

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 ornamentation 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 $O : \text{Orn } e D E$ be an ornament from $D : \text{Desc } I$ to $E : \text{Desc } J$. Then there is a **classifying algebra** for O

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

whose the carrier type is defined by

$$\text{Inv}' e : I \rightarrow \text{Set}$$

$$\text{Inv}' e i = \Sigma[j : J] e j \equiv i$$

such that there are isomorphisms

$$(j : J) \rightarrow \mu \lfloor \text{algOD } D \text{ (clsAlg } O) \rfloor (e j, j, \text{refl}) \cong \mu E j$$

That is, the algebraic ornamentation of D using the classifying algebra derived from O produces a datatype isomorphic to μE , 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 $\lceil \text{OrdListOD} \rceil$ from lists to ordered lists:

$$\begin{aligned} R : \mathbb{F} (\text{ListD } Val) (\text{Inv}' !) &\rightsquigarrow \text{Inv}' ! \quad \text{-- } ! : Val \rightarrow \top \\ R ('nil, \quad \quad \quad \blacksquare) (b, \text{refl}) &= \top \\ R ('cons, x, (b', \text{refl}), \blacksquare) (b, \text{refl}) &= (b \leq x) \times (b' \equiv x) \end{aligned}$$

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

$$\begin{aligned} \text{AlgList } Val \text{ } R (b, \text{refl}) &\cong \Sigma [xs : \text{List } Val] (\lfloor R \rfloor xs) (b, \text{refl}) \\ &\cong \Sigma [xs : \text{List } Val] \text{Ordered } b \text{ } xs \quad \cong \text{OrdList } b \end{aligned}$$

For another example, consider the ornament $\text{NatD-ListD } A$, for which we can use the following classifying algebra S :

$$\begin{aligned} S : \mathbb{F} \text{NatD} (\text{Inv}' !) &\rightsquigarrow \text{Inv}' ! \quad \text{-- where } ! : \top \rightarrow \top \\ S ('nil, \quad \quad \quad \blacksquare) (\blacksquare, \text{refl}) &= \top \\ S ('cons, (\blacksquare, \text{refl}), \blacksquare) (\blacksquare, \text{refl}) &= A \end{aligned}$$

The result of folding a natural number n with S is uninteresting, as it can only be $'\blacksquare'$. 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 $\text{Vec } A \text{ } n$. Thus

$$\begin{aligned}
\mu \llbracket \text{algOD NatD } S \rrbracket (\blacksquare, \blacksquare, \text{refl}) &\cong \Sigma[n : \text{Nat}] (\llbracket S \rrbracket n (\blacksquare, \text{refl})) \\
&\cong \Sigma[n : \text{Nat}] \text{Vec } A \ n \qquad \cong \text{List } A
\end{aligned}$$

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 (in the sense that all one cares about proofs of set membership is their existence). \square

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 $\text{List } A$ is refined by the collection of types $\text{Vec } A \ n$ for all $n : \text{Nat}$. A list, say $a :: b :: [] : \text{List } A$, can be reconstructed as a vector by starting in the type $\text{Vec } A \ \text{zero}$ as $[]$, jumping to the next type $\text{Vec } A \ (\text{suc zero})$ as $b :: []$, and finally landing in $\text{Vec } A \ (\text{suc} (\text{suc zero}))$ 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 $D : \text{Desc } I$ be a description and $R : \mathbb{F} D X \rightsquigarrow X$

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

$$\text{fun } h \cdot R \simeq \text{clsAlg } [\text{algOD } D\ R] \cdot \mathbb{R}\ D\ (\text{fun } h)$$

or diagrammatically:

$$\begin{array}{ccc} X & \xleftarrow{\cong} \xrightarrow{\text{fun } h} & \text{Inv}'\ \text{outl} \\ \uparrow R & & \uparrow \text{clsAlg } [\text{algOD } D\ R] \\ \mathbb{F}\ D\ X & \xleftarrow{\cong} \xrightarrow{\mathbb{R}\ D\ (\text{fun } h)} & \mathbb{F}\ D\ (\text{Inv}'\ \text{outl}) \end{array}$$

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 $\text{SliceCategory ORN } (I, D)$ 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; reduced formalisation

6.1 Ornaments and horizontal transformations

We first 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 ???:

$$\text{OrnEq} : \{I J : \text{Set}\} \{ef : J \rightarrow I\} \{D : \text{Desc } I\} \{E : \text{Desc } J\} \rightarrow \\ \text{Orn } e D E \rightarrow \text{Orn } f D E \rightarrow \text{Set}$$

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

- that one uses σS 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 middle two independent markings,

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 $\llbracket D \rrbracket X$ where $D : \text{RDesc } I$ and $X : I \rightarrow \text{Set}$, 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 $P : S \rightarrow \text{Set}$ of **positions**, and a function $\text{next} : (s : S) \rightarrow P s \rightarrow I$ 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

$$\lambda X \mapsto \Sigma[s : S] ((p : P s) \rightarrow X (\text{next } s p)) : (I \rightarrow \text{Set}) \rightarrow \text{Set}$$

That is, given a type family $X : I \rightarrow \text{Set}$, 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, $\text{next}\ s : P\ s \rightarrow I$ 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

$$\begin{aligned} S &: \{I : \text{Set}\} \rightarrow \text{RDesc}\ I \rightarrow \text{Set} \\ S\ (\vee\ is) &= \top \\ S\ (\sigma\ S\ D) &= \Sigma[s : S]\ S\ (D\ s) \end{aligned}$$

and the function *next* by

$$\begin{aligned} \text{next} &: \{I : \text{Set}\} (D : \text{RDesc}\ I) \rightarrow S\ D \rightarrow \text{List}\ I \\ \text{next}\ (\vee\ is) &\quad \blacksquare \quad =\ is \\ \text{next}\ (\sigma\ S\ D)\ (s, ss) &= \text{next}\ (D\ s)\ ss \end{aligned}$$

We can then express that a piece of horizontal data of type $\llbracket D \rrbracket X$ can be separated into a shape and a series of contained elements via the following isomorphism:

$$\begin{aligned} \text{horizontal-iso} &: \{I : \text{Set}\} (D : \text{RDesc}\ I) (X : I \rightarrow \text{Set}) \rightarrow \\ &\quad \llbracket D \rrbracket X \cong \Sigma (S\ D) (\text{flip}\ \mathbb{P}\ X \circ \text{next}\ D) \end{aligned}$$

The implementation is just lossless rearrangement of data — for example, the left-to-right direction is defined by

$$\begin{aligned} \text{hori-decomp} &: \{I : \text{Set}\} (D : \text{RDesc}\ I) (X : I \rightarrow \text{Set}) \rightarrow \\ &\quad \llbracket D \rrbracket X \rightarrow \Sigma (S\ D) (\text{flip}\ \mathbb{P}\ X \circ \text{next}\ D) \\ \text{hori-decomp}\ (\vee\ is)\ X\ xs &= \blacksquare, xs \\ \text{hori-decomp}\ (\sigma\ S\ D)\ X\ (s, hs) &= (-, -\ s * \text{id}) (\text{hori-decomp}\ (D\ s)\ X\ hs) \end{aligned}$$

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

$$\begin{aligned}
\text{erase}_S &: \{I J : \text{Set}\} \{e : J \rightarrow I\} \{D : \text{RDesc } I\} \{E : \text{RDesc } J\} \\
&\quad (O : \text{ROrn } e D E) \rightarrow S E \rightarrow S D \\
\text{erase}_S (\vee \text{ eqs}) &\quad \blacksquare = \blacksquare \\
\text{erase}_S (\sigma S O) (s, ss) &= s, \text{erase}_S (O s) ss \\
\text{erase}_S (\Delta T O) (t, ss) &= \text{erase}_S (O t) ss \\
\text{erase}_S (\nabla s O) ss &= s, \text{erase}_S O \quad ss
\end{aligned}$$

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

$$\begin{aligned}
\text{OrnEq} &: \{I J : \text{Set}\} \{ef : J \rightarrow I\} \{D : \text{Desc } I\} \{E : \text{Desc } J\} \rightarrow \\
&\quad \text{Orn } e D E \rightarrow \text{Orn } f D E \rightarrow \text{Set} \\
\text{OrnEq } \{I\} \{J\} \{e\} \{f\} O P &= \\
&\quad (e \doteq f) \times ((j : J) \rightarrow \text{erase}_S (O (\text{ok } j))) \cong \text{erase}_S (P (\text{ok } j))
\end{aligned}$$

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 erase_S , and combining the resulting shape with the recursive values. Since the real work is done by erase_S , requiring extensional equality of erase_S is sufficient. This argument is then extended vertically by induction.

We have seen that response ornaments induce shape transformations (by erase_S). Conversely, can we derive response ornaments from shape transformations? More specifically: Let $D : \text{RDesc } I$, $E : \text{RDesc } J$, and $t : S E \rightarrow S D$, and we wish to construct a response ornament from D to E . The shape transformation t expects a complete $S E$ 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 $ss : S E$, resulting in a shape $t ss : S D$, and fill out all fields of D using values in this shape by ∇ . Finally, we have to use \vee to end the response ornament, which requires an index coherence proof of type $\mathbb{E} e (\text{next } E ss) (\text{next } D (t ss))$ for some $e : J \rightarrow I$ — 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) →  $\mathbb{E}$  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- $\nabla$  : {I J : Set} {e : J → I} {D : RDesc I} {js : List J} →
  (ss : S D) →  $\mathbb{E}$  e js (next D ss) → ROrn e D (v js)
normROrn- $\nabla$  {D := v is } ■ eqs = v eqs
normROrn- $\nabla$  {D :=  $\sigma$  S D} (s , ss) eqs =  $\nabla$ [s] normROrn- $\nabla$  {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- $\nabla$  (HTrans.t tr ■) (HTrans.c tr ■)
normROrn {E :=  $\sigma$  S E} tr =
   $\Delta$ [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 $\mathbb{FHTRANS}$ of descriptions and horizontal transformations, which is isomorphic to \mathbb{ORN} . Its objects are of type $\Sigma \text{ Set Desc}$ as in \mathbb{ORN} , and its sets of morphisms are

$$\lambda \{ (J, E) (I, D) \mapsto \Sigma[e : J \rightarrow I] \text{ FHTrans } e D E \}$$

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 = (j : J) → HTrans e (D (e j)) (E 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 $\text{ERASE} : \text{Functor } \mathbf{ORN} \rightarrow \mathbf{FHTRANS}$ and $\text{NORMAL} : \text{Functor } \mathbf{FHTRANS} \rightarrow \mathbf{ORN}$ 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 erases_S (with suitable coherence proofs) and NORMAL maps families of horizontal transformations to normal ornaments (i.e., componentwise normal response ornaments). For any $tr : \text{HTrans } e \rightarrow D \rightarrow E$, we can prove that

$$\text{erases}_S (\text{normROrn } tr) \doteq \text{HTrans.t } tr$$

which guarantees that the morphism parts of ERASE and NORMAL are inverse to each other.

Knowing that \mathbf{ORN} and $\mathbf{FHTRANS}$ 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 \mathbf{ORN} (??) by reasoning in terms of ornaments. By switching to $\mathbf{FHTRANS}$, however, the proof becomes conceptually much simpler: Due to the isomorphism between \mathbf{ORN} and $\mathbf{FHTRANS}$, it suffices to prove that the image of the square (??) under ERASE in $\mathbf{FHTRANS}$ is a pullback. We have the following functor SHAPE from $\mathbf{FHTRANS}$ to \mathbf{FAM} , which maps descriptions to indexed sets of shapes and discards index coherence proofs in horizontal transformations:

$\text{SHAPE} : \text{Functor } \mathbf{FHTRANS} \rightarrow \mathbf{FAM}$

$\text{SHAPE} = \text{record}$

$\{ \text{object} = \lambda \{ (I, D) \mapsto I, \lambda i \mapsto S(D\ i) \}$
 $;\text{ morphism} = \lambda \{ (e, ts) \mapsto e, \lambda \{j\} \mapsto \text{HTrans.t } (ts\ j) \}$
 $;\text{ proofs of laws} \}$

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

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

with the two equations

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

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

6.2 Ornaments and relational algebras

In this section we seek to characterise ornaments and relational algebras as essentially the same entities via an **equivalence of categories**, which subsumes the completeness theorem (and its dual) in the beginning of this chapter. The definition of category equivalence is unfolded below:

- An equivalence between two categories consists of two functors back and forth such that their compositions are **naturally isomorphic** to the identity functors.
 - A natural isomorphism between two functors $F, G : \text{Functor } C\ D$ is an isomorphism in the **functor category** from C to D , whose objects are functors from C to D and morphisms are **natural transformations**.
 - A natural transformation from F to G is a way of relating the image of C under F to the image of C under G : it consists of a morphism $\phi\ X : object\ F\ X \Rightarrow_D object\ G\ X$ for each object X in C , which we call the components of the natural transformation, and for any morphism

$m : X \Rightarrow_C Y$ the following **naturality** diagram commutes:

$$\begin{array}{ccc}
 \text{object } F X & \xrightarrow{\phi X} & \text{object } G X \\
 \text{morphism } F m \downarrow & & \downarrow \text{morphism } G m \\
 \text{object } F Y & \xrightarrow{\phi Y} & \text{object } G Y
 \end{array}$$

Both equivalence and composition for natural transformations are componentwise.

- Natural isomorphism is an equivalence relation on functors, so we can assemble a category whose objects are (smaller) categories and morphisms are functors, with natural isomorphism as the equivalence on morphisms. An equivalence of categories is then an isomorphism in this category of categories and functors.

Fixing $D : \text{Desc } I$, the completeness theorem is a consequence of one direction of the object part of an equivalence between $\text{SliceCategory } \mathbf{ORN} (I, D)$ — which we denote by $\mathbf{ORN} / (I, D)$ from now on — and the category $\mathbf{RALG } D$ whose objects are relational D -algebras

```

record RAlgebra  $D$  : Set1 where
  constructor  $-, -$ 
  field
    Carrier :  $I \rightarrow \text{Set}$ 
    Alg      :  $\mathbb{F} D \text{ Carrier} \rightsquigarrow \text{Carrier}$ 

```

and morphisms are algebra homomorphisms (which we will elaborate on later): The two functors forming the equivalence are

$\text{CLSALG} : \text{Functor } (\mathbf{ORN} / (I, D)) (\mathbf{RALG } D)$

and

$\text{ALGORN} : \text{Functor } (\mathbf{RALG } D) (\mathbf{ORN} / (I, D))$

and their object parts are respectively

$$\begin{array}{ccc}
 J, E & \text{Inv}' e & X \\
 e, O \downarrow & \mapsto \uparrow \text{clsAlg } O & R \uparrow \\
 I, D & \mathbb{F} D (\text{Inv}' e) & \mathbb{F} D X
 \end{array}
 \quad \text{and} \quad
 \begin{array}{ccc}
 X & \Sigma I X, [\text{algOD } D R] & \\
 R \uparrow & \mapsto \downarrow \text{outl}, [\text{algOD } D R] & \\
 \mathbb{F} D X & & I, D
 \end{array}$$

The natural isomorphism between $\text{ALGORN} \diamond \text{CLSALG}$ and the identity functor on $\text{ORN} / (I, D)$ consists of two natural transformations component-wise inverse to each other; their components constitute, for each ornament $O : \text{Orn} \rightarrow D \rightarrow E$, the following isomorphism in $\text{ORN} / (I, D)$:

$$\begin{array}{ccc}
 \Sigma I (\text{Inv}' e), [\text{algOD } D (\text{clsAlg } O)] & \xleftarrow{\cong} & J, E \\
 & \searrow & \swarrow \\
 \text{outl}, [\text{algOD } D (\text{clsAlg } O)] & & e, O \\
 & \searrow & \swarrow \\
 & I, D &
 \end{array} \quad (6.1)$$

The completeness theorem is then obtained by mapping these isomorphisms into FAM by $\text{IND} \diamond \text{SLICEF}$.

Now we turn to the morphisms of $\text{RALG } D$. A first thought might be adopting relational homomorphisms, which we have seen in ?? — a relational homomorphism from an algebra (X, R) to another algebra (Y, S) is a relation $H : X \rightsquigarrow Y$ such that $H \cdot R \simeq S \cdot R \circ D H$, or diagrammatically:

$$\begin{array}{ccc}
 X & \xrightarrow{H} & Y \\
 R \uparrow & & \uparrow S \\
 \text{F } D X & \xrightarrow{\text{R } D H} & \text{F } D Y
 \end{array}$$

Morphism equivalence is the equivalence of the relations between carriers, ignoring the homomorphism condition. It turns out, however, that this is not the right notion of homomorphisms for establishing the categorical equivalence. The right notion will surface as we construct the morphism part of ALGORN below.

The morphism part of ALGORN maps a homomorphism from (X, R) to (Y, S) to a morphism in $\text{ORN} / (I, D)$:

$$\begin{array}{ccc}
 \Sigma I X, [\text{algOD } D R] & \xrightarrow{\{\} _0, \{\} _1} & \Sigma I Y, [\text{algOD } D S] \\
 & \searrow & \swarrow \\
 \text{outl}, [\text{algOD } D R] & & \text{outl}, [\text{algOD } D S] \\
 & \searrow & \swarrow \\
 & I, D &
 \end{array}$$

To make the diagram commute, Goal 0 should be of the form $\text{id} * h$ for some function $h : X \rightrightarrows Y$, but the closest thing we have is merely a relation between

X and Y — to have an ornament between descriptions, we need to specify computation on the index sets of the descriptions, which correspond to the carriers of relational algebras, and hence we should restrict $\mathbb{R}ALG D$ to include only functional homomorphisms:

$$\begin{array}{ccc} X & \xrightarrow{\text{fun } h} & Y \\ R \uparrow & & \uparrow S \\ \mathbb{F} D X & \xrightarrow{\mathbb{R} D (\text{fun } h)} & \mathbb{F} D Y \end{array}$$

The equivalence on them is refined to pointwise equality of the functions between carriers.

We continue with Goal 1, which requires an ornament from $\lfloor algOD D S \rfloor$ to $\lfloor algOD D R \rfloor$. Switching to the perspective of horizontal transformations, we should construct a function which essentially replaces a proof about R with a proof about S inside a shape derived from algebraic ornamentation: For such shapes we have the following isomorphisms

$$\begin{aligned} algROD\text{-}iso : \{I : \text{Set}\} (D' : \text{RDesc } I) \{X : I \rightarrow \text{Set}\} (P : \mathcal{P} (\llbracket D' \rrbracket X)) \rightarrow \\ S (toRDesc (algROD D' P)) \cong \Sigma (\llbracket D' \rrbracket X) P \end{aligned}$$

whose implementation is, again, just lossless rearrangement of data — for example, the left-to-right direction is defined by

$$\begin{aligned} algROD\text{-}decomp : \\ \{I : \text{Set}\} (D' : \text{RDesc } I) (X : I \rightarrow \text{Set}) (P : \mathcal{P} (\llbracket D' \rrbracket X)) \rightarrow \\ S (toRDesc (algROD D P)) \rightarrow \Sigma (\llbracket D' \rrbracket X) P \\ algROD\text{-}decomp (\vee is) \quad X P (xs, p, \blacksquare) = xs, p \\ algROD\text{-}decomp (\sigma S D') X P (s, ss) = \\ (_, _ \circ s * id) (algROD\text{-}decomp (D' s) X (curry P s) ss) \end{aligned}$$

By *algROD-iso* and *horizontal-iso*, the type of shapes for an algebraic response ornamentation can be decomposed into three parts:

- the type of shapes for the basic response description,
- the type of series of indices for the recursive positions, and
- the type of proofs about the basic shape and the indices.

(For example, in the case of $\text{AlgList } A \ R$, the three parts are

- the constructor field and the element field $a : A$ from $\text{ListD } A$,
- the field $x' : X$, and
- the $rnil$ and $rcons$ fields.)

Expanding the definition of $\text{algOD } (??)$, at Goal 1 we should construct for every $i : I$ and $x : X \ i$ a shape transformation

$$\begin{array}{c}
 S \ (toRDesc \ (algROD \ (D \ i) \ ((R^\circ) \ x))) \\
 \downarrow \text{Iso.to } (algROD\text{-iso } (D \ i) \ ((R^\circ) \ x)) \\
 \Sigma \ (\llbracket D \ i \rrbracket X) \ ((R^\circ) \ x) \\
 \downarrow \text{mapRD } (D \ i) \ h * \{ \ }_2 \\
 \Sigma \ (\llbracket D \ i \rrbracket Y) \ ((S^\circ) \ (h \ x)) \\
 \downarrow \text{Iso.from } (algROD\text{-iso } (D \ i) \ ((S^\circ) \ (h \ x))) \\
 S \ (toRDesc \ (algROD \ (D \ i) \ ((S^\circ) \ (h \ x))))
 \end{array} \tag{6.2}$$

where Goal 2 requires a function of type

$$\{xs : \llbracket D \ i \rrbracket X\} \rightarrow R \ xs \ x \rightarrow S \ (\text{mapRD } (D \ i) \ h \ xs) \ (h \ x)$$

which, when fully quantified, is a pointwise form of

$$\text{fun } h \cdot R \subseteq S \cdot \mathbb{R} D \ (\text{fun } h)$$

i.e., one direction of the homomorphism condition. This is actually the only direction of the homomorphism condition that we should impose on the functions between carriers, which is denoted by the inclusion sign in the “semi-commutative” diagram:

$$\begin{array}{ccc}
 X & \xrightarrow{\text{fun } h} & Y \\
 R \uparrow & \subseteq & \uparrow S \\
 \mathbb{F} D \ X & \xrightarrow{\mathbb{R} D \ (\text{fun } h)} & \mathbb{F} D \ Y
 \end{array}$$

The reason for requiring only one direction will become clear when we consider the morphism part of CLSALG .

Having constructed the morphism part of ALGORN , one of the functor laws we should prove is that it preserves equivalence, and it is when doing so that we encounter the final problem: we need to prove that the shape transformation (6.2) — whose behaviour depends on the computational content of the homomorphism condition — is extensionally the same for equivalent homomorphisms, but homomorphism equivalence as we defined it does not include any information about the homomorphism condition, and is hence not strong enough. We thus see that proofs of the homomorphism condition should not be treated irrelevantly since they are used computation-relevantly in the shape transformations, and we must take their behaviour into account when defining homomorphism equivalence. This brings us to the right notion of algebra homomorphisms, defined by

```
record RAlgMorphism D (X , R) (Y , S) : Set1 where
  constructor  $-, -$ 
  field
     $h : X \Rightarrow Y$ 
     $c : (i : I) (xs : \llbracket D\ i \rrbracket X) (x : X\ i) \rightarrow R\ xs\ x \rightarrow S\ (\text{mapRD}\ (D\ i)\ h\ xs)\ (h\ x)$ 
```

the equivalence on which is pointwise equality on both components h and c . At the beginning of this chapter, we saw in the example about the classifying algebra for the ornament from natural numbers to lists that the relational language is used proof-relevantly, deviating from traditional relational theories; that deviation is fully manifested in this proof-relevant category of relational algebras and homomorphisms.

Before we move on to the natural isomorphisms, it still remains to define CLSALG , in particular the clsAlg function. Considering the completeness theorem, the classifying algebra $\text{clsAlg}\ O$ for an ornament $O : \text{Orn}\ e\ D\ E$ where $E : \text{Desc}\ J$ should encode enough information for promoting a given inhabitant of $S\ (D\ (e\ j))$ to one of $S\ (E\ j)$ for any $j : J$, so the algebraic ornamentation of D using $\text{clsAlg}\ O$ can yield a datatype isomorphic to $\mu\ E$. Here for simplicity we require a complete inhabitant of $S\ (E\ j)$ that erases to the given inhabitant of $S\ (D\ (e\ j))$ by $\text{erase}_S\ O\ (\text{ok}\ j)$:

```
 $\text{clsAlg}\ O : \mathbb{F}\ D\ (\text{Inv}'\ e) \rightsquigarrow \text{Inv}'\ e$ 
 $\text{clsAlg}\ O\ js\ (j , \text{refl}) =$ 
```

$$\begin{aligned} \Sigma[ss : S(Ej)] \text{ erases } (O(okj)) ss \equiv \\ \text{outl}(\text{Iso.to}(\text{horizontal-iso}(D(ej))(\text{Inv}'e))js) \\ \times \{ \}_{3} \end{aligned}$$

Goal 3 is left for talking about the series of recursively computed indices contained in js , which should match with those specified for the recursive positions by Ej . (An example is the proof obligation $b' \equiv x$ in the classifying algebra for the ornament from lists to ordered lists given at the beginning of this chapter.) The type of the series of recursively computed indices

$$\text{outr}(\text{Iso.to}(\text{horizontal-iso}(D(ej))(\text{Inv}'e))js)$$

gets nasty, however; it suffices to know that Goal 3 requires that the above series of indices, when coerced to a List J , is equal to $\text{next}(Ej)ss$. The morphism part of CLSALG can then be constructed:

$$\begin{array}{ccc} J, E \xrightarrow{g, Q} K, F & & \text{Inv}'e \xrightarrow{\text{fun } \{ \}_{4}} \text{Inv}'f \\ e, O \searrow \quad \swarrow f, P & \mapsto & \begin{array}{ccc} \text{clsAlg } O \uparrow & \{ \subseteq \}_{5} & \uparrow \text{clsAlg } P \\ \mathbb{F} D(\text{Inv}'e) & \xrightarrow{\mathbb{R} D(\text{fun } -)} & \mathbb{F} D(\text{Inv}'f) \end{array} \\ I, D & & \end{array}$$

Goal 4 has type

$$\{i : I\} \rightarrow (\Sigma[j : J] \text{ ej} \equiv i) \rightarrow \Sigma[k : K] f k \equiv i$$

and can be easily discharged by g and the proof $f \circ g \doteq e$ extracted from the commutative triangle. At Goal 5, we should transform a proof about $\text{clsAlg } O$ to one about $\text{clsAlg } P$; expanding the definition of clsAlg , we should transform an inhabitant of $S(Ej)$ to one of $S(F(gj))$ for every $j : J$, so $\text{erases}(Q(okj))$ does the job, modulo manipulation of the associated equations. (Here we see clearly that the homomorphism condition can only require the left-to-right direction, since it is the only direction that can be constructed from Q .)

Now we look at the natural isomorphism between $\text{ALGORN} \diamond \text{CLSALG}$ and the identity functor on $\text{ORN} / (I, D)$ — the construction of the other natural isomorphism between $\text{CLSALG} \diamond \text{ALGORN}$ and the identity functor on $\mathbb{R}\text{ALG } D$ is based on a similar analysis and omitted from the presentation. The components (indexed by ornaments) of the natural isomorphism were shown in (6.1). Switching to horizontal transformations, we should construct an isomorphism

between

$$S \text{ (} toRDesc \text{ (} algROD \text{ (} D \text{ (} e \text{ } j \text{)) (} (clsAlg \text{ } O^\circ) \text{ (} j \text{ , refl) \text{))} \text{))} \quad \text{and} \quad S \text{ (} E \text{ } j \text{)}$$

for each $j : J$, and make sure that its two directions make the corresponding triangles commute. For the left-to-right direction, by *algROD-iso*, the left-hand side decomposes into some $js : \llbracket D \text{ (} e \text{ } j \text{) } \rrbracket (Inv' \text{ } e)$ and a proof of $clsAlg \text{ } O \text{ } js \text{ (} j \text{ , refl) }$ — the latter contains an inhabitant of $S \text{ (} E \text{ } j \text{)}$, which is exactly what we need. Conversely, any $ss : S \text{ (} E \text{ } j \text{)}$ by itself determines a $js : \llbracket D \text{ (} e \text{ } j \text{) } \rrbracket (Inv' \text{ } e)$ and a proof of $clsAlg \text{ } O \text{ } js \text{ (} j \text{ , refl) }$ that contains ss , since the equations in *clsAlg* completely specifies what js should be. The two directions are inverse to each other because all that is involved is lossless rearrangement of data, and the corresponding triangles commute because (in particular) the fields from D are not modified. We also need to establish naturality of one of the natural transformations (naturality of the other natural transformation can be derived from the componentwise isomorphisms) — here we choose the left-to-right direction. Let m be a morphism in $\mathbf{ORN} / (I, D)$:

$$\begin{array}{ccc} J, E & \xrightarrow{g, Q} & K, F \\ e, O & \searrow & \swarrow f, P \\ & I, D & \end{array}$$

Because the equivalence on slice morphisms ignores the commutative triangles, we only need to establish commutativity of the following square in \mathbf{ORN} :

$$\begin{array}{ccc} \Sigma I \text{ (} Inv' \text{ } e \text{) , } \llbracket algOD \text{ } D \text{ (} clsAlg \text{ } O \text{) } \rrbracket & \xrightarrow{\phi \text{ } O} & J, E \\ \text{morphism (SLICEF} \diamond \text{ ALGORN} \diamond \text{ CLSALG) } m \downarrow & & \downarrow g, Q \\ \Sigma I \text{ (} Inv' \text{ } f \text{) , } \llbracket algOD \text{ } D \text{ (} clsAlg \text{ } P \text{) } \rrbracket & \xrightarrow{\phi \text{ } P} & K, F \end{array}$$

where ϕ denotes the components of the natural transformation sketched above. Switching to $\mathbf{FHTRANS}$, the vertical morphisms essentially replace a shape of type $S \text{ (} E \text{ } j \text{)}$ (for some suitable $j : J$) with one of type $S \text{ (} F \text{ (} g \text{ } j \text{))}$, while the horizontal ones extract those shapes. The two operations are independent, and hence the diagram commutes.

6.3 Consequences

We began this chapter with the completeness theorem, and, with some effort, have extended it to a categorical equivalence. The extra effort is not wasted: the categorical equivalence allows us to perform reasoning across the realms of ornaments and relational algebras easily. Below we give two examples that connect ornamental and relational algebraic constructions via the categorical equivalence.

6.3.1 Ornamental algebraic ornamentation

The idea of computing promotion predicates and conversion isomorphisms from ornaments was first fully formulated by Ko and Gibbons [2011], where the promotion predicates were computed by **ornamental algebraic ornamentation**, i.e., algebraic ornamentation with an ornamental algebra. This thesis, on the other hand, uses parallel composition with singleton ornamentation to compute (optimised) ornamental promotion predicates. Using a part of the categorical equivalence, we can show that the two approaches indeed yield isomorphic predicates.

Let $O : \text{Orn } e \ D \ E$ where $D : \text{Desc } I$ and $E : \text{Desc } J$. Our goal is to show that the datatype obtained by algebraic ornamentation of E with the ornamental algebra

$$\text{fun } (\text{ornAlg } O) : \mathbb{F} E (\mu D \circ e) \rightsquigarrow (\mu D \circ e)$$

is isomorphic to the optimised predicate datatype for O , i.e.,

$$\text{Iso } \mathbb{FAM} (\Sigma J (\mu D \circ e), \mu \lfloor \text{algOD } E (\text{fun } (\text{ornAlg } O)) \rfloor) \\ (e \bowtie \text{outl} \quad , \mu \lfloor O \otimes \text{singletonOD } D \rfloor)$$

By isomorphism preservation of IND , it suffices to establish

$$\text{Iso } \mathbb{ORN} (\Sigma J (\mu D \circ e), \lfloor \text{algOD } E (\text{fun } (\text{ornAlg } O)) \rfloor) \\ (e \bowtie \text{outl} \quad , \lfloor O \otimes \lceil \text{singletonOD } D \rceil \rfloor)$$

We are attempting to establish that an algebraic ornamentation of E is isomorphic to some other description; if that description is also an algebraic ornament-

tation of E , then we can just reason in terms of their algebras. This is easy: from any ornament $P : \text{Orn} f E \lfloor O \otimes [\text{singletonOD } D] \rfloor$ we get an isomorphism

$$\text{Iso ORN} (e \bowtie \text{outl} \quad , \lfloor O \otimes [\text{singletonOD } D] \rfloor) \\ (\Sigma J (\text{Inv}' f) , \lfloor \text{algOD } E (\text{clsAlg } P) \rfloor)$$

and there is an obvious choice of P — $\text{diffOrn-l } O [\text{singletonOD } D]$. The proof obligation thus reduces to

$$\text{Iso ORN} (\Sigma J (\mu D \circ e \quad , \lfloor \text{algOD } E (\text{fun } (\text{ornAlg } O)) \rfloor) \\ (\Sigma J (\text{Inv}' \text{outl}_{\bowtie}) , \lfloor \text{algOD } E (\text{clsAlg } (\text{diffOrn-l } O [\text{singletonOD } D])) \rfloor))$$

which, by isomorphism preservation of $\text{SLICEF} \diamond \text{ALGORN}$, is further reduced to

$$\text{Iso } (\mathbb{R}\text{ALG } E) (\mu D \circ e \quad , \text{fun } (\text{ornAlg } O)) \\ (\text{Inv}' \text{outl}_{\bowtie} , \text{clsAlg } (\text{diffOrn-l } O [\text{singletonOD } D]))$$

6.3.2 Parallel composition and the banana-split law

algebras corresponding to singleton ornaments and ornaments for optimised predicates

6.4 Discussion

implicit change from Inv to Inv' ; (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*)

Starting from ??, we have been carefully evading the formalisation of isomorphism of categories.

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 19 and 22

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 20

Hsiang-Shang Ko and Jeremy GIBBONS [2011]. Modularising inductive families. In *Workshop on Generic Programming, WGP’11*, pages 13–24. ACM. doi: 10.1145/2036918.2036921. ↱ page 18

Hsiang-Shang Ko and Jeremy GIBBONS [2013]. Modularising inductive families. *Progress in Informatics*, 10:65–88. doi: 10.2201/NiiPi.2013.10.5. ↱ page 19

Conor MCBRIDE [2011]. Ornamental algebras, algebraic ornaments. To appear in *Journal of Functional Programming*. ↱ pages 3, 19, and 22

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

Todo list

see if the above can be revised to be slightly more specific; reduced formalisation	4
algebras corresponding to singleton ornaments and ornaments for optimised predicates	19
implicit change from Inv to Inv' ; (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., <i>fun-preserves-fold</i>)	19