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: by setting the carrier of the algebra R, we can choose whatever index set we want, 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 to relate the targeted index $x$, the head element $a$, and the index $x'$ of the recursive position 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

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

whose carrier type is defined by

$$Inv'\ e\ :\ I \to \mathsf{Set}$$
$$Inv'\ e\ i\ =\ \Sigma[j : J]\ e\ j \equiv i$$

such that there are isomorphisms

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

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 $\lceil OrdListOD \rceil$ from lists to ordered lists:

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

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 $\mathsf{Ordered}\ b\ xs$, and hence

$$\begin{aligned}
\mathsf{AlgList}\ Val\ R\ (b\ ,\ \mathsf{refl}) &\cong \Sigma[\, xs : \mathsf{List}\ Val \,]\ (\llbracket R \rrbracket)\ xs\ (b\ ,\ \mathsf{refl}) \\
&\cong \Sigma[\, xs : \mathsf{List}\ Val \,]\ \mathsf{Ordered}\ b\ xs \qquad \cong \mathsf{OrdList}\ b
\end{aligned}$$

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

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

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 $\mathsf{Vec}\ A\ n$. Thus

$$\begin{aligned}\mu\,\lfloor algOD\ NatD\ S\rfloor\ (\blacksquare\ ,\ \blacksquare\ ,\ \mathsf{refl}) &\cong \Sigma[\,n : \mathsf{Nat}\,]\ (\!|\,S\,|\!)\ n\ (\blacksquare\ ,\ \mathsf{refl}) \\ &\cong \Sigma[\,n : \mathsf{Nat}\,]\ \mathsf{Vec}\ A\ n \qquad\qquad \cong \mathsf{List}\ A\end{aligned}$$

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). □

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

$X\ i$ and $Inv'\ outl\ i$ for every $i\ :\ I$ (where $outl\ :\ \Sigma\ I\ X \to I$); call the forward direction of this family of isomorphisms $h\ :\ X \rightrightarrows Inv'\ outl$. Then we have

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

or diagrammatically:

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

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 a **categorical equivalence** between $SliceCategory\ \mathbb{O}\textsc{rn}\ (I\ ,\ D)$ 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 categorical 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. The results in this chapter are only partially formalised, largely due to the dreadful manoeuvrability of complicatedly typed terms. This issue, along with some others, is discussed in Section 6.4.

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 **??**:

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

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 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 $[\![\ D\ ]\!]\ X$ 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 $next\ :\ (s\ :\ S) \to P\ s \to I$ 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

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

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, $next\ s : P\ s \to I$ for any $s : S$ can be represented by a $\mathsf{List}\ I$, and there is no longer need to specify P. We thus define the set of shapes derived from a response description by

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

and the function *next* by

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

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:

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

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

$$
\begin{aligned}
&hori\text{-}decomp : \{I : \mathsf{Set}\}\ (D : \mathsf{RDesc}\ I)\ (X : I \to \mathsf{Set}) \to \\
&\qquad\qquad [\![\ D\ ]\!]\ X \to \Sigma\ (\mathsf{S}\ D)\ (flip\ \mathbb{P}\ X \circ next\ D) \\
&hori\text{-}decomp\ (\mathsf{v}\ is) \quad X\ xs \qquad = \blacksquare\ ,\ xs \\
&hori\text{-}decomp\ (\sigma\ S\ D)\ X\ (s\ ,\ xs) \ = (\_,\_\ s * id)\ (hori\text{-}decomp\ (D\ s)\ X\ xs)
\end{aligned}
$$

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

$$
\begin{aligned}
&\mathit{erase}_{\mathsf{S}} \;:\; \{I\,J \;:\; \mathsf{Set}\}\;\{e \;:\; J \to I\}\;\{D \;:\; \mathsf{RDesc}\;I\}\;\{E \;:\; \mathsf{RDesc}\;J\} \\
&\qquad\quad (O \;:\; \mathsf{ROrn}\;e\;D\;E) \to \mathsf{S}\;E \to \mathsf{S}\;D \\
&\mathit{erase}_{\mathsf{S}}\;(\mathsf{v}\;\mathit{eqs})\quad \blacksquare \qquad\quad = \blacksquare \\
&\mathit{erase}_{\mathsf{S}}\;(\sigma\;S\;O)\;(s\;,\;ss) \;=\; s\;,\;\mathit{erase}_{\mathsf{S}}\;(O\;s)\;ss \\
&\mathit{erase}_{\mathsf{S}}\;(\Delta\;T\;O)\;(t\;,\;ss) \;=\; \quad \mathit{erase}_{\mathsf{S}}\;(O\;t)\;ss \\
&\mathit{erase}_{\mathsf{S}}\;(\nabla\;s\;O)\;\;ss \qquad\quad = s\;,\;\mathit{erase}_{\mathsf{S}}\;O \qquad ss
\end{aligned}
$$

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

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

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  =  (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 Erase : Functor Orn IFHTrans and Normal : Functor IFHTrans 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 : 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 Orn and IFHTrans 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. Switching to IFHTrans, however, the proof becomes conceptually much simpler: Due to the isomorphism between Orn and IFHTrans, it suffices to prove that the image of the square (**??**) under Erase in IFHTrans is a pullback. We have the following functor Shape from IFHTrans to IFam, which maps descriptions to indexed sets of shapes and discards index coherence proofs in horizontal transformations:

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

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

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

with 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 : $\mathsf{ROrn}\ f\ D\ F$, 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

We now 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 categorical 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 : $\mathsf{Functor}\ C\ D$ 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 $\phi\ X$: $\mathit{object}\ F\ X \Rightarrow_D \mathit{object}\ G\ X$ 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:

$$\begin{array}{ccc} \mathit{object}\ F\ X & \xrightarrow{\phi\ X} & \mathit{object}\ G\ X \\ {\scriptstyle \mathit{morphism}\ F\ m}\downarrow & & \downarrow{\scriptstyle \mathit{morphism}\ G\ m} \\ \mathit{object}\ F\ Y & \xrightarrow[\phi\ Y]{} & \mathit{object}\ G\ Y \end{array}$$

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 $D : \mathsf{Desc}\ I$, the completeness theorem is a consequence of one direction of the object part of an equivalence between *SliceCategory* $\mathbb{O}\textsc{rn}\ (I\ ,\ D)$ — which we denote by $\mathbb{O}\textsc{rn}/(I\ ,\ D)$ from now on — and the category $\mathbb{R}\textsc{Alg}\ D$ whose objects are relational D-algebras

```
record RAlgebra D : Set₁ where
  constructor _,_
  field
    Carrier : I → Set
    Alg     : 𝔽 D Carrier ⇝ Carrier
```

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

```
ClsAlg : Functor (𝕆rn/(I , D)) (ℝAlg D)
```

and

```
AlgOrn : Functor (ℝAlg D) (𝕆rn/(I , D))
```

and their object parts are respectively

$$\begin{array}{ccc} J\ ,\ E & & \mathit{Inv'}\ e \\ {\scriptstyle e\ ,\ O}\downarrow & \mapsto & \uparrow{\scriptstyle \mathit{clsAlg}\ O} \\ I\ ,\ D & & \mathbb{F}\ D\ (\mathit{Inv'}\ e) \end{array} \qquad \text{and} \qquad \begin{array}{ccc} X & & \Sigma\ I\ X\ ,\ \lfloor \mathit{algOD}\ D\ R \rfloor \\ {\scriptstyle R}\uparrow & \mapsto & \downarrow{\scriptstyle \mathit{outl}\ ,\ \lceil \mathit{algOD}\ D\ R \rceil} \\ \mathbb{F}\ D\ X & & I\ ,\ D \end{array}$$

The natural isomorphism between AlgOrn ⋄ ClsAlg and the identity functor on $\mathbb{O}\text{rn}\,/\,(I\ ,\ D)$ consists of two natural transformations componentwise inverse to each other; their components constitute, for each ornament $O : \mathsf{Orn}\ e\ D\ E$, the following isomorphism in $\mathbb{O}\text{rn}\,/\,(I\ ,\ D)$:

$$\begin{array}{ccc} \Sigma\ I\ (Inv'\ e)\ ,\ \lfloor algOD\ D\ (clsAlg\ O)\rfloor & \longleftarrow\cong\longrightarrow & J\ ,\ E \\ & \searrow \quad \swarrow & \\ outl\ ,\ \lceil algOD\ D\ (clsAlg\ O)\rceil & & e\ ,\ O \\ & I\ ,\ D & \end{array} \tag{6.1}$$

The completeness theorem is then obtained by mapping these isomorphisms into $\mathbb{F}\text{am}$ by Ind ⋄ SliceF.

For the morphisms of $\mathbb{R}\text{Alg}\ D$, 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 $H \bullet R \simeq S \bullet \mathbb{R}\ D\ H$, or diagrammatically:

$$\begin{array}{ccc} X & \xrightarrow{\quad H \quad} & Y \\ {\scriptstyle R}\uparrow & & \uparrow{\scriptstyle S} \\ \mathbb{F}\ D\ X & \xrightarrow[\mathbb{R}\ D\ H]{} & \mathbb{F}\ D\ Y \end{array}$$

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 $\mathbb{O}\text{rn}\,/\,(I\ ,\ D)$:

$$\begin{array}{ccc} \Sigma\ I\ X\ ,\ \lfloor algOD\ D\ R\rfloor & \xrightarrow{\ \{\ \}_0\ ,\ \{\ \}_1\ } & \Sigma\ I\ Y\ ,\ \lfloor algOD\ D\ S\rfloor \\ & \searrow \quad \swarrow & \\ outl\ ,\ \lceil algOD\ D\ R\rceil & & outl\ ,\ \lceil algOD\ D\ S\rceil \\ & I\ ,\ D & \end{array}$$

To make the diagram commute, Goal 0 should be of the form $id * h$ 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 $\mathbb{R}\textsc{Alg}\ D$ to include only functional homomorphisms:

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

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 shape transformation which essentially replaces a proof about R with a proof about S within a shape derived from algebraic ornamentation: For such shapes we have the following isomorphisms

$$\begin{aligned} &algROD\text{-}iso\ :\ \{I\ :\ \mathsf{Set}\}\ (D'\ :\ \mathsf{RDesc}\ I)\ \{X\ :\ I \to \mathsf{Set}\}\ (P\ :\ \mathscr{P}\ (\llbracket\ D'\ \rrbracket\ X)) \to \\ &\qquad \mathsf{S}\ (toRDesc\ (algROD\ D'\ P)) \cong \Sigma\ (\llbracket\ D'\ \rrbracket\ X)\ P \end{aligned}$$

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

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

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 of the recursive positions, and
- the type of proofs about the basic shape and the indices.

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

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

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

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

where Goal 2 requires a function of type

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

which, when suitably quantified, is a pointwise form of

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

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:

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

(The reason for requiring only one direction will become clear when we consider the morphism part of ClsAlg.) In short, the morphism part of AlgOrn treats

the input homomorphism condition as a proof transformer, and apply that transformer to the proof encapsulated in the input shape.

One of the functor laws we should prove for AlgOrn is that its morphism part 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 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 ⇉ Y
    c : (i : I) (xs : ⟦ D i ⟧ X) (x : X i) → R xs x → S (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, we should define the function *clsAlg* used in its object part:

```
clsAlg O : 𝔽 D (Inv′ e) ⇝ Inv′ e
clsAlg O js (j , refl) =
  let (ss , ps) = Iso.to (horizontal-iso (D (e j)) (Inv′ e)) js
  in Σ[ ts ∈ S (E j) ] erase_S (O (ok j)) ts ≡ ss
                       ℙ-toList outl (next (D (e j)) ss) ps ≡ next (E j) ts
```

The condition for saying that $js : [\![D\ (e\ j)]\!]\ (Inv'\ e)$ is mapped by *clsAlg O* to $j : J$ is stated in terms of the two parts of *js* decomposed by *horizontal-iso*:

- For the shape part $ss : \mathsf{S}\ (D\ (e\ j))$, considering the completeness theorem, there should be enough information for promoting *ss* to an inhabitant of $\mathsf{S}\ (E\ j)$. Here for simplicity we require a complete inhabitant of $\mathsf{S}\ (E\ j)$ that erases to the given inhabitant of $\mathsf{S}\ (D\ (e\ j))$ by $erase_{\mathsf{S}}\ (O\ (\mathsf{ok}\ j))$. (In the example about the classifying algebra for the ornament from lists to ordered lists given at the beginning of this chapter, the representation of this part of the condition is optimised — only an inhabitant of $b \leqslant x$ is required.)
- As for *ps*, which is a series of recursively computed indices when *clsAlg O* is used in a fold, they should be equal to *next* $(E\ j)$ *ts* when coerced to a $\mathsf{List}\ J$, ensuring that the recursive classifications are successful. (This part of the condition corresponds to the proof obligation $b' \equiv x$ in the aforementioned example.)

The morphism part of ClsAlg can then be constructed:

$$
\begin{array}{ccc}
J\ ,\ E & \xrightarrow{g\ ,\ Q} & K\ ,\ F \\
e\ ,\ O & \searrow \quad \swarrow & f\ ,\ P \\
 & I\ ,\ D &
\end{array}
\qquad \mapsto \qquad
\begin{array}{ccc}
Inv'\ e & \xrightarrow{fun\ \{\ \}_3} & Inv'\ f \\
{\scriptstyle clsAlg\ O}\uparrow & \{\subseteq\}_4 & \uparrow{\scriptstyle clsAlg\ P} \\
\mathbb{F}\ D\ (Inv'\ e) & \xrightarrow[\mathbb{R}\ D\ (fun\ \_)]{} & \mathbb{F}\ D\ (Inv'\ f)
\end{array}
$$

Goal 3 has type

$$\{i : I\} \to (\Sigma[j : J]\ e\ j \equiv i) \to \Sigma[k : K]\ f\ k \equiv i$$

and can be easily discharged by g and the proof $f \circ g \doteq e$ extracted from the commutative triangle. At Goal 4, we should transform a proof about *clsAlg O* to one about *clsAlg P*; expanding the definition of *clsAlg*, we should transform an inhabitant of $\mathsf{S}\ (E\ j)$ to one of $\mathsf{S}\ (F\ (g\ j))$ for every $j : J$, so $erase_{\mathsf{S}}\ (Q\ (\mathsf{ok}\ j))$ 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 AlgOrn $\diamond$ ClsAlg and the identity functor on $\mathbb{O}\mathrm{rn}\ /\ (I\ ,\ D)$ — the construction of the other natural isomorphism between ClsAlg $\diamond$ AlgOrn and the identity functor on

$\mathbb{R}$RAlg 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

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

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 $js : [\![\ D\ (e\ j)\ ]\!]\ (\mathit{Inv}'\ e)$ and a proof of $\mathit{clsAlg}\ O\ js\ (j\ ,\ \mathsf{refl})$ — the latter contains an inhabitant of $\mathsf{S}\ (E\ j)$, which is exactly what we need. Conversely, any $ss : \mathsf{S}\ (E\ j)$ by itself uniquely determines a $js : [\![\ D\ (e\ j)\ ]\!]\ (\mathit{Inv}'\ e)$ and a proof of $\mathit{clsAlg}\ O\ js\ (j\ ,\ \mathsf{refl})$ 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 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 $\mathbb{O}$rn$/(I\ ,\ D)$:

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

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

$$\begin{array}{rcl} & \Sigma\ I\ (\mathit{Inv}'\ e)\ ,\ \lfloor \mathit{algOD}\ D\ (\mathit{clsAlg}\ O)\ \rfloor & \xrightarrow{\ \phi\ O\ } J\ ,\ E \\ \mathit{morphism}\ (\textsc{SliceF} \diamond \textsc{AlgOrn} \diamond \textsc{ClsAlg})\ m & \Big\downarrow & \Big\downarrow\ g\ ,\ Q \\ & \Sigma\ I\ (\mathit{Inv}'\ f)\ ,\ \lfloor \mathit{algOD}\ D\ (\mathit{clsAlg}\ P)\ \rfloor & \xrightarrow[\ \phi\ P\ ]{} K\ ,\ F \end{array}$$

where ϕ denotes the components of the natural transformation sketched above. Switching to horizontal transformations, the vertical morphisms essentially replace a shape of type $\mathsf{S}\ (E\ j)$ (for some suitable $j : J$) with one of type

$\mathsf{S}\ (F\ (g\ j))$, 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 boundary between ornaments and relational algebras easily. Below we give two examples that connect ornamental and relational algebraic constructions via the categorical equivalence.

6.3.1 Parallel composition and the banana-split law

A standard categorical result is that any functor taking part in a categorical equivalence necessarily preserves all universal constructions (limits and colimits, to be precise). The same type of universal constructions in two equivalent categories can then be thought of as being in correspondence. For example: We have seen in **??** that parallel composition gives rise to a pullback square (**??**) in $\mathbb{O}$Rn, which is a product in $\mathbb{O}\text{Rn}\,/\,(I\ ,\ D)$. The functor ClsAlg — being part of a categorical equivalence — preserves products, so the image of (**??**) under ClsAlg in $\mathbb{R}\text{Alg}\ D$ is also a product, and is thus isomorphic to any product in $\mathbb{R}\text{Alg}\ D$; conversely, any product in $\mathbb{R}\text{Alg}\ D$ when mapped to $\mathbb{O}\text{Rn}\,/\,(I\ ,\ D)$ by AlgOrn is also isomorphic to (**??**). Below we look at a consequence of the latter direction.

In $\mathbb{R}\text{Alg}\ D$, a product construction is inspired by the **banana-split law** [Bird and de Moor, 1997, Section 3.1], whose original, functional form is as follows: Define the functorial mapping $\mathbb{F}$-*map* D by

$$\mathbb{F}\text{-}map\ D\ :\ \{Z\ W\ :\ I \to \mathsf{Set}\} \to (Z \rightrightarrows W) \to (\mathbb{F}\ D\ Z \rightrightarrows \mathbb{F}\ D\ W)$$
$$\mathbb{F}\text{-}map\ D\ f\ \{i\}\ =\ mapRD\ (D\ i)\ f$$

For any $f\ :\ \mathbb{F}\ D\ X \rightrightarrows X$ and $g\ :\ \mathbb{F}\ D\ Y \rightrightarrows Y$ we have

$$\mathit{fold}\ f \vartriangle \mathit{fold}\ g \doteq \mathit{fold}\ ((f \circ \mathbb{F}\text{-}\mathit{map}\ D\ \mathit{outl}) \vartriangle (g \circ \mathbb{F}\text{-}\mathit{map}\ D\ \mathit{outr}))$$

The left-hand side of the equation traverses a $(\mu\ D)$-inhabitant by two independent folds, while the right-hand side combines the two traversals into a single fold using a composite algebra. This composite algebra, when generalised straightforwardly to relations, gives rise to a product in $\mathbb{R}\textsc{Alg}\ D$: Let $R : \mathbb{F}\ D\ X \rightsquigarrow X$ and $S : \mathbb{F}\ D\ Y \rightsquigarrow Y$. We define the "banana-split algebra" of R and S by

$$\begin{aligned}&\mathit{BSAlg}\ R\ S : \mathbb{F}\ D\ (X \mathbin{\dot\times} Y) \rightsquigarrow (X \mathbin{\dot\times} Y)\\ &\mathit{BSAlg}\ R\ S\ \mathit{xys}\ (x\ ,\ y) = R\ (\mathbb{F}\text{-}\mathit{map}\ D\ \mathit{outl}\ \mathit{xys})\ x \times S\ (\mathbb{F}\text{-}\mathit{map}\ D\ \mathit{outr}\ \mathit{xys})\ y\end{aligned}$$

where $(X \mathbin{\dot\times} Y)\ i = X\ i \times Y\ i$. We can then construct the following span

$$\begin{array}{ccccc} X & \xleftarrow{\quad \mathit{fun}\ \mathit{outl} \quad} & X \mathbin{\dot\times} Y & \xrightarrow{\quad \mathit{fun}\ \mathit{outr} \quad} & Y \\ {\scriptstyle R}\uparrow & \supseteq & \uparrow{\scriptstyle \mathit{BSAlg}\ R\ S} & \subseteq & \uparrow{\scriptstyle S} \\ \mathbb{F}\ D\ X & \xleftarrow[\mathbb{R}\ D\ (\mathit{fun}\ \mathit{outl})]{} & \mathbb{F}\ D\ (X \mathbin{\dot\times} Y) & \xrightarrow[\mathbb{R}\ D\ (\mathit{fun}\ \mathit{outr})]{} & \mathbb{F}\ D\ Y \end{array}$$

where the homomorphism conditions are simply projections, and it is easy to prove that the span is a product. The categorical equivalence between $\mathbb{R}\textsc{Alg}\ D$ and $\mathbb{O}\textsc{rn}\ /\ (I\ ,\ D)$ implies that the image of the product under $\textsc{AlgOrn}$ is a pullback in $\mathbb{O}\textsc{rn}$:

$$\begin{array}{ccc} \Sigma\ I\ (X \mathbin{\dot\times} Y)\ ,\ \lfloor \mathit{algOD}\ D\ (\mathit{BSAlg}\ R\ S)\rfloor & \xrightarrow{\mathit{id} * \mathit{outr}\ ,\ \_} & \Sigma\ I\ Y\ ,\ \lfloor \mathit{algOD}\ D\ S\rfloor \\ \Big\downarrow{\scriptstyle \mathit{id} * \mathit{outl}\ ,\ \_} & \searrow^{\mathit{outl}\ ,\ \_} & \Big\downarrow{\scriptstyle \mathit{outl}\ ,\ \lceil \mathit{algOD}\ D\ S\rceil} \\ \Sigma\ I\ X\ ,\ \lfloor \mathit{algOD}\ D\ R\rfloor & \xrightarrow[\mathit{outl}\ ,\ \lceil \mathit{algOD}\ D\ R\rceil]{} & I\ ,\ D \end{array}$$

which is isomorphic to the pullback square derived from parallel composition:

$$\begin{array}{ccc} \mathit{outl} \bowtie \mathit{outl}\ ,\ \lfloor \lceil \mathit{algOD}\ D\ R\rceil \otimes \lceil \mathit{algOD}\ D\ S\rceil \rfloor & \xrightarrow{\mathit{outr}_{\bowtie}\ ,\ \_} & \Sigma\ I\ Y\ ,\ \lfloor \mathit{algOD}\ D\ S\rfloor \\ \Big\downarrow{\scriptstyle \mathit{outl}_{\bowtie}\ ,\ \_} & \searrow^{\mathit{pull}\ ,\ \_} & \Big\downarrow{\scriptstyle \mathit{outl}\ ,\ \lceil \mathit{algOD}\ D\ S\rceil} \\ \Sigma\ I\ X\ ,\ \lfloor \mathit{algOD}\ D\ R\rfloor & \xrightarrow[\mathit{outl}\ ,\ \lceil \mathit{algOD}\ D\ R\rceil]{} & I\ ,\ D \end{array}$$

This shows, in particular, that the parallel composition of the two algebraic ornamentations using R and S respectively is isomorphic to the algebraic ornamentation using *BSAlg R S*.

6.3.2 Ornamental algebraic ornamentation

In previous work [Ko and Gibbons, 2011], promotion predicates were computed from ornaments by "ornamental algebraic ornamentation", i.e., algebraic ornamentation with an ornamental algebra. This dissertation, 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 by reasoning in terms of relational algebras.

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

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

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

$$\begin{aligned}&\mathsf{Iso}\ \mathbb{F}\textsc{am}\ (\Sigma\ J\ (\mu\ D \circ e)\ ,\ \mu\ \lfloor algOD\ E\ (fun\ (ornAlg\ O))\rfloor)\\ &\qquad\quad (e \bowtie outl\qquad\ ,\ \mu\ \lfloor O \otimes \lceil S \rceil\rfloor)\end{aligned}$$

where $S = singletonOD\ D$. By isomorphism preservation of Ind, it suffices to establish

$$\begin{aligned}&\mathsf{Iso}\ \mathbb{O}\textsc{rn}\ (\Sigma\ J\ (\mu\ D \circ e)\ ,\ \lfloor algOD\ E\ (fun\ (ornAlg\ O))\rfloor)\\ &\qquad\quad (e \bowtie outl\qquad\ ,\ \lfloor O \otimes \lceil S \rceil\rfloor)\end{aligned} \tag{6.3}$$

We are attempting to establish that an algebraic ornamentation of E is isomorphic to some other description; if that description is also an algebraic ornamentation of E, then we can just reason in terms of their algebras. This is easy: from any ornament $P : \mathsf{Orn}\ f\ E\ \lfloor O \otimes \lceil S \rceil\rfloor$ we get an isomorphism

$$\begin{aligned}&\mathsf{Iso}\ \mathbb{O}\textsc{rn}\ (e \bowtie outl\qquad\ ,\ \lfloor O \otimes \lceil S \rceil\rfloor)\\ &\qquad\quad (\Sigma\ J\ (Inv'\ f)\ ,\ \lfloor algOD\ E\ (clsAlg\ P)\rfloor)\end{aligned}$$

and there is an obvious choice of P — *diffOrn-l* $O\ \lceil S \rceil$. The proof obligation thus reduces to

$$\begin{aligned}&\mathsf{Iso}\ \mathbb{O}\textsc{rn}\ (\Sigma\ J\ (\mu\ D \circ e) \quad , \lfloor\, \mathit{algOD}\ E\ (\mathit{fun}\ (\mathit{ornAlg}\ O))\,\rfloor)\\ &\qquad\quad (\Sigma\ J\ (\mathit{Inv}'\ \mathit{outl}_{\bowtie}) , \lfloor\, \mathit{algOD}\ E\ (\mathit{clsAlg}\ (\mathit{diffOrn\text{-}l}\ O\ \lceil S \rceil))\,\rfloor)\end{aligned}$$

which, by isomorphism preservation of $\textsc{SliceF} \diamond \textsc{AlgOrn}$, is further reduced to

$$\begin{aligned}&\mathsf{Iso}\ (\mathbb{R}\textsc{Alg}\ E)\ (\mu\ D \circ e \quad , \mathit{fun}\ (\mathit{ornAlg}\ O))\\ &\qquad\qquad (\mathit{Inv}'\ \mathit{outl}_{\bowtie}\ , \mathit{clsAlg}\ (\mathit{diffOrn\text{-}l}\ O\ \lceil S \rceil))\end{aligned} \tag{6.4}$$

Through the categorical equivalence, reasoning about datatypes is now reduced to reasoning about algebras. For the carriers, we need to show

$$\mu\ D\ (e\ j) \ \cong\ \Sigma[\, p : e \bowtie \mathit{outl}\,]\ \mathit{outl}_{\bowtie}\ p \ \equiv\ j$$

for every $j : J$, which is easy since the canonical form of p is $(\mathsf{ok}\ j\ ,\ \mathsf{ok}\ (e\ j\ ,\ d))$ for some $d : \mu\ D\ (e\ j)$. Call the left-to-right direction of the isomorphism *wrap*:

$$\begin{aligned}&\mathit{wrap} : \{j : J\} \to \mu\ D\ (e\ j) \to \Sigma[\, p : e \bowtie \mathit{outl}\,]\ \mathit{outl}_{\bowtie}\ p \equiv j\\ &\mathit{wrap}\ \{j\}\ d \ =\ (\mathsf{ok}\ j\ ,\ \mathsf{ok}\ (e\ j\ ,\ d))\ ,\ \mathsf{refl}\end{aligned}$$

For the algebras, however, we have to dig painfully into the definitions. Here is an informal (but already painful) analysis explaining why the two algebras express the same relationship: Let $j : J$, $ds : [\![\ E\ j\ ]\!]\ (\mu\ D \circ e)$, and $d : \mu\ D\ (e\ j)$.

- The type $\mathit{fun}\ (\mathit{ornAlg}\ O)\ ds\ d$ reduces to $\mathsf{con}\ (\mathit{erase}\ O\ ds) \ \equiv\ d$. Matching d with $\mathsf{con}\ ds'$ where $ds' : [\![\ D\ (e\ j)\ ]\!]\ (\mu\ D)$, we get to an isomorphic type $\mathit{erase}\ O\ ds \ \equiv\ ds'$. Since *erase* modifies only the shape part, this decomposes into two conditions: (i) that the result of applying $\mathit{erase}_{\mathsf{S}}\ (O\ (\mathsf{ok}\ j))$ to the shape part of ds is equal to the shape part of ds', and (ii) that the series of $(\mu\ D)$-inhabitants contained in ds and ds' are equal.
- On the other hand, setting $\mathit{wds} \ =\ \mathit{mapRD}\ (E\ j)\ \mathit{wrap}\ ds$, a proof of type

  $$\mathit{clsAlg}\ (\mathit{diffOrn\text{-}l}\ O\ \lceil S \rceil)\ \mathit{wds}\ (\mathit{wrap}\ (\mathsf{con}\ ds'))$$

  contains a shape of type

  $$\mathsf{S}\ (\mathit{toRDesc}\ (\mathit{pcROD}\ (O\ (\mathsf{ok}\ j))\ (\lceil S \rceil\ (\mathsf{ok}\ (e\ j\ ,\ \mathsf{con}\ ds')))))$$

  satisfying the associated equations. By *pcROD-iso*, this decomposes into two shapes $ts : \mathsf{S}\ (E\ j)$ and $ss : \mathsf{S}\ (\lfloor S \rfloor\ (e\ j\ ,\ \mathsf{con}\ ds'))$ such that

  $$\mathit{erase}_{\mathsf{S}}\ (O\ (\mathsf{ok}\ j))\ ts \ \equiv\ \mathit{erase}_{\mathsf{S}}\ (\lceil S \rceil\ (\mathsf{ok}\ (e\ j\ ,\ \mathsf{con}\ ds')))\ ss$$

The right-hand side is equal to the shape part of ds', and by the first equation associated with ts, the left-hand side is equal to the result of applying $erase_{\mathsf{S}}\ (O\ (\mathsf{ok}\ j))$ to the shape part of wds, i.e., the shape part of ds — this corresponds to condition (i). For condition (ii), $\lfloor S \rfloor$ uses the $(\mu\ D)$-inhabitants contained in ds' as the indices at the recursive positions, which, by the second equation associated with ts, are just the $(\mu\ D)$-inhabitants contained in wds, i.e., those contained in ds.

6.4 Discussion

As mentioned in the beginning, the results of this chapter are only partially formalised. Specifically, the formalised results include

- everything in Section 6.1, (except pull-back)
- general definition of categorical equivalences (which, unlike general isomorphisms of categories, can be easily defined with our formalisation of categories),
- definition of $\mathbb{R}$RAlg,
- the isomorphism *algROD-iso*,
- the completeness theorem (6.1) for a different version of classifying algebras, and
- the banana-split algebra as a product in $\mathbb{R}$RAlg.

The rest are worked out on paper, skipping over tedious type-theoretic detail. In general, while the internalist typing of constructions about descriptions and ornaments works well in **??**, ensuring that every expression we write down makes sense, the encoded constraints need to be reasoned about non-trivially in this chapter, and this is greatly hindered because the constraints are implicitly coupled with the expressions in a particular way — here externalism can be much more helpful. (For a simplest example, in this chapter we silently abandon the internalist type Inv and switch to the externalist type Inv'; Goal 3 is much easier to discharge as a consequence.)

The categorical equivalence prompts us to contemplate again why we need ornaments: if relational algebras can express the same refinement relationships between datatypes as ornaments and can offer corresponding constructions as shown in Section 6.3.1, what extra benefit do ornaments provide that justifies their existence? Firstly, relational algebras describe how to classify inhabitants of existing datatypes to obtain more informative datatypes, and hence correspond more closely to ornamental descriptions, which are more restrictive than ornaments as explained in the beginning of **??** — indeed, the categorical equivalence is between $\mathbb{R}$RAlg and the slice category on top of $\mathbb{O}$Rn. Even when we restrict the comparison to one between relational algebras and ornamental descriptions, the latter still offer finer-grained control of representation, down to the level of individual fields. For example, while we can get a datatype of vectors by specialising AlgList with a suitable algebra, the index-first, representation-optimised version can only be obtained via ornamentation. We can thus think of ornamental descriptions as a form of relational algebras that allow us to additionally specify representation optimisation, and the categorical equivalence lets us switch between the two forms of specifications of datatype refinement relationships.

The example about ornamental algebraic ornamentation in Section 6.3.2 shows that we can reduce reasoning about datatypes to reasoning about algebras via the categorical equivalence. The discharging of the proof obligation (6.4) is unsatisfactory, however — discharging the earlier proof obligation (6.3) could in fact be more straightforward. For the reduction to be justified, reasoning about algebras has to be easier, e.g., in the calculational style of Bird and de Moor [1997]. This requires formulation of calculational laws for algebras defined datatype-generically on top of the universes of descriptions and ornaments; furthermore, these laws have to be strengthened to establish isomorphisms — rather than merely bi-implications — between algebras. (The lemma *fun-preserves-fold* stated at the end of **??** is an example.) In short: the categorical equivalence points out not only the possibility of reasoning about datatypes algebraically, but also the need for algebraic reasoning to receive a proof-relevant revision for that purpose.

Bibliography

Richard Bird and Oege de Moor [1997]. *Algebra of Programming*. Prentice-Hall. ↰ pages 3, 18, and 23

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 20

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

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

Todo list