positions derived from a natural number n be a finite set of size n, enumerable in a fixed order. The container type is then isomorphic to the type of lists, whose elements are indexed in accordance with the *next* function.) Response descriptions can be regarded as a special case of indexed containers: values of fields constitute a shape, and sets of positions are restricted to finite sets; consequently, $\mathit{next}\ s\ :\ P\ s \to I$ for any $s\ :\ S$ can be represented by a List I, and there is no longer need to specify P. We thus define the set of shapes derived from a response description by

$$\begin{aligned}
&\mathsf{S}\ :\ \{I\ :\ \mathsf{Set}\} \to \mathsf{RDesc}\ I \to \mathsf{Set} \\
&\mathsf{S}\ (\mathsf{v}\ \mathit{is}) \quad\ = \top \\
&\mathsf{S}\ (\sigma\ S\ D)\ =\ \Sigma[\, s : S \,]\ \mathsf{S}\ (D\ s)
\end{aligned}$$

and the function *next* by

$$\begin{aligned}
&\mathit{next}\ :\ \{I\ :\ \mathsf{Set}\}\ (D\ :\ \mathsf{RDesc}\ I) \to \mathsf{S}\ D \to \mathsf{List}\ I \\
&\mathit{next}\ (\mathsf{v}\ \mathit{is}) \quad \blacksquare \qquad\ = \mathit{is} \\
&\mathit{next}\ (\sigma\ S\ D)\ (s\ ,\ ss)\ =\ \mathit{next}\ (D\ s)\ ss
\end{aligned}$$

We can then express that a piece of horizontal data of type $[\![\ D\ ]\!]\ X$ can be separated into a shape and a series of contained elements via the following isomorphism, whose implementation is straightforward:

$$\begin{aligned}
&\mathit{horizontal\text{-}iso}\ :\ \{I\ :\ \mathsf{Set}\}\ (D\ :\ \mathsf{RDesc}\ I)\ (X\ :\ I \to \mathsf{Set}) \to \\
&\qquad\qquad [\![\ D\ ]\!]\ X \cong \Sigma\ (\mathsf{S}\ D)\ (\mathit{flip}\ \mathbb{P}\ X \circ \mathit{next}\ D)
\end{aligned}$$

Back to the semantic equivalence of ornaments. Define a specialised version of *erase* on shapes:

$$\begin{aligned}
&\mathit{erase}_{\mathsf{S}}\ :\ \{I\ J\ :\ \mathsf{Set}\}\ \{e\ :\ J \to I\}\ \{D\ :\ \mathsf{RDesc}\ I\}\ \{E\ :\ \mathsf{RDesc}\ J\} \\
&\qquad\qquad (O\ :\ \mathsf{ROrn}\ e\ D\ E) \to \mathsf{S}\ E \to \mathsf{S}\ D \\
&\mathit{erase}_{\mathsf{S}}\ (\mathsf{v}\ \mathit{eqs}) \quad \blacksquare \qquad\ = \blacksquare
\end{aligned}$$

$$
\begin{aligned}
&\mathit{erase}_{\mathsf{S}}\ (\sigma\ S\ O)\ (s\ ,\ ss) &&= s\ ,\ \mathit{erase}_{\mathsf{S}}\ (O\ s)\ ss\\
&\mathit{erase}_{\mathsf{S}}\ (\Delta\ T\ O)\ (t\ ,\ ss) &&= \mathit{erase}_{\mathsf{S}}\ (O\ t)\ ss\\
&\mathit{erase}_{\mathsf{S}}\ (\nabla\ s\ O)\ ss &&= s\ ,\ \mathit{erase}_{\mathsf{S}}\ O\ ss
\end{aligned}
$$

This is the aspect of *erase* that we are interested in, and we can now define equivalence of ornaments as

$$
\begin{aligned}
&\mathit{OrnEq}\ :\ \{I\ J\ :\ \mathsf{Set}\}\ \{e\ f\ :\ J \to I\}\ \{D\ :\ \mathsf{Desc}\ I\}\ \{E\ :\ \mathsf{Desc}\ J\} \to\\
&\qquad \mathsf{Orn}\ e\ D\ E \to \mathsf{Orn}\ f\ D\ E \to \mathsf{Set}\\
&\mathit{OrnEq}\ \{I\}\ \{J\}\ \{e\}\ \{f\}\ O\ P\ =\\
&\quad (e \doteq f) \times ((j\ :\ J) \to \mathit{erase}_{\mathsf{S}}\ (O\ (\mathsf{ok}\ j)) \cong \mathit{erase}_{\mathsf{S}}\ (P\ (\mathsf{ok}\ j)))
\end{aligned}
$$

which can be proved to imply pointwise equality of *forget* O and *forget* P by (datatype-generic) induction on their input: With the help of *horizontal-iso*, the general *erase* can be seen as first separating a piece of horizontal data into a shape and a series of recursive values, processing the shape by $\mathit{erase}_{\mathsf{S}}$, and combining the resulting shape with the recursive values. Since the real work is done by $\mathit{erase}_{\mathsf{S}}$, requiring extensional equality of $\mathit{erase}_{\mathsf{S}}$ is sufficient. This argument is then extended vertically by induction.

We have seen that response ornaments induce shape transformations (by $\mathit{erase}_{\mathsf{S}}$). Conversely, can we derive response ornaments from shape transformations? More specifically: Let D : $\mathsf{RDesc}\ I$, E : $\mathsf{RDesc}\ J$, and t : $\mathsf{S}\ E \to \mathsf{S}\ D$, and we wish to construct a response ornament from D to E. The shape transformation t expects a complete $\mathsf{S}\ E$ as its argument, which, in the response ornament, can be assembled by first marking all fields of E as additional by Δ. We then apply t to the assembled shape ss : $\mathsf{S}\ E$, resulting in a shape $t\ ss$: $\mathsf{S}\ D$, and fill out all fields of D using values in this shape by ∇. Finally, we have to use v to end the response ornament, which requires an index coherence proof of type $\mathbb{E}\ e\ (\mathit{next}\ E\ ss)\ (\mathit{next}\ D\ (t\ ss))$ for some e : $J \to I$ — this is the extra condition that we need to impose on the shape transformation for deriving a response ornament. We thus define **horizontal transformations** as

```
record HTrans {I J : Set} (e : J → I) (D : RDesc I) (E : RDesc J) : Set
  where
    constructor _,_
```

field
  t : S E → S D
  c : $(ss$: S $E)$ → 𝔼 e $(next\ E\ ss)$ $(next\ D\ (t\ ss))$

The response ornaments derived by the above process are called **normal response ornaments**, which are defined by

```
normROrn-∇ : {I J : Set} {e : J → I} {D : RDesc I} {js : List J} →
               (ss : S D) → 𝔼 e js (next D ss) → ROrn e D (v js)
normROrn-∇ {D := v is  } ▪       eqs = v eqs
normROrn-∇ {D := σ S D} (s , ss) eqs = ∇[ s ] normROrn-∇ {D := D s} ss eqs
normROrn : {I J : Set} {e : J → I} {D : RDesc I} {E : RDesc J} →
           HTrans e D E → ROrn e D E
normROrn {E := v js  } tr = normROrn-∇ (HTrans.t tr ▪) (HTrans.c tr ▪)
normROrn {E := σ S E} tr =
  Δ[ s : S ] normROrn {E := E s} (curry (HTrans.t tr) s , curry (HTrans.c tr) s)
```

The top-level function *normROrn* exhausts all fields of E by Δ, partially applying the transformation along the way, and then *normROrn*-∇ takes over and inserts values obtained from the result of the transformation into fields of D by ∇, ending with the placement of the index coherence proof.

We can now easily arrange a category 𝔽HTrans of descriptions and horizontal transformations, which is isomorphic to Orn. Its objects are of type Σ Set Desc as in Orn, and its sets of morphisms are

```
λ { (J , E) (I , D) ↦ Σ[ e : J → I ] FHTrans e D E }
```

where FHTrans is the type of **families of horizontal transformations**, defined in the usual way:

```
FHTrans : {I J : Set} → (J → I) → Desc I → Desc J → Set
FHTrans {I} {J} e D E = {i : I} (j : e ⁻¹ i) → HTrans e (D i) (E (und j))
```

Morphism equivalence is defined to be pointwise equality of the shape transformations extracted by HTrans.t, and identities and composition are defined in terms of functional identities and composition. We then have two functors Erase : Functor Orn 𝔽HTrans and Normal : Functor 𝔽HTrans Orn back

and forth between the two categories: the object parts of both functors are identities, and for the morphism parts, Erase maps ornaments to componentwise $erase_{\mathsf{S}}$ (with suitable coherence proofs) and Normal maps families of horizontal transformations to normal ornaments (i.e., componentwise normal response ornaments). For any $tr : \mathsf{HTrans}\ e\ D\ E$, we can prove that

$$erase_{\mathsf{S}}\ (normROrn\ tr) \doteq \mathsf{HTrans}.t\ tr$$

which guarantees that the morphism parts of Erase and Normal are inverse to each other.

Knowing that $\mathbb{O}$rn and $\mathbb{F}$HTrans are isomorphic means that, extensionally, we can regard ornaments and horizontal transformations as the same entities and freely switch between the two notions. For example, in **??**, where we did not have the notion of horizontal transformations yet, it was hard to establish the pullback property of parallel composition in Orn (**??**) by reasoning in terms of ornaments. By switching to $\mathbb{F}$HTrans, however, the proof becomes conceptually much tidier: Due to the isomorphism between $\mathbb{O}$rn and $\mathbb{F}$HTrans, it suffices to prove that the image of the square (**??**) under Erase in $\mathbb{F}$HTrans is a pullback. We have the following functor from $\mathbb{F}$HTrans to $\mathbb{F}$am which maps descriptions to indexed sets of shapes and discards index coherence proofs in horizontal transformations:

```
Shape : Functor FHTrans Fam
Shape = record
  { object   = λ { (I , D) ↦ I , λ i ↦ S (D i)                    }
  ; morphism = λ { (e , ts) ↦ e , λ {j} ↦ HTrans.t (ts (ok j)) }
  ; proofs of laws }
```

The functor Shape can be proved to be **pullback-reflecting**: if the image of a square in $\mathbb{F}$HTrans under Shape is a pullback in $\mathbb{F}$am, then the square itself is a pullback in $\mathbb{F}$HTrans. (The proof proceeds by manipulating shape transformations in $\mathbb{F}$am and then constructing the missing index coherence proof.) The problem is thus reduced to proving that the square (**??**) mapped into $\mathbb{F}$am by Erase and then Shape is a pullback, which boils down to the isomorphism

$$\begin{aligned}&\mathit{pcROD\text{-}iso}\ O\ P\ :\ \mathsf{S}\ (\mathit{toRDesc}\ (\mathit{pcROD}\ O\ P))\\ &\qquad\qquad\cong\ \Sigma[\,p : \mathsf{S}\ E \times \mathsf{S}\ F\,]\ \mathit{erase}_{\mathsf{S}}\ O\ (\mathit{outl}\ p)\ \equiv\ \mathit{erase}_{\mathsf{S}}\ P\ (\mathit{outr}\ p)\end{aligned}$$

and the two equations

$$\begin{aligned}&\mathit{erase}_{\mathsf{S}}\ (\mathit{diffROrn\text{-}l}\ O\ P)\ \doteq\ \mathit{outl}\circ\mathit{outl}\circ \mathsf{Iso}.\mathit{to}\ (\mathit{pcROD\text{-}iso}\ O\ P)\\ &\mathit{erase}_{\mathsf{S}}\ (\mathit{diffROrn\text{-}r}\ O\ P)\ \doteq\ \mathit{outr}\circ\mathit{outl}\circ \mathsf{Iso}.\mathit{to}\ (\mathit{pcROD\text{-}iso}\ O\ P)\end{aligned}$$

for any O : $\mathsf{ROrn}\ e\ D\ E$ and P : $\mathsf{ROrn}\ f\ D\ F$, all provable by induction on O and P. (The isomorphism and equations allow us to prove that the derived square in $\mathbb{F}\textsc{am}$ is isomorphic to the componentwise set-theoretic pullback, and thus the square itself is also a pullback.)

6.2 Ornaments and relational algebras

$$\begin{aligned}&\mathit{algROD\text{-}iso}\ :\ \{I\ :\ \mathsf{Set}\}\ (D\ :\ \mathsf{RDesc}\ I)\ (X\ :\ I\to\mathsf{Set})\ (P\ :\ \mathscr{P}\ (\llbracket\ D\ \rrbracket\ X))\to\\ &\qquad\mathsf{S}\ (\mathit{toRDesc}\ (\mathit{algROD}\ D\ P))\ \cong\ \Sigma\ (\llbracket\ D\ \rrbracket\ X)\ P\\ &\mathit{algROD\text{-}decomp}\ :\ \{I\ :\ \mathsf{Set}\}\ (D\ :\ \mathsf{RDesc}\ I)\ (X\ :\ I\to\mathsf{Set})\ (P\ :\ \mathscr{P}\ (\llbracket\ D\ \rrbracket\ X))\to \mathsf{S}\ (\mathit{toRDesc}\ (\mathit{algR}\\ &\mathit{algROD\text{-}decomp}\ (\mathsf{v}\ \mathit{is})\quad X\ P\ (\mathit{xs}\ ,\ p\ ,\ \blacksquare)\ =\ \mathit{xs}\ ,\ p\\ &\mathit{algROD\text{-}decomp}\ (\sigma\ S\ D)\ X\ P\ (s\ ,\ \mathit{hs})\quad =\ (\_,\_\ s * \mathit{id})\ (\mathit{algROD\text{-}decomp}\ (D\ s)\ X\ (\mathit{curry}\ P\ s)\ \mathit{hs})\\ &\mathit{algROD\text{-}comp}\ :\ \{I\ :\ \mathsf{Set}\}\ (D\ :\ \mathsf{RDesc}\ I)\ (X\ :\ I\to\mathsf{Set})\ (P\ :\ \mathscr{P}\ (\llbracket\ D\ \rrbracket\ X))\to(\mathit{xs}\ :\ \llbracket\ D\ \rrbracket\ X)\to\\ &\mathit{algROD\text{-}comp}\ (\mathsf{v}\ \mathit{is})\quad X\ P\ \mathit{xs}\ p\quad =\ \mathit{xs}\ ,\ p\ ,\ \blacksquare\\ &\mathit{algROD\text{-}comp}\ (\sigma\ S\ D)\ X\ P\ (s\ ,\ \mathit{xs})\ p\ =\ s\ ,\ \mathit{algROD\text{-}comp}\ (D\ s)\ X\ (\mathit{curry}\ P\ s)\ \mathit{xs}\ p\end{aligned}$$

6.3 Consequences

6.3.1 Ornamental algebraic ornamentation

$$\begin{aligned}&\mathsf{Iso}\ (\mathbb{R}\textsc{Alg}\ E)\ ((\mu\ D\circ e)\ ,\mathit{fun}\ (\mathit{ornAlg}\ O))\\ &\qquad\qquad((\lambda\ j\ \mapsto\ \Sigma[\,p : e\bowtie \mathit{outl}\,]\ \mathit{outl}_{\bowtie}\ p\ \equiv\ j)\ ,\mathit{clsAlg}\ (\mathit{diffOrn\text{-}l}\ O\ \lceil \mathit{singletonOD}\ D\rceil))\end{aligned}$$

$$\begin{aligned}&\mathsf{Iso}\ \mathbb{O}\textsc{rn}\ (\Sigma\ J\ (\mu\ D\circ e)\ ,\ \lfloor \mathit{algOD}\ D\ (\mathit{fun}\ (\mathit{ornAlg}\ O))\rfloor)\\ &\qquad((\Sigma[\,j : J\,]\ \Sigma[\,p : e\bowtie \mathit{outl}\,]\ \mathit{outl}_{\bowtie}\ p\ \equiv\ j)\ ,\ \lfloor \mathit{algOD}\ D\ (\mathit{clsAlg}\ (\mathit{diffOrn\text{-}l}\ O\ \lceil \mathit{singletonOD}\ D\end{aligned}$$

$$\mathsf{Iso}\ \mathbb{O}\textsc{rn}\ (\Sigma\ J\ (\mu\ D \circ e)\ ,\ \lfloor algOD\ D\ (fun\ (ornAlg\ O))\rfloor)$$
$$(e \bowtie outl\ ,\ \lfloor O \otimes \lceil singletonOD\ D\rceil\rfloor)$$

$$\mathsf{Iso}\ \mathbb{F}\textsc{am}\ (\Sigma\ J\ (\mu\ D \circ e)\ ,\ \mu\ \lfloor algOD\ D\ (fun\ (ornAlg\ O))\rfloor)$$
$$(e \bowtie outl\ ,\ \mu\ \lfloor OptPOD\ O\rfloor)$$

6.3.2 Parallel composition and the banana-split law

algebras corresponding to singleton ornaments and ornaments for optimised predicates

6.4 Discussion

bad computational behaviour; ornaments for optimised representation; compare the McBride [2011] version (compatible with the two-constructor universe) and the Dagand and McBride [2012] version of algebraic ornamentation in terms of "quality" (amount of σ's used); proof-relevant Algebra of Programming (e.g., *fun-preserves-fold*)

Perhaps the most important consequence of the completeness theorem (in its present form) is that it provides a new perspective on the expressive power of ornaments and inductive families. We showed in a previous paper Ko and Gibbons [2013] that every ornament induces a promotion predicate and a corresponding family of isomorphisms. But one question was untouched: can we determine (independently from ornaments) the range of predicates induced by ornaments? An answer to this question would tell us something about the expressive power of ornaments, and also about the expressive power of inductive families in general, since the inductive families we use are usually ornamentations of simpler algebraic datatypes from traditional functional programming. The completeness theorem offers such an answer: ornament-induced promotion predicates are exactly those expressible as relational folds (up to pointwise isomorphism). In other words, a predicate can be baked into a datatype by ornamentation if and only if it can be thought of as a nondeterministic classification of the inhabitants of the datatype with a relational fold. This is more a

guideline than a precise criterion, though, as the closest work about characterisation of the expressive power of folds discusses only functional folds Gibbons et al. [2001] (however, we believe that those results generalise to relations too). But this does encourage us to think about ornamentation computationally and to design new datatypes with relational algebraic methods.

Bibliography

Richard Bird and Oege de Moor [1997]. *Algebra of Programming*. Prentice-Hall. ↰ page 3

Pierre-Évariste Dagand and Conor McBride [2012]. Transporting functions across ornaments. In *International Conference on Functional Programming*, ICFP'12, pages 103–114. ACM. doi: 10.1145/2364527.2364544. ↰ pages 11 and 14

Jeremy Gibbons, Graham Hutton, and Thorsten Altenkirch [2001]. When is a function a fold or an unfold? *Electronic Notes in Theoretical Computer Science*, 44(1):146–160. doi: 10.1016/S1571-0661(04)80906-X. ↰ page 12

Hsiang-Shang Ko and Jeremy Gibbons [2013]. Modularising inductive families. *Progress in Informatics*, 10:65–88. doi: 10.2201/NiiPi.2013.10.5. ↰ page 11

Conor McBride [2011]. Ornamental algebras, algebraic ornaments. To appear in *Journal of Functional Programming*. ↰ pages 3, 11, and 14

Peter Morris [2007]. *Constructing Universes for Generic Programming*. Ph.D. thesis, University of Nottingham. ↰ page 5

Todo list