Chapter 4

Categorical organisation of the ornament–refinement framework

?? left some obvious holes in the theory of ornaments. For instance:

- When it comes to composition of ornaments, the following **sequential composition** is probably the first that comes to mind (rather than parallel composition), which is evidence that the ornamental relation is transitive:

  $$
  \begin{aligned}
  \_\odot\_ \;:\; & \{I\;J\;K \;:\; \mathsf{Set}\}\;\{e \;:\; J \to I\}\;\{f \;:\; K \to J\} \to \\
  & \{D \;:\; \mathsf{Desc}\;I\}\;\{E \;:\; \mathsf{Desc}\;J\}\;\{F \;:\; \mathsf{Desc}\;K\} \to \\
  & \mathsf{Orn}\;e\;D\;E \to \mathsf{Orn}\;f\;E\;F \to \mathsf{Orn}\;(e \circ f)\;D\;F
  \end{aligned}
  $$

  -- definition in Figure 4.4

  Correspondingly, we expect that

  $$\mathit{forget}\;(O \odot P) \qquad \text{and} \qquad \mathit{forget}\;O \circ \mathit{forget}\;P$$

  are extensionally equal. That is, the sequential compositional structure of ornaments corresponds to the compositional structure of forgetful functions. We wish to state such correspondences in concise terms.

- While parallel composition of ornaments (**??**) has a sensible definition, it is defined by case analysis at the microscopic level of individual fields. Such a microscopic definition is difficult to comprehend, and so are any subsequent definitions and proofs. It is desirable to have a macroscopic characterisation of parallel composition, so the nature of parallel composition is immediately clear, and subsequent definitions and proofs can be done in a more abstract manner.

- The ornamental conversion isomorphisms (**??**) and the modularity isomorphisms (**??**) were left unimplemented. Both sets of isomorphisms are about the optimised predicates (**??**), which are defined in terms of parallel composition with singleton ornamentation (**??**). We thus expect that the existence of these isomorphisms can be explained in terms of properties of parallel composition and singleton ornamentation.

A lightweight organisation of the ornament–refinement framework in basic category theory [Mac Lane, 1998] can help to fill up all these holes. In more detail:

- Categories and functors are abstractions for compositional structures and structure-preserving maps between them. Facts about translations between ornaments, refinements, and functions can thus be neatly organised under the categorical language (Section 4.1).
- Parallel composition merges two compatible ornaments and does nothing more; in other words, it computes the least informative ornament that contains the information of both ornaments. Characterisation of such **universal constructions** is a speciality of category theory; in our case, parallel composition can be shown to be a categorical **pullback** (Section 4.2).
- Universal constructions are unique up to isomorphism, so it is convenient to establish isomorphisms about universal constructions. The status of parallel composition being a pullback can thus help to construct the ornamental conversion isomorphisms (in Section 4.3.1) and the modularity isomorphisms (in Section 4.3.2).

Section 4.4 concludes with some discussion.

$$
\begin{tikzcd}
& \mathbb{O}\textsc{rn} \arrow[dl, bend right, "\textsc{RSem}"'] \arrow[d, "\textsc{Ind}"] & \\
\mathbb{F}\textsc{ref} \arrow[r, bend left, "\textsc{FRefF}"] \arrow[r, phantom, "\cong"] & \mathbb{F}\textsc{am} \arrow[l, bend left, "\textsc{FRefC}"] \arrow[r, bend left, "\textsc{Com}"] \arrow[r, bend right, "\textsc{FamF}"'] & \mathbb{F}\textsc{un}
\end{tikzcd}
$$

Figure 4.1 Categories and functors for the ornament–refinement framework.

4.1 Categories and functors

A first approximation of a category is a (directed multi-) **graph**, which consists of a set of objects (nodes) and a collection of sets of morphisms (edges) indexed with their source and target objects:

$$
\begin{array}{l}
\textbf{record}\ \mathsf{Graph}\ \{l\ m\ :\ \mathsf{Level}\}\ :\ \mathsf{Set}\ (\mathsf{suc}\ (l \sqcup m))\ \textbf{where} \\
\quad \textbf{field} \\
\qquad \mathit{Object}\ :\ \mathsf{Set}\ l \\
\qquad \_{\Rightarrow}\_\quad :\ \mathit{Object} \to \mathit{Object} \to \mathsf{Set}\ m
\end{array}
$$

For example, the underlying graph of the category $\mathbb{F}\textsc{un}$ of (small) sets and (total) functions is just

$$
\begin{array}{l}
\mathbb{F}\textsc{un}\text{-}\mathit{graph}\ :\ \mathsf{Graph} \\
\mathbb{F}\textsc{un}\text{-}\mathit{graph}\ =\ \textbf{record}\ \{\ \mathit{Object}\ =\ \mathsf{Set} \\
\qquad\qquad\qquad\qquad\qquad ;\ \_{\Rightarrow}\_\ \ =\ \lambda\, A\ B \mapsto A \to B\ \}
\end{array}
$$

A category is a graph whose morphisms are equipped with a monoid-like compositional structure — there is a morphism composition operator of type

$$
\_\cdot\_\ :\ \{X\ Y\ Z\ :\ \mathit{Object}\} \to (Y \Rightarrow Z) \to (X \Rightarrow Y) \to (X \Rightarrow Z)
$$

which has identities and is associative.

Syntactic remark (*universe polymorphism*). Many definitions in this chapter (like Graph above) employ AGDA's **universe polymorphism** [Harper and Pollack, 1991], so the definitions can be instantiated at suitable levels of the Set hierarchy as needed. (For example, the type of $\mathbb{F}$UN-*graph* is implicitly instantiated as Graph $\{1\}$ $\{1\}$, since both Set and any $A \to B$ (where A, B : Set) are of type Set_1.) We will give the first few universe-polymorphic definitions with full detail about the levels, but will later suppress the syntactic noise wherever possible. □

Before we move on to the definition of categories, though, special attention must be paid to equality on morphisms, which is usually coarser than definitional equality — in $\mathbb{F}$UN, for example, it is necessary to identify functions up to extensional equality (so uniqueness of morphisms in universal properties would make sense). One ad hoc way to achieve this in AGDA's intensional setting is to use **setoids** [Barthe et al., 2003] — sets with an explicitly specified equivalence relation — to represent sets of morphisms. The type of setoids can be defined as a record which contains a carrier set, an equivalence relation on the set, and the three laws for the equivalence relation:

```
record Setoid {c d : Level} : Set (suc (c ⊔ d)) where
  field
    Carrier : Set c
    _≈_     : Carrier → Carrier → Set d
    refl  : {x : Carrier} → x ≈ x
    sym   : {x y : Carrier} → x ≈ y → y ≈ x
    trans : {x y z : Carrier} → x ≈ y → y ≈ z → x ≈ z
```

For example, we can define a setoid of functions that uses extensional equality:

```
FunSetoid : Set → Set → Setoid
FunSetoid A B = record { Carrier = A → B
                       ; _≈_     = _≐_
                       ; proofs of laws }
```

Proofs of the three laws are omitted from the presentation.

Similarly, we can define the type of categories as a record containing a set of objects, a collection of **setoids** of morphisms indexed by source and target

```
record Category {l m n : Level} : Set (suc (l ⊔ m ⊔ n)) where
  field
    Object    : Set l
    Morphism  : Object → Object → Setoid {m} {n}

  _⇒_ : Object → Object → Set m
  X ⇒ Y = Setoid.Carrier (Morphism X Y)
  _≈_ : {X Y : Object} → (X ⇒ Y) → (X ⇒ Y) → Set n
  _≈_ {X} {Y} = Setoid._≈_ (Morphism X Y)

  field
    _·_ : {X Y Z : Object} → (Y ⇒ Z) → (X ⇒ Y) → (X ⇒ Z)
    id  : {X : Object} → (X ⇒ X)
    id-l   : {X Y : Object} (f : X ⇒ Y) →
             id · f ≈ f
    id-r   : {X Y : Object} (f : X ⇒ Y) →
             f · id ≈ f
    assoc  : {X Y Z W : Object} (f : Z ⇒ W) (g : Y ⇒ Z) (h : X ⇒ Y) →
             (f · g) · h ≈ f · (g · h)
    cong-l : {X Y Z : Object} {f g : Y ⇒ Z} (h : X ⇒ Y) →
             f ≈ g → f · h ≈ g · h
    cong-r : {X Y Z : Object} (h : Y ⇒ Z) {f g : X ⇒ Y} →
             f ≈ g → h · f ≈ h · g
```

Figure 4.2 Definition of categories.

objects, the composition operator on morphisms, the identity morphisms, and the identity and associativity laws for composition. The definition is shown in Figure 4.2. Two notations are introduced to improve readability: $X \Rightarrow Y$ is defined to be the carrier set of the setoid of morphisms from X to Y, and $f \approx g$ is defined to be the equivalence between the morphisms f and g as specified by the setoid to which f and g belong. The last two laws *cong-l* and *cong-r* require morphism composition to preserve the equivalence on morphisms; they are given in this form to work better with the equational reasoning combinators commonly used in AGDA (see, for example, the AoPA library [Mu et al., 2009]).

Now we can define the category $\mathbb{F}\text{UN}$ of sets and functions as

$$
\begin{array}{l}
\mathbb{F}\text{UN} \;:\; \mathsf{Category} \\
\mathbb{F}\text{UN} \;=\; \textbf{record}\;\{\; \mathit{Object} \;=\; \mathsf{Set} \\
\qquad\qquad\qquad\;\; ;\; \mathit{Morphism} \;=\; \mathit{FunSetoid} \\
\qquad\qquad\qquad\;\; ;\; \_\cdot\_ \;=\; \_\circ\_ \\
\qquad\qquad\qquad\;\; ;\; \mathit{id} \;=\; \lambda\, x \mapsto x \\
\qquad\qquad\qquad\;\; ;\; \boxed{\text{proofs of laws}}\; \}
\end{array}
$$

Another important category that we will make use of is $\mathbb{F}\text{AM}$, the category of indexed families of sets and indexed families of functions, which is useful for talking about componentwise structures. An object in $\mathbb{F}\text{AM}$ has type $\Sigma[\, I : \mathsf{Set}\,]\; I \to \mathsf{Set}$, i.e., it is a set I and a family of sets indexed by I; a morphism from $(J\,,\,Y)$ to $(I\,,\,X)$ is a function $e \,:\, J \to I$ and a family of functions from $Y\,j$ to $X\,(e\,j)$ for each $j \,:\, J$.

$$
\begin{array}{l}
\mathbb{F}\text{AM} \;:\; \mathsf{Category} \\
\mathbb{F}\text{AM} \;=\; \textbf{record} \\
\quad \{\; \mathit{Object} \;=\; \Sigma[\, I : \mathsf{Set}\,]\; I \to \mathsf{Set} \\
\quad ;\; \mathit{Morphism} \;=\; \lambda\,(J\,,\,Y)\;(I\,,\,X) \mapsto \textbf{record} \\
\qquad\qquad\qquad\qquad \{\; \mathit{Carrier} \;=\; \Sigma[\, e : J \to I\,]\; Y \rightrightarrows (X \circ e) \\
\qquad\qquad\qquad\qquad ;\; \_\approx\_ \;=\; \lambda\,(e\,,\,u)\;(e'\,,\,u') \mapsto \\
\qquad\qquad\qquad\qquad\qquad\qquad (e \doteq e') \times ((j \,:\, J) \to u\,\{j\} \approxeq u'\,\{j\}) \\
\qquad\qquad\qquad\qquad ;\; \boxed{\text{proofs of laws}}\; \} \\
\quad ;\; \_\cdot\_ \;=\; \lambda\,(e\,,\,u)\;(f\,,\,v) \mapsto (e \circ f)\,,\,(\lambda\,\{k\} \mapsto u\,\{f\,k\} \circ v\,\{k\})
\end{array}
$$

$$
\begin{array}{l}
\textbf{record}\ \mathsf{Functor}\ \{l\ m\ n\ l'\ m'\ n'\ :\ \mathsf{Level}\} \\
\quad (C\ :\ \mathsf{Category}\ \{l\}\ \{m\}\ \{n\})\ (D\ :\ \mathsf{Category}\ \{l'\}\ \{m'\}\ \{n'\})\ : \\
\quad \mathsf{Set}\ (l \sqcup m \sqcup n \sqcup l' \sqcup m' \sqcup n')\ \textbf{where} \\
\quad \textbf{field} \\
\qquad \mathit{object}\ :\ \mathit{Object}\ C \to \mathit{Object}\ D \\
\qquad \mathit{morphism}\ :\ \{X\ Y\ :\ \mathit{Object}\ C\} \to X \Rightarrow_C Y \to \mathit{object}\ X \Rightarrow_D \mathit{object}\ Y \\
\qquad \mathit{equiv\text{-}preserving}\ :\ \{X\ Y\ :\ \mathit{Object}\ C\}\ \{f\ g\ :\ X \Rightarrow_C Y\} \to \\
\qquad\qquad f\ \approx_C\ g \to \mathit{morphism}\ f\ \approx_D\ \mathit{morphism}\ g \\
\qquad \mathit{id\text{-}preserving}\ :\ \{X\ :\ \mathit{Object}\ C\} \to \\
\qquad\qquad \mathit{morphism}\ (\mathit{id}\ C\ \{X\})\ \approx_D\ \mathit{id}\ D\ \{\mathit{object}\ X\} \\
\qquad \mathit{comp\text{-}preserving}\ :\ \{X\ Y\ Z\ :\ \mathit{Object}\ C\}\ (f\ :\ Y \Rightarrow_C Z)\ (g\ :\ X \Rightarrow_C Y) \to \\
\qquad\qquad \mathit{morphism}\ (f \cdot_C g)\ \approx_D\ (\mathit{morphism}\ f \cdot_D \mathit{morphism}\ g)
\end{array}
$$

Figure 4.3 Definition of functors. Subscripts are used to indicate to which category an operator belongs.

$$
\begin{array}{l}
;\ \mathit{id}\ =\ (\lambda\, x\ \mapsto\ x)\ ,\ (\lambda\ \{i\}\ x\ \mapsto\ x) \\
;\ \boxed{\text{proofs of laws}}\ \}
\end{array}
$$

Note that the equivalence on morphisms is defined to be componentwise extensional equality, which is formulated with the help of McBride's "John Major" heterogeneous equality $\_\cong\_$ [McBride, 1999] — the equivalence $\_\dot{\cong}\_$ is defined by $g\ \dot{\cong}\ h\ =\ \forall\, x \to g\, x\ \cong\ h\, x$. (Given $y\ :\ Y\, j$ for some $j\ :\ J$, the types of $u\ \{j\}\ y$ and $u'\ \{j\}\ y$ are not definitionally equal but only provably equal, so it is necessary to employ heterogeneous equality.)

Categories are graphs with a compositional structure, and **functors** are transformations between categories that preserve the compositional structure. The definition of functors is shown in Figure 4.3: a functor consists of two mappings, one on objects and the other on morphisms, where the morphism part preserves all structures on morphisms, including equivalence, identity, and composition. For example, we have two functors from $\mathbb{F}\textsc{am}$ to $\mathbb{F}\textsc{un}$, one summing components together

$$\begin{aligned}
&\textsc{Com} \;:\; \mathsf{Functor}\ \mathbb{F}\textsc{am}\ \mathbb{F}\textsc{un} \quad \text{-- the comprehension functor}\\
&\textsc{Com} \;=\; \textbf{record}\ \{\ \mathit{object} \quad = \lambda\,(I\,,X) \mapsto \Sigma\, I\, X\\
&\qquad\qquad\qquad ;\ \mathit{morphism} = \lambda\,(e\,,u\,) \mapsto e * u\\
&\qquad\qquad\qquad ;\ \text{proofs of laws}\ \}
\end{aligned}$$

and the other extracting the index part.

$$\begin{aligned}
&\textsc{FamF} \;:\; \mathsf{Functor}\ \mathbb{F}\textsc{am}\ \mathbb{F}\textsc{un} \quad \text{-- the family fibration functor}\\
&\textsc{FamF} \;=\; \textbf{record}\ \{\ \mathit{object} \quad = \lambda\,(I\,,X) \mapsto I\\
&\qquad\qquad\qquad ;\ \mathit{morphism} = \lambda\,(e\,,u\,) \mapsto e\\
&\qquad\qquad\qquad ;\ \text{proofs of laws}\ \}
\end{aligned}$$

The functor laws should be proved for both functors alongside their object and morphism maps. In particular, we need to prove that the morphism part preserves equivalence: for Com, this means we need to prove, for all $e : J \to I$, $u : Y \rightrightarrows (X \circ e)$ and $f : J \to I$, $v : Y \rightrightarrows (X \circ f)$, that

$$(e \doteq f) \times ((j : J) \to u\,\{j\} \stackrel{.}{\approx} v\,\{j\}) \to (e * u \doteq f * v)$$

and for FamF we need to prove

$$(e \doteq f) \times ((j : J) \to u\,\{j\} \stackrel{.}{\approx} v\,\{j\}) \to (e \doteq f)$$

both of which can be easily discharged.

Categories for refinements and ornaments

Some constructions in **??** can now be organised under several categories and functors. For a start, we already saw that refinements are interesting only because of their intensional contents; this is reflected in an isomorphism of categories between the category $\mathbb{F}$am and the category $\mathbb{F}$ref of type families and refinement families (i.e., there are two functors back and forth inverse to each other). An object in $\mathbb{F}$ref is an indexed family of sets as in $\mathbb{F}$am, and a morphism from $(J\,,Y)$ to $(I\,,X)$ consists of a function $e : J \to I$ on the indices and a refinement family of type $\mathsf{FRefinement}\ e\ X\ Y$. As for the equivalence on morphisms, it suffices to use extensional equality on the index functions and componentwise extensional equality on refinement families, where extensional equality on refinements means extensional equality on their forgetful functions (extracted by $\mathsf{Refinement}.\mathit{forget}$), which we have shown in **??** to be the core of refinements. Formally:

$$
\begin{array}{l}
\mathbb{F}\textsc{Ref} : \mathsf{Category} \\
\mathbb{F}\textsc{Ref} = \textbf{record} \\
\quad \{\ \mathit{Object} = \Sigma[\, I : \mathsf{Set}\,]\ I \to \mathsf{Set} \\
\quad ;\ \mathit{Morphism} = \lambda\,(J , Y)\ (I , X) \mapsto \textbf{record} \\
\qquad\qquad \{\ \mathit{Carrier} = \Sigma[\, e : J \to I\,]\ \mathsf{FRefinement}\ e\ X\ Y \\
\qquad\qquad ;\ \_{\approx}\_ = \lambda\,(e , rs)\ (e' , rs') \mapsto \\
\qquad\qquad\qquad (e \doteq e') \times \\
\qquad\qquad\qquad ((j : J) \to \mathsf{Refinement}.\mathit{forget}\ (rs\ (\mathsf{ok}\ j)) \mathrel{\dot{\approx}} \\
\qquad\qquad\qquad\qquad \mathsf{Refinement}.\mathit{forget}\ (rs'\ (\mathsf{ok}\ j))) \\
\qquad\qquad ;\ \boxed{\text{proofs of laws}}\ \} \\
\quad ;\ \boxed{\text{proofs of laws}}\ \}
\end{array}
$$

Note that a refinement family from $X : I \to \mathsf{Set}$ to $Y : J \to \mathsf{Set}$ is deliberately cast as a morphism in the opposite direction from (J , Y) to (I , X); think of this as suggesting the direction of the forgetful functions of refinements. $\mathbb{F}\textsc{Ref}$ is no more powerful than $\mathbb{F}\textsc{am}$ since $\mathbb{F}\textsc{Ref}$ ignores the intensional contents of refinements by using an extensional equality, and consequently there are two functors between $\mathbb{F}\textsc{Ref}$ and $\mathbb{F}\textsc{am}$ that are inverse to each other, forming an isomorphism of categories:

- We have a forgetful functor $\mathrm{FR}\textsc{ef}\mathrm{F} : \mathsf{Functor}\ \mathbb{F}\textsc{Ref}\ \mathbb{F}\textsc{am}$ which is identity on objects and componentwise $\mathsf{Refinement}.\mathit{forget}$ on morphisms (which preserves equivalence automatically):

$$
\begin{array}{l}
\mathrm{FR}\textsc{ef}\mathrm{F} : \mathsf{Functor}\ \mathbb{F}\textsc{Ref}\ \mathbb{F}\textsc{am} \\
\mathrm{FR}\textsc{ef}\mathrm{F} = \textbf{record} \\
\quad \{\ \mathit{object} = \mathit{id} \\
\quad ;\ \mathit{morphism} = \lambda\,(e , rs) \mapsto e , (\lambda\, j \mapsto \mathsf{Refinement}.\mathit{forget}\ (rs\ (\mathsf{ok}\ j))) \\
\quad ;\ \boxed{\text{proofs of laws}}\ \}
\end{array}
$$

  Note that $\mathrm{FR}\textsc{ef}\mathrm{F}$ remains a familiar covariant functor rather than a contravariant one because of our choice of morphism direction.

- Conversely, there is a functor FRefC : Functor $\mathbb{F}$am $\mathbb{F}$ref whose object part is identity and whose morphism part is componentwise *canonRef*:

  $$\begin{array}{l}
  \text{FRefC} \;:\; \mathsf{Functor}\ \mathbb{F}\textsc{am}\ \mathbb{F}\textsc{ref} \\
  \text{FRefC} \;=\; \textbf{record} \\
  \quad \{\ \mathit{object} \quad\;\; = \ \mathit{id} \\
  \quad ;\ \mathit{morphism} \;=\; \lambda\ \{\ (e\ ,u) \ \mapsto\ e\ , \lambda\ \{\ (\mathsf{ok}\ j) \ \mapsto\ \mathit{canonRef}\ (u\ \{j\})\ \}\ \} \\
  \quad ;\ \text{proofs of laws}\,\}
  \end{array}$$

The two functors FRefF and FRefC are inverse to each other by definition.

There is another category $\mathbb{O}$rn, which has objects of type $\Sigma[\,I : \mathsf{Set}\,]\ \mathsf{Desc}\ I$, i.e., descriptions paired with index sets, and morphisms from $(J\ , E)$ to $(I\ , D)$ of type $\Sigma[\,e\ :\ J\ \to\ I\,]\ \ \mathsf{Orn}\ \ e\ \ D\ \ E$, i.e., ornaments paired with index erasure functions. To complete the definition of $\mathbb{O}$rn:

- We need to devise an equivalence on ornaments

  $$\begin{array}{l}
  \mathit{OrnEq} \;:\; \{I\,J\ :\ \mathsf{Set}\}\ \{e\,f\ :\ J \to I\}\ \{D\ :\ \mathsf{Desc}\ I\}\ \{E\ :\ \mathsf{Desc}\ J\} \to \\
  \qquad\qquad \mathsf{Orn}\ e\ D\ E \to \mathsf{Orn}\ f\ D\ E \to \mathsf{Set}
  \end{array}$$

  such that it implies extensional equality of e and f and that of ornamental forgetful functions:

  $$\begin{array}{l}
  \mathit{OrnEq\text{-}forget} \;:\; \{I\,J\ :\ \mathsf{Set}\}\ \{e\,f\ :\ J \to I\}\ \{D\ :\ \mathsf{Desc}\ I\}\ \{E\ :\ \mathsf{Desc}\ J\} \to \\
  \qquad\qquad (O\ :\ \mathsf{Orn}\ e\ D\ E)\ (P\ :\ \mathsf{Orn}\ f\ D\ E) \to \mathit{OrnEq}\ O\ P \to \\
  \qquad\qquad (e\ \doteq\ f) \times ((j\ :\ J) \to \mathit{forget}\ O\ \{j\} \approxeq \mathit{forget}\ P\ \{j\})
  \end{array}$$

  The actual definition of *OrnEq* is deferred to **??**.

- Morphism composition is sequential composition $\_\odot\_$, which merges two successive batches of modifications in a straightforward way. The definition is shown in Figure 4.4. There is also a family of **identity ornaments**:

  $$\begin{array}{l}
  \mathit{idOrn} \;:\; \{I\ :\ \mathsf{Set}\}\ (D\ :\ \mathsf{Desc}\ I) \to \mathsf{Orn}\ \mathit{id}\ D\ D \\
  \mathit{idOrn}\ \{I\}\ D\ (\mathsf{ok}\ i) \;=\; \mathit{idROrn}\ (D\ i) \\
  \quad \textbf{where} \\
  \qquad \mathbb{E}\text{-}\mathit{refl} \;:\; (\mathit{is}\ :\ \mathsf{List}\ I) \to \mathbb{E}\ \mathit{id}\ \mathit{is}\ \mathit{is} \\
  \qquad \mathbb{E}\text{-}\mathit{refl}\ [\,] \qquad\;\; = \ [\,] \\
  \qquad \mathbb{E}\text{-}\mathit{refl}\ (i :: \mathit{is}) \;=\; \mathsf{refl} :: \mathbb{E}\text{-}\mathit{refl}\ \mathit{is}
  \end{array}$$

$$
\begin{array}{l}
\mathbb{E}\text{-}\mathit{trans} : \{I\ J\ K : \mathsf{Set}\}\ \{e : J \to I\}\ \{f : K \to J\} \to \\
\qquad\qquad \{is : \mathsf{List}\ I\}\ \{js : \mathsf{List}\ J\}\ \{ks : \mathsf{List}\ K\} \to \\
\qquad\qquad \mathbb{E}\ e\ js\ is \to \mathbb{E}\ f\ ks\ js \to \mathbb{E}\ (e \circ f)\ ks\ is \\
\mathbb{E}\text{-}\mathit{trans}\qquad\quad [\,]\qquad\qquad [\,]\qquad\qquad = [\,] \\
\mathbb{E}\text{-}\mathit{trans}\ \{e := e\}\ (eeq :: eeqs)\ (feq :: feqs) = \mathit{trans}\ (\mathit{cong}\ e\ feq)\ eeq :: \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \mathbb{E}\text{-}\mathit{trans}\ eeqs\ feqs
\end{array}
$$

$$
\begin{array}{l}
\mathit{scROrn} : \{I\ J\ K : \mathsf{Set}\}\ \{e : J \to I\}\ \{f : K \to J\} \to \\
\qquad\qquad \{D : \mathsf{RDesc}\ I\}\ \{E : \mathsf{RDesc}\ J\}\ \{F : \mathsf{RDesc}\ K\} \to \\
\qquad\qquad \mathsf{ROrn}\ e\ D\ E \to \mathsf{ROrn}\ f\ E\ F \to \mathsf{ROrn}\ (e \circ f)\ D\ F
\end{array}
$$

$$
\begin{array}{lllll}
\mathit{scROrn}\ (\mathsf{v}\ eeqs) & (\mathsf{v}\ feqs) & = & \mathsf{v}\ (\mathbb{E}\text{-}\mathit{trans}\ eeqs\ feqs) & \\
\mathit{scROrn}\ (\mathsf{v}\ eeqs) & (\Delta\ T\ P) & = & \Delta[\,t : T\,]\ \ \mathit{scROrn}\ (\mathsf{v}\ eeqs) & (P\ t) \\
\mathit{scROrn}\ (\sigma\ S\ O) & (\sigma\ .S\ P) & = & \sigma[\,s : S\,]\ \ \mathit{scROrn}\ (O\ s) & (P\ s) \\
\mathit{scROrn}\ (\sigma\ S\ O) & (\Delta\ T\ P) & = & \Delta[\,t : T\,]\ \ \mathit{scROrn}\ (\sigma\ S\ O) & (P\ t) \\
\mathit{scROrn}\ (\sigma\ S\ O) & (\nabla\ s\ P) & = & \nabla[\,s\,]\ \ \mathit{scROrn}\ (O\ s) & P \\
\mathit{scROrn}\ (\Delta\ T\ O) & (\sigma\ .T\ P) & = & \Delta[\,t : T\,]\ \ \mathit{scROrn}\ (O\ t) & (P\ t) \\
\mathit{scROrn}\ (\Delta\ T\ O) & (\Delta\ U\ P) & = & \Delta[\,u : U\,]\ \ \mathit{scROrn}\ (\Delta\ T\ O) & (P\ u) \\
\mathit{scROrn}\ (\Delta\ T\ O) & (\nabla\ t\ P) & = & \qquad\qquad\ \mathit{scROrn}\ (O\ t) & P \\
\mathit{scROrn}\ (\nabla\ s\ O) & P & = & \nabla[\,s\,]\ \ \mathit{scROrn}\ O & P
\end{array}
$$

$$
\begin{array}{l}
\_\odot\_ : \{I\ J\ K : \mathsf{Set}\}\ \{e : J \to I\}\ \{f : K \to J\} \to \\
\qquad \{D : \mathsf{Desc}\ I\}\ \{E : \mathsf{Desc}\ J\}\ \{F : \mathsf{Desc}\ K\} \to \\
\qquad \mathsf{Orn}\ e\ D\ E \to \mathsf{Orn}\ f\ E\ F \to \mathsf{Orn}\ (e \circ f)\ D\ F \\
\_\odot\_\ \{f := f\}\ O\ P\ (\mathsf{ok}\ k) = \mathit{scROrn}\ (O\ (\mathsf{ok}\ (f\ k)))\ (P\ (\mathsf{ok}\ k))
\end{array}
$$

Figure 4.4 Definitions for sequential composition of ornaments.

$$
\begin{array}{lcl}
\mathit{idROrn} : (E : \mathsf{RDesc}\ I) \to \mathsf{ROrn}\ \mathit{id}\ E\ E \\
\mathit{idROrn}\ (\mathsf{v}\ \mathit{is}) & = & \mathsf{v}\ (\mathbb{E}\text{-}\mathit{refl}\ \mathit{is}) \\
\mathit{idROrn}\ (\sigma\ S\ E) & = & \sigma[\,s : S\,]\ \mathit{idROrn}\ (E\ s)
\end{array}
$$

which simply use σ and v everywhere to express that a description is identical to itself. Unsurprisingly, the identity ornaments serve as identity of sequential composition.

To summarise:

$$
\begin{array}{l}
\mathbb{O}\textsc{rn} : \mathsf{Category} \\
\mathbb{O}\textsc{rn} = \textbf{record} \\
\quad \{\ \mathit{Object} \quad = \Sigma[\,I : \mathsf{Set}\,]\ \mathsf{Desc}\ I \\
\quad ;\ \mathit{Morphism} = \lambda\,(J\,,E)\ (I\,,D) \mapsto \textbf{record} \\
\qquad\qquad\qquad \{\ \mathit{Carrier} = \Sigma[\,e : J \to I\,]\ \mathsf{Orn}\ e\ D\ E \\
\qquad\qquad\qquad ;\ \_{\approx}\_ \quad = \lambda\,(e\,,O)\ (f\,,P) \mapsto \mathit{OrnEq}\ O\ P \\
\qquad\qquad\qquad ;\ \boxed{\text{proofs of laws}}\ \} \\
\quad ;\ \_{\cdot}\_ = \lambda\,(e\,,O)\ (f\,,P) \mapsto (e \circ f)\,,(O \odot P) \\
\quad ;\ \mathit{id} \quad = \lambda\,\{I\,,D\} \mapsto \mathit{id}\,,\mathit{idOrn}\ D \\
\quad ;\ \boxed{\text{proofs of laws}}\ \}
\end{array}
$$

A functor $\textsc{Ind} : \mathsf{Functor}\ \mathbb{O}\textsc{rn}\ \mathbb{F}\textsc{am}$ can then be constructed, which gives the ordinary semantics of descriptions and ornaments: the object part of $\textsc{Ind}$ decodes a description $(I\,,D)$ to its least fixed point $(I\,,\mu\,D)$, and the morphism part translates an ornament $(e\,,O)$ to the forgetful function $(e\,,\mathit{forget}\ O)$, the latter preserving equivalence by virtue of *OrnEq-forget*.

$$
\begin{array}{l}
\textsc{Ind} : \mathsf{Functor}\ \mathbb{O}\textsc{rn}\ \mathbb{F}\textsc{am} \\
\textsc{Ind} = \textbf{record}\ \{\ \mathit{object} \quad = \lambda\,(I\,,D) \mapsto I\,,\mu\,D \\
\qquad\qquad\quad ;\ \mathit{morphism} = \lambda\,(e\,,O) \mapsto e\,,\mathit{forget}\ O \\
\qquad\qquad\quad ;\ \boxed{\text{proofs of laws}}\ \}
\end{array}
$$

To translate $\mathbb{O}\textsc{rn}$ to $\mathbb{F}\textsc{ref}$, i.e., datatype declarations to refinements, a naive way is to use the composite functor

$$
\mathbb{O}\textsc{rn} \xrightarrow{\textsc{Ind}} \mathbb{F}\textsc{am} \xrightarrow{\text{F}\textsc{Ref}\text{C}} \mathbb{F}\textsc{ref}
$$

The resulting refinements would then use the canonical promotion predicates. However, the whole point of incorporating $\mathbb{O}$RN in the framework is that we can construct an alternative functor RSEM directly from $\mathbb{O}$RN to $\mathbb{F}$REF. The functor RSEM is extensionally equal to the above composite functor, but intensionally very different. While its object part still takes the least fixed point of a description, its morphism part is the refinement semantics of ornaments given in **??**, whose promotion predicates are the optimised predicates and have a more efficient representation.

$$\begin{aligned}
&\textsc{RSem} \;:\; \mathsf{Functor}\ \mathbb{O}\textsc{rn}\ \mathbb{F}\textsc{am} \\
&\textsc{RSem} \;=\; \textbf{record}\ \{\ \mathit{object} \;=\; \lambda\,(I\,,D) \mapsto I\,,\mu\,D \\
&\qquad\qquad\qquad\quad ;\ \mathit{morphism} \;=\; \lambda\,(e\,,O) \mapsto e\,,\mathit{RSem}\ O \\
&\qquad\qquad\qquad\quad ;\ \text{proofs of laws}\ \}
\end{aligned}$$

Categorical isomorphisms

So far switching to the categorical language offers no obvious benefits.

Define the type of isomorphisms between two objects X and Y in C as

$$\begin{aligned}
&\textbf{record}\ \mathsf{Iso}\ C\ X\ Y \;:\; \mathsf{Set}\ _\ \textbf{where} \\
&\quad \textbf{field} \\
&\qquad \mathit{to} \;:\; X \Rightarrow Y \\
&\qquad \mathit{from} \;:\; Y \Rightarrow X \\
&\qquad \mathit{from\text{-}to\text{-}inverse} \;:\; \mathit{from} \cdot \mathit{to} \;\approx\; \mathit{id} \\
&\qquad \mathit{to\text{-}from\text{-}inverse} \;:\; \mathit{to} \cdot \mathit{from} \;\approx\; \mathit{id}
\end{aligned}$$

(The relation $\_\cong\_$ is formally defined as $\mathsf{Iso}\ \mathbb{F}\textsc{un}$.)

functors preserve isomorphisms (a quick demonstration of preorder reasoning); TBC

4.2 Pullback properties of parallel composition

One of the great advantages of category theory is the ability to formulate the idea of **universal constructions** generically and concisely, which we will use to give parallel composition a useful macroscopic characterisation. An intuitive way to understand the idea of a universal construction is to think of it as a "best" solution to some specification. More precisely, the specification is represented as a category whose objects are all possible solutions and whose morphisms are evidence of how the solutions "compare" with each other, and a "best" solution is a **terminal object** in this category, meaning that it is "evidently better" than all objects in the category. For the actual definition: an object in a category C is **terminal** when it satisfies the **universal property** that for every object X' there is a unique morphism from X' to X, i.e., the setoid *Morphism* X' X has a unique inhabitant: name scoping

$$
\begin{aligned}
&\mathit{Terminal}\ C \ :\ \mathit{Object} \to \mathsf{Set}\ \_\\
&\mathit{Terminal}\ C\ X \ =\ (X' \ :\ \mathit{Object}) \to \mathit{Singleton}\ (\mathit{Morphism}\ X'\ X)
\end{aligned}
$$

where *Singleton* is defined by

$$
\begin{aligned}
&\mathit{Singleton}\ :\ (S\ :\ \mathsf{Setoid}) \to \mathsf{Set}\ \_\\
&\mathit{Singleton}\ S\ =\ \mathsf{Setoid}.\mathit{Carrier}\ S \times ((x\ y\ :\ \mathsf{Setoid}.\mathit{Carrier}\ S) \to x \approx_S y)
\end{aligned}
$$

The uniqueness condition ensures that terminal objects are unique up to (a unique) isomorphism — that is, if two objects are both terminal in C, then there is an isomorphism between them:

$$
\begin{aligned}
&\mathit{terminal\text{-}iso}\ C\ :\ (X\ Y\ :\ \mathit{Object}) \to \mathit{Terminal}\ C\ X \to \mathit{Terminal}\ C\ Y \to \mathsf{Iso}\ C\ X\ Y\\
&\mathit{terminal\text{-}iso}\ C\ X\ Y\ tX\ tY\ =\\
&\quad \mathbf{let}\, f\ :\ X \Rightarrow Y\\
&\qquad f\ =\ \mathit{outl}\ (tY\ X)\\
&\qquad g\ :\ Y \Rightarrow X\\
&\qquad g\ =\ \mathit{outl}\ (tX\ Y)\\
&\quad \mathbf{in}\ \ \mathbf{record}\ \{\ \mathit{to}\ =\ f\\
&\qquad\qquad\qquad ;\ \mathit{from}\ =\ g\\
&\qquad\qquad\qquad ;\ \mathit{from\text{-}to\text{-}inverse}\ =\ \mathit{outr}\ (tX\ X)\ (g \cdot f)\ \mathit{id}
\end{aligned}
$$

$$; \ \textit{to-from-inverse} \ = \ \textit{outr} \ (tY \ Y) \ (f \cdot g) \ \textit{id} \ \}$$

Thus, to prove that two constructions are isomorphic, one way would be to prove that they are universal in the same sense, i.e., they are both terminal objects in the same category. This is the main method we use to construct the ornamental conversion isomorphisms in Section 4.3.1 and the modularity isomorphisms in Section 4.3.2, both involving parallel composition. The goal of the rest of this section, then, is to find suitable universal properties that characterise parallel composition, preparing for Sections 4.3.1 and 4.3.2.

As said earlier, parallel composition computes the least informative ornament that contains the information of two compatible ornaments, and this is exactly a categorical **product**. Below we construct the definition of categorical products step by step. Let C be a category and L, R two objects in C. A **span** over L and R is defined by

record $\mathsf{Span}\ C\ L\ R\ :\ \mathsf{Set}\ \_$ **where**
  constructor span
  field
    $M\ :\ \mathit{Object}$
    $l\ \ :\ M \Rightarrow L$
    $r\ \ :\ M \Rightarrow R$

or diagrammatically:

$$L \xleftarrow{\ l\ } M \xrightarrow{\ r\ } R$$

If we interpret a morphism $X \Rightarrow Y$ as evidence that X is more informative than Y, then a span over L and R is essentially an object which is more informative than both L and R. Spans over the same objects can be "compared": define a morphism between two spans by

record $\mathsf{SpanMorphism}\ C\ L\ R\ (s\ s'\ :\ \mathsf{Span}\ C\ L\ R)\ :\ \mathsf{Set}\ \_$ **where**
  constructor $\mathsf{spanMorphism}$
  field
    $m\ :\ \mathsf{Span}.M\ s \Rightarrow \mathsf{Span}.M\ s'$
    $\textit{triangle-l}\ :\ \mathsf{Span}.l\ s' \cdot m\ \approx\ \mathsf{Span}.l\ s$
    $\textit{triangle-r}\ :\ \mathsf{Span}.r\ s' \cdot m\ \approx\ \mathsf{Span}.r\ s$

or diagrammatically (abbreviating $\mathsf{Span}.l\ s'$ to l' and so forth):

$$\begin{tikzcd}
& M \arrow[dl, "l"'] \arrow[dr, "r"] \arrow[dd, "m"] & \\
L & & R \\
& M' \arrow[ul, "l'"] \arrow[ur, "r'"'] &
\end{tikzcd}$$

where the two triangles are required to commute. Thus a span s is more informative than another span s' when $\mathsf{Span}.M\ s$ is more informative than $\mathsf{Span}.M\ s'$ and the morphisms factorise appropriately. We can then form a category of spans over L and R:

diagram commutativity not yet defined

morphism equivalence and proof irrelevance

$$\begin{aligned}
&\mathit{SpanCategory}\ C\ L\ R\ :\ \mathsf{Category}\\
&\mathit{SpanCategory}\ C\ L\ R\ =\ \textbf{record}\\
&\quad\{\ \mathit{Object}\quad =\ \mathsf{Span}\ C\ L\ R\\
&\quad;\ \mathit{Morphism}\ =\\
&\qquad \lambda\, s\ s'\ \mapsto\ \textbf{record}\\
&\qquad\quad \{\ \mathit{Carrier}\ =\ \mathsf{SpanMorphism}\ C\ L\ R\ s\ s'\\
&\qquad\quad ;\ _{\approx}_\ =\ \lambda\, f\ g\ \mapsto\ \mathsf{SpanMorphism}.m\ f\ \approx\ \mathsf{SpanMorphism}.m\ g\\
&\qquad\quad ;\ \text{proofs of laws}\ \}\\
&\quad;\ \text{proofs of laws}\ \}
\end{aligned}$$

and a product of L and R is a terminal object in this category:

$$\begin{aligned}
&\mathit{Product}\ C\ L\ R\ :\ \mathsf{Span}\ C\ L\ R \to \mathsf{Set}\ _\\
&\mathit{Product}\ C\ L\ R\ =\ \mathit{Terminal}\ (\mathit{SpanCategory}\ C\ L\ R)
\end{aligned}$$

In particular, a product of L and R contains the least informative object that is more informative than both L and R.

product diagram; "morphism relevance"?

We thus aim to characterise parallel composition as a product of two compatible ornaments. This means that ornaments should be the objects of some category, but so far we only know that ornaments are morphisms of the category Orn. We are thus directed to construct a category whose objects are morphisms in an ambient category C, so when we use Orn as the ambient category, parallel composition can be characterised as a product in the derived category. Such a category is in general a **comma category** [Mac Lane, 1998, § II.6], whose objects are morphisms with arbitrary source and target objects, but here we should restrict ourselves to a special case called a **slice category**, since we seek to form products of only compatible ornaments (whose less informative end coincide) rather than arbitrary ones. A slice category is parametrised with an ambient category C and an object B in C, and has

- objects: all the morphisms in C with target B,

  record Slice C B : Set _ **where**
    constructor slice
    field
      T : $Object$
      s : $T \Rightarrow B$

  and

- morphisms: mediating morphisms giving rise to commutative triangles,

  record SliceMorphism C B $(s\ s'$: Slice C $B)$: Set _ **where**
    constructor sliceMorphism
    field
      m : Slice.T $s \Rightarrow$ Slice.T s'
      $triangle$: Slice.s $s' \cdot m \approx$ Slice.s s

or diagrammatically:

$$\text{objects}\quad \begin{tikzcd} T \arrow[d, "s"'] \\ B \end{tikzcd} \qquad \text{and} \qquad \text{morphisms}\quad \begin{tikzcd} T \arrow[rr, "m"] \arrow[dr, "s"'] & & T' \arrow[dl, "s'"] \\ & B & \end{tikzcd}$$

The definitions above are assembled into the definition of slice categories in much the same way as span categories:

$SliceCategory$ C B : Category
$SliceCategory$ C B = **record**
  { $Object$ = Slice C B
  ; $Morphism$ =
      $\lambda\, s\ s' \mapsto$ **record**

```
      { Carrier = SliceMorphism C B s s′
      ; _≈_ = λ f g ↦ SliceMorphism.m f ≈ SliceMorphism.m g
      ; proofs of laws }
  ; proofs of laws }
```

Objects in a slice category are thus morphisms with a common target, and when the ambient category is ORN, they are exactly the compatible ornaments that can be composed in parallel.

We have arrived at the characterisation of parallel composition as a product in a slice category on top of ORN. The composite term "product in a slice category" has become a multi-layered concept and can be confusing; to facilitate comprehension, we give several new definitions that can sometimes deliver better intuition. Let C be an ambient category and X an object in C. We refer to spans over two slices f, g : Slice C X alternatively as **squares** over f and g:

```
Square C f g : Set _
Square C f g = Span (SliceCategory C X) f g
```

since diagrammatically a square looks like

$$\begin{tikzcd} & W \arrow[dl, "l"'] \arrow[dr, "r"] \arrow[dd] & \\ Y \arrow[dr, "f"'] & & Z \arrow[dl, "g"] \\ & X & \end{tikzcd} \qquad \text{which is the same as} \qquad \begin{tikzcd} Y \arrow[d, "f"] & W \arrow[l, "l"'] \arrow[r, "r"] \arrow[d] & Z \arrow[d, "g"'] \\ X \arrow[r, phantom, "="] & X \arrow[r, phantom, "="] & X \end{tikzcd}$$

In a square q, we will refer to the object Slice.T (Span.M q), i.e., the node W in the diagrams above, as the **vertex** of q:

```
vertex : Square C f g → Object
vertex = Slice.T ∘ Span.M
```

A product of f and g is alternatively referred to as a **pullback** of f and g; that is, it is a square over f and g satisfying

```
Pullback C f g : Square C f g → Set _
Pullback C f g = Product (SliceCategory C X) f g
```

Equivalently, if we define the **square category** over f and g as

$$\begin{aligned}&\mathit{SquareCategory}\ C\ f\ g\ :\ \mathsf{Category}\\&\mathit{SquareCategory}\ C\ f\ g\ =\ \mathit{SpanCategory}\ (\mathit{SliceCategory}\ C\ X)\ f\ g\end{aligned}$$

then a pullback of f and g is a terminal object in the square category over f and g — indeed, $\mathit{Product}\ (\mathit{SliceCategory}\ C\ X)\ f\ g$ is definitionally equal to $\mathit{Terminal}\ (\mathit{SquareCategory}\ C\ f\ g)$. This means that, by *terminal-iso*, there is an isomorphism between any two pullbacks p and q of the same slices f and g:

diagram of pullback

$$\mathsf{Iso}\ (\mathit{SquareCategory}\ C\ f\ g)\ p\ q$$

Subsequently, since there is a forgetful functor from $\mathit{SquareCategory}\ C\ f\ g$ to C whose object part is *vertex*, and functors preserve isomorphisms, we also have an isomorphism

$$\mathsf{Iso}\ C\ (\mathit{vertex}\ p)\ (\mathit{vertex}\ q) \tag{4.1}$$

which is what we actually use in Sections 4.3.1 and 4.3.2.

pullback diagram; pullback preservation

We are now ready to state precisely the pullback properties for parallel composition that we make use of later. We could attempt to establish that, for any two ornaments $O\ :\ \mathsf{Orn}\ e\ D\ E$ and $P\ :\ \mathsf{Orn}\ f\ D\ F$ where $D\ :\ \mathsf{Desc}\ I$, $E\ :\ \mathsf{Desc}\ J$, and $F\ :\ \mathsf{Desc}\ K$, the following square in $\mathbb{O}\mathrm{RN}$ is a pullback:

$$\begin{tikzcd}
e \bowtie f\ ,\ \lfloor O \otimes P \rfloor \arrow[r, "\mathit{outr}_{\bowtie}\ ,\ \mathit{diffOrn\text{-}r}\ O\ P"] \arrow[d, "\mathit{outl}_{\bowtie}\ ,\ \mathit{diffOrn\text{-}l}\ O\ P"'] \arrow[dr, "\mathit{pull},\ \lceil O \otimes P \rceil" description] & K\ ,\ F \arrow[d, "f\ ,\ P"] \\
J\ ,\ E \arrow[r, "e\ ,\ O"'] & I\ ,\ D
\end{tikzcd} \tag{4.2}$$

The AGDA term for this square is

$$\begin{aligned}&\mathit{pc\text{-}square}\ :\ \mathsf{Square}\ \mathbb{O}\mathrm{RN}\ (\mathsf{slice}\ (J\ ,\ E)\ (e\ ,\ O))\ (\mathsf{slice}\ (K\ ,\ F)\ (f\ ,\ P))\\&\mathit{pc\text{-}square}\ =\ \mathsf{span}\ (\mathsf{slice}\ (e \bowtie f\ ,\ \lfloor O \otimes P \rfloor)\ (\mathit{pull},\ \lceil O \otimes P \rceil))\\&\qquad\qquad (\mathsf{sliceMorphism}\ (\mathit{outl}_{\bowtie}\ ,\ \mathit{diffOrn\text{-}l}\ O\ P)\ \{\ \}_0)\\&\qquad\qquad (\mathsf{sliceMorphism}\ (\mathit{outr}_{\bowtie}\ ,\ \mathit{diffOrn\text{-}r}\ O\ P)\ \{\ \}_1)\end{aligned}$$

where Goal 0 has type $\mathit{OrnEq}\ (O \odot \mathit{diffOrn\text{-}l}\ O\ P)\ \lceil O \otimes P \rceil$ and Goal 1 has type

OrnEq $(P \odot \mathit{diffOrn\text{-}r}\ O\ P)\ \lceil O \otimes P \rceil$, both of which can be discharged. Comparing the commutative diagram (4.2) and the AGDA term *pc-square*, it should be obvious how concise the categorical language can be — the commutative diagram expresses the structure of the AGDA term in a clean and visually intuitive way. Since terms like *pc-square* can be reconstructed from commutative diagrams and the categorical definitions, from now on we will present commutative diagrams as representations of the corresponding AGDA terms and omit the latter. The pullback property of (4.2) is not too useful by itself, though: $\mathbb{O}$RN is a quite restricted category, so a universal property established in $\mathbb{O}$RN has limited applicability. Instead, we are more interested in the pullback property of the image of (4.2) under IND in $\mathbb{F}$AM:

$$\begin{tikzcd}
e \bowtie f\ ,\ \mu\ \lfloor O \otimes P \rfloor \arrow[r, "\mathit{outr}_{\bowtie}\ ,\ \mathit{forget}\ (\mathit{diffOrn\text{-}r}\ O\ P)"] \arrow[d, "\mathit{outl}_{\bowtie}\ ,\ \mathit{forget}\ (\mathit{diffOrn\text{-}l}\ O\ P)"'] \arrow[dr, "\mathit{pull}, \mathit{forget}\ \lceil O \otimes P \rceil" description] \arrow[dr, phantom, very near start, "\lrcorner"] & K\ ,\ \mu\ F \arrow[d, "f\ ,\ \mathit{forget}\ P"] \\
J\ ,\ \mu\ E \arrow[r, "e\ ,\ \mathit{forget}\ O"'] & I\ ,\ \mu\ D
\end{tikzcd} \tag{4.3}$$

We assert that the above square is a pullback by marking its vertex with "$\lrcorner$". The proof of its universal property boils down to, very roughly speaking, datatype-generic construction of an inverse to

$$\mathit{forget}\ (\mathit{diffOrn\text{-}l}\ O\ P) \vartriangle \mathit{forget}\ (\mathit{diffOrn\text{-}r}\ O\ P)$$

which involves tricky manipulation of equality proofs but is achievable. After the pullback square (4.3) is established in $\mathbb{F}$AM, since the functor COM is pullback-preserving, we also get a pullback square in $\mathbb{F}$UN:

$$\begin{tikzcd}
\Sigma\ (e \bowtie f)\ (\mu\ \lfloor O \otimes P \rfloor) \arrow[r, "\mathit{outr}_{\bowtie} * \mathit{forget}\ (\mathit{diffOrn\text{-}r}\ O\ P)"] \arrow[d, "\mathit{outl}_{\bowtie} * \mathit{forget}\ (\mathit{diffOrn\text{-}l}\ O\ P)"'] \arrow[dr, "\mathit{pull} * \mathit{forget}\ \lceil O \otimes P \rceil" description] \arrow[dr, phantom, very near start, "\lrcorner"] & \Sigma\ K\ (\mu\ F) \arrow[d, "f * \mathit{forget}\ P"] \\
\Sigma\ J\ (\mu\ E) \arrow[r, "e * \mathit{forget}\ O"'] & \Sigma\ I\ (\mu\ D)
\end{tikzcd} \tag{4.4}$$

mention commutativity, associativity; should probably refactor the pullback square in $\mathbb{F}$AM into the pullback square in ORN and pullback preservation of IND

4.3 Consequences

4.3.1 The ornamental conversion isomorphisms

We restate the ornamental conversion isomorphisms as follows: for any ornament $O : \mathsf{Orn}\ e\ D\ E$ where $D : \mathsf{Desc}\ I$ and $E : \mathsf{Desc}\ J$, we have

$$\mu\ E\ j \ \cong\ \Sigma[\, x : \mu\ D\ (e\ j)\,]\ \mathsf{OptP}\ O\ (\mathsf{ok}\ j)\ x$$

for all $j : J$. Since the optimised predicates $\mathsf{OptP}\ O$ are defined by parallel composition of O and the singleton ornament $S = \mathit{singletonOD}\ D$, the isomorphism expands to

$$\mu\ E\ j \ \cong\ \Sigma[\, x : \mu\ D\ (e\ j)\,]\ \mu\ \lfloor O \otimes \lceil S \rceil \rfloor\ (\mathsf{ok}\ j\ ,\ \mathsf{ok}\ (e\ j\ ,\ x)) \tag{4.5}$$

How do we derive this from the pullback properties for parallel composition? It turns out that the pullback property in $\mathbb{F}$UN (4.4) can help.

- First, observe that we have the following pullback square:

$$\begin{tikzcd}
\Sigma\ J\ (\mu\ E) \arrow[rr, "(e * \mathit{forget}\ O) \vartriangle (\mathit{singleton} \circ \mathit{forget}\ O \circ \mathit{outr})"] \arrow[d, "\mathit{id}"'] \arrow[drr, "e * \mathit{forget}\ O"] \arrow[dr, phantom, "\lrcorner", very near start] & & \Sigma\ (\Sigma\ I\ (\mu\ D))\ (\mu\ \lfloor S \rfloor) \arrow[d, "\mathit{outl} * \mathit{forget}\ \lceil S \rceil"] \\
\Sigma\ J\ (\mu\ E) \arrow[rr, "e * \mathit{forget}\ O"'] & & \Sigma\ I\ (\mu\ D)
\end{tikzcd} \tag{4.6}$$

  Viewing pullbacks as products of slices, since a singleton ornament does not add information to a datatype, the vertical slice on the right-hand side

  $$s \ =\ \mathsf{slice}\ (\Sigma\ (\Sigma\ I\ (\mu\ D))\ (\mu\ \lfloor S \rfloor))\ (\mathit{outl} * \mathit{forget}\ \lceil S \rceil)$$

  behaves like a "multiplicative unit": any (compatible) slice s' alone gives rise to a product of s and s'. As a consequence, we have the bottom-left type

$\Sigma\ J\ (\mu\ E)$ as the vertex of the pullback. This pullback square is over the same slices as the pullback square (4.4) with P substituted by $\lceil S \rceil$, so by (4.1) we obtain an isomorphism

$$\Sigma\ J\ (\mu\ E)\ \cong\ \Sigma\ (e \bowtie \mathit{outl})\ (\mu\ \lfloor O \otimes \lceil S \rceil \rfloor) \tag{4.7}$$

- To get from (4.7) to (4.5), we need to look more closely into the construction of (4.7). The right-to-left direction of (4.7) is obtained by applying the universal property of (4.6) to the square (4.4) (with P substituted by $\lceil S \rceil$), so it is the unique mediating morphism m that makes the following diagram commute:

$$\begin{tikzcd}
& \Sigma\ (e \bowtie \mathit{outl})\ (\mu\ \lfloor O \otimes \lceil S \rceil \rfloor) \arrow[dl, "\mathit{outl}_{\bowtie} * \mathit{forget}\ (\mathit{diffOrn\text{-}l}\ O\ P)"'] \arrow[dr, "\mathit{outr}_{\bowtie} * \mathit{forget}\ (\mathit{diffOrn\text{-}r}\ O\ P)"] \arrow[dd, dashed, "m"'] & \\
\Sigma\ J\ (\mu\ E) & & \Sigma\ (\Sigma\ I\ (\mu\ D))\ (\mu\ \lfloor S \rfloor) \\
& \Sigma\ J\ (\mu\ E) \arrow[ul, "\mathit{id}"] \arrow[ur, "(e * \mathit{forget}\ O) \vartriangle (\mathit{singleton} \circ \mathit{forget}\ O \circ \mathit{outr})"'] &
\end{tikzcd}$$

  From the left commuting triangle, we see that, extensionally, the morphism m is just $\mathit{outl}_{\bowtie} * \mathit{forget}\ (\mathit{diffOrn\text{-}l}\ O\ P)$.

- This leads us to the following general lemma: if there is an isomorphism

  $$\Sigma\ K\ X\ \cong\ \Sigma\ L\ Y$$

  whose right-to-left direction is extensionally equal to some $f * g$, then we have

  $$X\ k\ \cong\ \Sigma[\, l : f^{-1}\ k\,]\ \ Y\ (\mathit{und}\ l)$$

  for all $k : K$. For a justification: fixing $k : K$, an element of the form $(k\ ,\ x) : \Sigma\ K\ X$ must correspond, under the given isomorphism, to some element $(l\ ,\ y) : \Sigma\ L\ Y$ such that $f\ l\ \equiv\ k$, so the set $X\ k$ corresponds to exactly the sum of the sets $Y\ l$ such that $f\ l\ \equiv\ k$.

- Specialising the lemma above for (4.7), we get

  $$\mu\ E\ j\ \cong\ \Sigma[\, \mathit{jix} : \mathit{outl}_{\bowtie}{}^{-1}\ j\,]\ \ \mu\ \lfloor O \otimes \lceil S \rceil \rfloor\ (\mathit{und}\ \mathit{jix}) \tag{4.8}$$

for all $j : J$. Finally, observe that a canonical element of type $\mathit{outl}_{\bowtie}{}^{-1}\ j$ must be of the form $\mathsf{ok}\ (\mathsf{ok}\ j\ ,\ \mathsf{ok}\ (e\ j\ ,\ x))$ for some $x : \mu\ D\ (e\ j)$, so we perform a change of variables for the summation, turning the right-hand side of (4.8) into

$$\Sigma[\,x : \mu\ D\ (e\ j)\,]\ \ \mu\ \lfloor O \otimes \lceil S \rceil \rfloor\ (\mathsf{ok}\ j\ ,\ \mathsf{ok}\ (e\ j\ ,\ x))$$

and arriving at (4.5).

Formalisation detail. There is a twist when it comes to formalisation of the proof in Agda, however, due to Agda's intensionality: It is possible to formalise the lemma and the change of variables individually and chain them together, but the resulting isomorphisms would have a very complicated definition due to suspended type casts. If we use them to construct the refinement family in the morphism part of RSem, it would be rather difficult to prove that the morphism part of RSem preserves equivalence. We are thus forced to fuse all the above reasoning into one step to get a clean Agda definition such that RSem preserves equivalence automatically, but the idea is still essentially the same. □

4.3.2 The modularity isomorphisms

The other important family of isomorphisms we should construct from the pullback properties of parallel composition is the modularity isomorphisms, which is restated as follows: Suppose that there are descriptions $D : \mathsf{Desc}\ I$, $E : \mathsf{Desc}\ J$ and $F : \mathsf{Desc}\ K$, and ornaments $O : \mathsf{Orn}\ e\ D\ E$, and $P : \mathsf{Orn}\ f\ D\ F$. Then we have

$$\mathsf{OptP}\ \lceil O \otimes P \rceil\ (\mathsf{ok}\ (j\ ,\ k))\ x\ \cong\ \mathsf{OptP}\ O\ j\ x \times \mathsf{OptP}\ P\ k\ x$$

for all $i : I$, $j : e^{-1}\ i$, $k : f^{-1}\ i$, and $x : \mu\ D\ i$. The isomorphism expands to

$$\begin{aligned}&\mu\ \lfloor \lceil O \otimes P \rceil \otimes \lceil S \rceil \rfloor\ (\mathsf{ok}\ (j\ ,\ k)\ ,\ \mathsf{ok}\ (i\ ,\ x))\\ &\quad\cong\ \mu\ \lfloor O \otimes \lceil S \rceil \rfloor\ (j\ ,\ \mathsf{ok}\ (i\ ,\ x)) \times \mu\ \lfloor P \otimes \lceil S \rceil \rfloor\ (k\ ,\ \mathsf{ok}\ (i\ ,\ x))\end{aligned} \tag{4.9}$$

where again $S = \mathit{singletonOD}\ D$. A quick observation is that they are componentwise isomorphisms between the two families of sets

$$M \;=\; \mu\ \lfloor \lceil O \otimes P \rceil \otimes \lceil S \rceil \rfloor$$

and

$$\begin{aligned} N \;=\; & \lambda\,(\mathsf{ok}\ (j\ ,k)\ ,\mathsf{ok}\ (i\ ,x))\ \mapsto \\ & \quad \mu\ \lfloor O \otimes \lceil S \rceil \rfloor\ (j\ ,\mathsf{ok}\ (i\ ,x)) \times \mu\ \lfloor P \otimes \lceil S \rceil \rfloor\ (k\ ,\mathsf{ok}\ (i\ ,x)) \end{aligned}$$

both indexed by $\mathit{pull} \bowtie \mathit{outl}$ where pull has type $e \bowtie f \to I$ and outl has type $\Sigma\ I\ X \to I$. This is just an isomorphism in $\mathbb{F}$AM between $(\mathit{pull} \bowtie \mathit{outl}\ ,M)$ and $(\mathit{pull} \bowtie \mathit{outl}\ ,N)$ whose index part (i.e., the isomorphism obtained under the functor FAMF) is identity. Thus we seek to prove that both $(\mathit{pull} \bowtie \mathit{outl}\ ,M)$ and $(\mathit{pull} \bowtie \mathit{outl}\ ,N)$ are vertices of pullbacks of the same slices.

- We look at $(\mathit{pull} \bowtie \mathit{outl}\ ,N)$ first. For fixed i, j, k, and x, the set

  $$N\ (\mathsf{ok}\ (j\ ,k)\ ,\mathsf{ok}\ (i\ ,x))$$

  along with the cartesian projections is a product, which trivially extends to a pullback since there is a forgetful function from each of the two component sets to the **singleton** set $\mu\ \lfloor S \rfloor\ (i\ ,x)$, as shown in the following diagram:

  $$\begin{tikzcd} N\ (\mathsf{ok}\ (j\ ,k)\ ,\mathsf{ok}\ (i\ ,x)) \arrow[r, "\mathit{outr}"] \arrow[d, "\mathit{outl}"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \mu\ \lfloor P \otimes \lceil S \rceil \rfloor\ (k\ ,\mathsf{ok}\ (i\ ,x)) \arrow[d, "\mathit{forget}\ (\mathit{diffOrn\text{-}r}\ P\ \lceil S \rceil)"] \\ \mu\ \lfloor O \otimes \lceil S \rceil \rfloor\ (j\ ,\mathsf{ok}\ (i\ ,x)) \arrow[r, "\mathit{forget}\ (\mathit{diffOrn\text{-}r}\ O\ \lceil S \rceil)"'] & \mu\ \lfloor S \rfloor\ (i\ ,x) \end{tikzcd}$$

  Note that this pullback square is possible because of the common x in the indices of the two component sets — otherwise they cannot project to the same singleton set. Collecting all such pullback squares together, we get the following pullback square in $\mathbb{F}$AM:

  $$\begin{tikzcd} \mathit{pull} \bowtie \mathit{outl}\ ,N \arrow[r, "\_\ ,\mathit{outr}"] \arrow[d, "\_\ ,\mathit{outl}"'] \arrow[dr, phantom, very near start, "\lrcorner"] & f \bowtie \mathit{outl}\ ,\mu\ \lfloor P \otimes \lceil S \rceil \rfloor \arrow[d, "\mathit{outr}_{\bowtie}\ ,\mathit{forget}\ (\mathit{diffOrn\text{-}r}\ P\ \lceil S \rceil)"] \\ e \bowtie \mathit{outl}\ ,\mu\ \lfloor O \otimes \lceil S \rceil \rfloor \arrow[r, "\mathit{outr}_{\bowtie}\ ,\mathit{forget}\ (\mathit{diffOrn\text{-}r}\ O\ \lceil S \rceil)"'] & \Sigma\ I\ (\mu\ D)\ ,\mu\ \lfloor S \rfloor \end{tikzcd} \tag{4.10}$$

- Next we prove that $(\mathit{pull} \bowtie \mathit{outl}\ ,\ M)$ is also the vertex of a pullback of the same slices as (4.10). This second pullback arises as a consequence of the following lemma (illustrated in the diagram below): In any category, consider the objects X, Y, their product $X \Leftarrow X \boxtimes Y \Rightarrow Y$, and products of each of the three objects X, Y, and $X \boxtimes Y$ with an object Z. (All the projections are shown as solid arrows in the diagram.) Then $(X \boxtimes Y) \boxtimes Z$ is the vertex of a pullback of the two projections $X \boxtimes Z \Rightarrow Z$ and $Y \boxtimes Z \Rightarrow Z$.

$$\begin{tikzcd}
X & & X \boxtimes Y \arrow[ll] \arrow[rr] & & Y \\
& & (X \boxtimes Y) \boxtimes Z \arrow[u] \arrow[dl, dashed] \arrow[dr, dashed] \arrow[dd] & & \\
& X \boxtimes Z \arrow[uul] \arrow[dr] & & Y \boxtimes Z \arrow[uur] \arrow[dl] & \\
& & Z & &
\end{tikzcd}$$

We again intend to view a pullback as a product of slices, and instantiate the lemma in *SliceCategory* $\mathbb{F}$AM $(I\ ,\ \mu\ D)$, substituting all the objects by slices consisting of relevant ornamental forgetful functions in (4.9). The substitutions are as follows:

$$\begin{aligned}
X &\mapsto \mathsf{slice}\ _\ (_\ , \mathit{forget}\ O) \\
Y &\mapsto \mathsf{slice}\ _\ (_\ , \mathit{forget}\ P) \\
X \boxtimes Y &\mapsto \mathsf{slice}\ _\ (_\ , \mathit{forget}\ \lceil O \otimes P \rceil) \\
Z &\mapsto \mathsf{slice}\ _\ (_\ , \mathit{forget}\ \lceil S \rceil) \\
X \boxtimes Z &\mapsto \mathsf{slice}\ _\ (_\ , \mathit{forget}\ \lceil O \otimes \lceil S \rceil \rceil) \\
Y \boxtimes Z &\mapsto \mathsf{slice}\ _\ (_\ , \mathit{forget}\ \lceil P \otimes \lceil S \rceil \rceil) \\
(X \boxtimes Y) \boxtimes Z &\mapsto \mathsf{slice}\ _\ (_\ , \mathit{forget}\ \lceil \lceil O \otimes P \rceil \otimes \lceil S \rceil \rceil)
\end{aligned}$$

where $X \boxtimes Y$, $X \boxtimes Z$, $Y \boxtimes Z$, and $(X \boxtimes Y) \boxtimes Z$ indeed give rise to products in *SliceCategory* $\mathbb{F}$AM $(I\ ,\ \mu\ D)$, i.e., pullbacks in $\mathbb{F}$AM, by instantiating (4.3). What we get out of this instantiation of the lemma is a pullback in *SliceCategory* $\mathbb{F}$AM $(I\ ,\ \mu\ D)$ rather than $\mathbb{F}$AM. This is easy to fix, since there is a forgetful functor from any *SliceCategory* C B to C whose object part is $\mathsf{Slice}.T$, and it is pullback-preserving. We thus get a pullback in $\mathbb{F}$AM of the same slices as (4.10) whose vertex is $(\mathit{pull} \bowtie \mathit{outl}\ , M)$.

Having the two pullbacks, by (4.1) we get an isomorphism in $\mathbb{F}\textsc{am}$ between $(\mathit{pull} \bowtie \mathit{outl}\ , M)$ and $(\mathit{pull} \bowtie \mathit{outl}\ , N)$, whose index part can be shown to be identity, so there are componentwise isomorphisms between M and N in $\mathbb{F}\textsc{un}$, arriving at (4.9).

4.4 Discussion

elimination of arbitrariness of type-theoretic constructions; functor-level abstraction; compare with purely categorical approach

Bibliography

Gilles BARTHE, Venanzio CAPRETTA, and Olivier PONS [2003]. Setoids in type theory. *Journal of Functional Programming*, 13(2):261–293. doi: 10.1017/S0956796802004501. ↰ page 4

Robert HARPER and Robert POLLACK [1991]. Type checking with universes. *Theoretical Computer Science*, 89(1):107–136. doi: 10.1016/0304-3975(90)90108-T. ↰ page 4

Saunders MAC LANE [1998]. *Categories for the Working Mathematician*. Springer-Verlag, second edition. ↰ pages 2 and 16

Conor MCBRIDE [1999]. *Dependently Typed Functional Programs and their Proofs*. Ph.D. thesis, University of Edinburgh. ↰ page 7

Shin-Cheng MU, Hsiang-Shang KO, and Patrik JANSSON [2009]. Algebra of Programming in Agda: Dependent types for relational program derivation. *Journal of Functional Programming*, 19(5):545–579. doi: 10.1017/S0956796809007345. ↰ page 6

Todo list