A category theory primer

Ralf Hinze
Computing Laboratory, University of Oxford
Wolfson Building, Parks Road, Oxford, OX1 3QD, England
ralf.hinze@comlab.ox.ac.uk
http://www.comlab.ox.ac.uk/ralf.hinze/

1 Categories, functors and natural transformations

Category theory is a game with objects and arrows between objects. We let \( C, D \) etc range over categories.

A category is often identified with its class of objects. For instance, we say that \( \text{Set} \) is the category of sets. In the same spirit, we write \( A \in C \) to express that \( A \) is an object of \( C \). We let \( A, B \) etc range over objects.

However, equally, if not more important are the arrows of a category. So, \( \text{Set} \) is really the category of sets and total functions. (There is also \( \text{Rel} \), the category of sets and relations.) If the objects have additional structure (monoids, groups etc.) then the arrows are typically structure-preserving maps. For every pair of objects \( A, B \in C \) there is a class of arrows from \( A \) to \( B \), denoted \( C(A, B) \). If \( C \) is obvious from the context, we abbreviate \( f \in C(A, B) \) by \( f : A \to B \). We will also loosely speak of \( A \to B \) as the type of \( f \). We let \( f, g \) etc range over arrows.

For every object \( A \in C \) there is an arrow \( \text{id}_A \in C(A, A) \), called the identity.

Two arrows can be composed if their types match: if \( f \in C(A, B) \) and \( g \in C(B, C) \), then \( g \cdot f \in C(A, C) \). We require composition to be associative with identity as its neutral element.

Every structure comes equipped with structure-preserving maps, so do categories, where these maps are called functors. Since a category consists of two parts, objects and arrows, a functor \( F : C \to D \) consists of a mapping on objects and a mapping on arrows. It is common practise to denote both mappings by the same symbol. We will also loosely speak of \( F \)'s arrow part as a 'map'. The action on arrows has to respect the types: if \( f \in C(A, B) \), then \( Ff \in D(FA, FB) \). Furthermore, \( F \) has to preserve identity, \( F\text{id}_A = \text{id}_{FA} \), and composition \( F(g \cdot f) = Fg \cdot Ff \). The force of functoriality lies in the action on arrows and in the preservation of composition. There is an identity functor, \( \text{id}_C : C \to C \), and functors can be composed: \( (G \circ F)A = G(FA) \) and \( (G \circ F)f = G(Ff) \). This data turns small categories and functors into a category, called \( \text{Cat} \).\(^1\) We let \( F, G \) etc range over functors.

Let \( F, G : C \to D \) be two parallel functors. A natural transformation \( \alpha : F \to G \) is a collection of arrows, so that for each object \( A \in C \) there is an arrow

\(^1\) To avoid paradoxes, we have to require that the objects of \( \text{Cat} \) are small, where a category is called small if the class of objects and the class of all arrows are sets.
\( \omega A \in \mathcal{D}(F A, G A) \) such that
\[
G h \cdot \omega A' = \omega A'' \cdot F h, \tag{1}
\]
for all arrows \( h \in \mathcal{C}(A', A'') \). Given \( \omega \) and \( h \), there are essentially two ways of turning \( F A' \) things into \( G A'' \) things. The coherence condition (1) demands that they are equivalent. We also write \( \alpha : \forall A . F A \rightarrow G A \) and furthermore \( \omega : \forall A . F A \cong G A \), if \( \omega \) is a natural isomorphism. There is an identity transformation \( \text{id}_F : F \rightarrow F \) defined \( \text{id}_F A = \text{id}_{F A} \). Natural transformations can be composed: if \( \alpha : F \rightarrow G \) and \( \beta : G \rightarrow H \), then \( \beta \cdot \alpha : F \rightarrow H \) is defined \((\beta \cdot \alpha) A = \beta A \cdot \alpha A \). Thus, functors of type \( \mathcal{C} \rightarrow \mathcal{D} \) and natural transformations between them form a category, the functor category \( \mathcal{D}^\mathcal{C} \). (Functor categories are exponentials in \( \mathcal{C} \), hence the notation.) We let \( \omega, \beta \) etc range over natural transformations.

## 2 Constructions on categories

New categories from old.

Let \( \mathcal{C} \) be a category. The opposite category \( \mathcal{C}^{\text{op}} \) has the same objects as \( \mathcal{C} \), arrows and composition, however, are flipped: \( f \in \mathcal{C}^{\text{op}}(A, B) \) if \( f \in \mathcal{C}(B, A) \), and \( g \cdot f \in \mathcal{C}^{\text{op}}(A, C) \) if \( f \cdot g \in \mathcal{C}(C, A) \). A functor of type \( \mathcal{C}^{\text{op}} \rightarrow \mathcal{D} \) or \( \mathcal{C} \rightarrow \mathcal{D}^{\text{op}} \) is sometimes called a contravariant functor from \( \mathcal{C} \) to \( \mathcal{D} \), the usual kind being styled covariant. An incestuous example of a contravariant functor is \( \mathcal{C}(-, B) : \mathcal{C}^{\text{op}} \rightarrow \text{Set} \), whose action on arrows is given by \( \mathcal{C}(h, B)f = f \cdot h \).\(^2\) The functor \( \mathcal{C}(-, B) \) maps an object \( A \) to the set of arrows from \( A \) to a fixed \( B \), and it takes an arrow \( h \in \mathcal{C}(A', A) \) to a function \( \mathcal{C}(h, B) : \mathcal{C}(A', B) \rightarrow \mathcal{C}(A'', B) \). Conversely, \( \mathcal{C}(A, -) : \mathcal{C} \rightarrow \text{Set} \) is a covariant functor defined \( \mathcal{C}(A, k)f = k \cdot f \).

Let \( \mathcal{C}_1 \) and \( \mathcal{C}_2 \) be a categories. An object of the product category \( \mathcal{C}_1 \times \mathcal{C}_2 \) is a pair \( \langle A_1, A_2 \rangle \) of objects \( A_1 \in \mathcal{C}_1 \) and \( A_2 \in \mathcal{C}_2 \); an arrow of \( \langle \mathcal{C}_1 \times \mathcal{C}_2 \rangle(\langle A_1, A_2 \rangle, \langle B_1, B_2 \rangle) \) is a pair \( \langle f_1, f_2 \rangle \) of arrows \( f_1 \in \mathcal{C}_1(A_1, B_1) \) and \( f_2 \in \mathcal{C}_2(A_2, B_2) \). Identity and composition are defined component-wise:

\[
id = \langle \text{id}, \text{id} \rangle \quad \text{and} \quad \langle g_1, g_2 \rangle \cdot \langle f_1, f_2 \rangle = \langle g_1 \cdot f_1, g_2 \cdot f_2 \rangle.
\]

The projection functors \( \text{Outl} : \mathcal{C}_1 \times \mathcal{C}_2 \rightarrow \mathcal{C}_1 \) and \( \text{Outr} : \mathcal{C}_1 \times \mathcal{C}_2 \rightarrow \mathcal{C}_2 \) are given by \( \text{Outl} \langle A_1, A_2 \rangle = A_1 \), \( \text{Outl} \langle f_1, f_2 \rangle = f_1 \) and \( \text{Outr} \langle A_1, A_2 \rangle = A_2 \), \( \text{Outr} \langle f_1, f_2 \rangle = f_2 \). Product categories avoid the need for functors of several arguments. Functors from a product category are sometimes called bifunctors. An example is the hom-functor \( \mathcal{C}(-, =) : \mathcal{C}^{\text{op}} \times \mathcal{C} \rightarrow \text{Set} \), which maps a pair of objects to the set of arrows between them. Its action on arrows is given by \( \mathcal{C}(f, g)h = g \cdot h \cdot f \). The diagonal functor \( \Delta : \mathcal{C} \rightarrow \mathcal{C} \times \mathcal{C} \) is an example of a functor into a product category: it duplicates its argument \( \Delta A = \langle A, A \rangle \) and \( \Delta f = \langle f, f \rangle \).

---

\(^2\) Partial applications of mappings and operators are written using 'categorical dummies', where \( - \) marks the first and \( = \) the optional second argument.
3 Products and coproducts

Constructions in category theory are typically given using so-called universal properties. The paradigmatic example of this approach is the definition of products — in fact, this is also historically the first example. The product of two objects $B_1$ and $B_2$ consists of an object written $B_1 \times B_2$ and a pair of arrows $\text{outl} : B_1 \times B_2 \rightarrow B_1$ and $\text{outr} : B_1 \times B_2 \rightarrow B_2$. These three things have to satisfy the following universal property: for each object $A$ and for each pair of arrows $f_1 : A \rightarrow B_1$ and $f_2 : A \rightarrow B_2$, there exists an arrow $f_1 \triangle f_2 : A \rightarrow B_1 \times B_2$ (pronounced “split”) such that

$$f_1 = \text{outl} \cdot g \quad \wedge \quad f_2 = \text{outr} \cdot g \quad \iff \quad f_1 \triangle f_2 = g,$$

(2)

for all $g : A \rightarrow B_1 \times B_2$. The property states the existence of the arrow $f_1 \triangle f_2$ and furthermore that it is the unique arrow satisfying the property on the left. (It is also called the mediating arrow). The following diagram summarises the type information.

This is an example of a commuting diagram: all paths from the same source to the same target lead to the same result by composition. The dotted arrow indicates that $f_1 \triangle f_2$ is the unique arrow from $A$ to $B_1 \times B_2$ that makes the diagram commute.

A universal property such as (2) has two immediate consequences that are worth singling out. If we substitute the right-hand side into the left-hand side, we obtain the computation laws (also known as $\beta$-rules):

$$f_1 = \text{outl} \cdot (f_1 \triangle f_2); \quad \quad f_2 = \text{outr} \cdot (f_1 \triangle f_2).$$

(3) (4)

They can be seen as defining equations for the arrow $f \triangle g$.

Instantiating $g$ in (2) to the identity $\text{id}_{A \times B}$ and substituting into the right-hand side, we obtain the reflection law (also known as the simple $\eta$-rule):

$$\text{outl} \triangle \text{outr} = \text{id}_{A \times B}.$$ 

(5)

The universal property enjoys two further consequences, which we shall later identify as naturality properties. The first consequence is the fusion law that allows us to fuse a split with an arrow to form another split:

$$(f_1 \triangle f_2) \cdot h = f_1 \cdot h \triangle f_2 \cdot h,$$

(6)
for all \( h : A' \to A'' \). The law states that \( \triangle \) is natural in \( A \). For the proof we reason

\[
\begin{align*}
\quad & f_1 \cdot h \triangle f_2 \cdot h = (f_1 \triangle f_2) \cdot h \\
\iff & \quad \{ \text{universal property (2)} \} \\
\quad & f_1 \cdot h = \text{outl} \cdot (f_1 \triangle f_2) \cdot h \land f_2 \cdot h = \text{outr} \cdot (f_1 \triangle f_2) \cdot h \\
\iff & \quad \{ \text{computation (3) and (4)} \} \\
\quad & f_1 \cdot h = f_1 \cdot h \land f_2 \cdot h = f_2 \cdot h
\end{align*}
\]

The definition of products is also parametric in \( B_1 \) and \( B_2 \) — note that both objects are totally passive in the description above. We capture this property by turning \( \times \) into a functor of type \( \mathbb{C} \times \mathbb{C} \to \mathbb{C} \) (to define products in \( \mathbb{C} \) we need \( \mathbb{C} \) we define products in \( \mathbb{C} \)). The action of \( \times \) on arrows is defined

\[
\begin{align*}
\quad & f_1 \times f_2 = f_1 \cdot \text{outl} \triangle f_2 \cdot \text{outr}.
\end{align*}
\]

We postpone the proof that \( \times \) preserves identity and composition.

The functor fusion law states that we can fuse a map after a split to form another split:

\[
\begin{align*}
\quad & (k_1 \times k_2) \cdot (f_1 \triangle f_2) = k_1 \cdot f_1 \triangle k_2 \cdot f_2,
\end{align*}
\]

for all \( k_1 : B_1' \to B_1'' \) and \( k_2 : B_2' \to B_2'' \). It formalises that \( \triangle \) is natural in \( B_1 \) and \( B_2 \). The proof of (8) builds on fusion and computation.

\[
\begin{align*}
\quad & (k_1 \times k_2) \cdot (f_1 \triangle f_2) \\
= & \quad \{ \text{definition of } \times (7) \} \\
= & \quad (k_1 \cdot \text{outl} \triangle k_2 \cdot \text{outr}) \cdot (f_1 \triangle f_2) \\
= & \quad \{ \text{fusion (6)} \} \\
= & \quad k_1 \cdot \text{outl} \cdot (f_1 \triangle f_2) \triangle k_2 \cdot \text{outr} \cdot (f_1 \triangle f_2) \\
= & \quad \{ \text{computation (3) and (4)} \} \\
= & \quad k_1 \cdot f_1 \triangle k_2 \cdot f_2
\end{align*}
\]

Given these prerequisites, it is straightforward to show that \( \times \) preserves identity

\[
\begin{align*}
\quad & \text{id}_A \times \text{id}_B \\
= & \quad \{ \text{definition of } \times (7) \} \\
= & \quad \text{id}_A \cdot \text{outl} \triangle \text{id}_B \cdot \text{outr} \\
= & \quad \{ \text{identity and reflection (5)} \} \\
= & \quad \text{id}_{A \times B}
\end{align*}
\]

and composition

\[
\begin{align*}
\quad & (g_1 \times g_2) \cdot (f_1 \times f_2)
\end{align*}
\]
\( g_1 \times g_2 \cdot (f_1 \cdot \text{outl} \triangle f_2 \cdot \text{outr}) = \) \{ definition of \( \times \) (7) \} \\
\( g_1 \cdot f_1 \cdot \text{outl} \triangle g_2 \cdot f_2 \cdot \text{outr} = \) \{ functor fusion (8) \} \\
\( g_1 \cdot f_1 \times g_2 \cdot f_2. \)

The projection arrows, \( \text{outl} \) and \( \text{outr} \) are natural transformations, as well.

\[
\begin{align*}
k_1 \cdot \text{outl} &= \text{outl} \cdot (k_1 \times k_2); \\
k_2 \cdot \text{outr} &= \text{outr} \cdot (k_1 \times k_2).
\end{align*}
\]

This is a direct consequence of the computation laws.

\[
\begin{align*}
\text{outl} \cdot (k_1 \times k_2) \\
= \text{definition of } \times (7) \\
\text{outl} \cdot (k_1 \cdot \text{outl} \triangle k_2 \cdot \text{outr}) \\
= \text{computation (3)} \\
k_1 \cdot \text{outl}
\end{align*}
\]

The naturality of \( \triangle \) can be captured precisely using product categories and hom-functors.

\( (\triangle): \forall A B . (C \times C)(\Delta A, B) \rightarrow C(A, B) \)

Split takes a pair of arrows as an argument and delivers an arrow to a product \( (B \text{ is an object in a product category}) \). The fusion law captures naturality in \( A \),

\[ C(h, xB) \cdot (\triangle) = (\triangle) \cdot (C \times C)(\Delta h, B), \]
and the functor fusion law naturality in \( B \),

\[ C(A, xk) \cdot (\triangle) = (\triangle) \cdot (C \times C)(\Delta A, k). \]

(A transformation between bifunctors is natural if and only if it is natural in both arguments separately.)

The naturality of \( \text{outl} \) and \( \text{outr} \) can be captured using the projection functors \( \text{Outl} \) and \( \text{Outr} \).

\[
\begin{align*}
\text{outl} : \forall B . C(\times B, \text{Outl} B); \\
\text{outr} : \forall B . C(\times B, \text{Outr} B).
\end{align*}
\]

The naturality conditions are

\[
\begin{align*}
\text{Outl} k \cdot \text{outl} &= \text{outl} \cdot xk, \\
\text{Outr} k \cdot \text{outr} &= \text{outr} \cdot xk.
\end{align*}
\]

The import of all this is that \( \times \) is right adjoint to the diagonal functor \( \Delta \), see Sec. 6.
The construction of products nicely dualises to coproducts, which are products in the opposite category. The *coproduct* of two objects $A_1$ and $A_2$ consists of an object written $A_1 + A_2$ and a pair of arrows $\text{inl} : A_1 \to A_1 + A_2$ and $\text{inr} : A_2 \to A_1 + A_2$. These three things have to satisfy the following *universal property*: for each object $B$ and for each pair of arrows $g_1 : A_1 \to B$ and $g_2 : A_2 \to B$, there exists an arrow $g_1 \triangledown g_2 : A_1 + A_2 \to B$ (pronounced “join”) such that

\[ f = g_1 \triangledown g_2 \iff f \cdot \text{inl} = g_1 \land f \cdot \text{inr} = g_2, \tag{11} \]

for all $f : A_1 + A_2 \to B$.

\[
\begin{array}{c}
A_1 \xrightarrow{\text{inl}} A_1 + A_2 \xrightarrow{\text{inr}} A_2 \\
\downarrow g_1 \triangledown g_2 \\
\downarrow \Psi \xrightarrow{\text{inl}} B \\
\downarrow \Psi \xrightarrow{\text{inr}} B
\end{array}
\]

Like for products, the universal property implies computation, reflection, fusion and functor fusion laws. *Computation law*:

\[
(g_1 \triangledown g_2) \cdot \text{inl} = g_1; \tag{12}
\]
\[
(g_1 \triangledown g_2) \cdot \text{inr} = g_2. \tag{13}
\]

*Reflection law*:

\[ \text{id}_{A + B} = \text{inl} \triangledown \text{inr}. \tag{14} \]

*Fusion law*:

\[ k \cdot (g_1 \triangledown g_2) = k \cdot g_1 \triangledown k \cdot g_2. \tag{15} \]

The arrow part of the coproduct functor is defined

\[ g_1 + g_2 = \text{inl} \cdot g_1 \triangledown \text{inr} \cdot g_2. \tag{16} \]

*Functor fusion law*:

\[
(g_1 \triangledown g_2) \cdot (h_1 + h_2) = g_1 \cdot h_1 \triangledown g_2 \cdot h_2. \tag{17}
\]

The two fusion laws identify $\triangledown$ as a natural transformation:

\[ (\triangledown) : \forall A B. \mathbb{C}(+A, B) \to (\mathbb{C} \times \mathbb{C})(A, \Delta B). \]

Finally, the injection arrows are natural transformations, as well.

\[
(h_1 + h_2) \cdot \text{inl} = \text{inl} \cdot h_1 \tag{18}
\]
\[
(h_1 + h_2) \cdot \text{inr} = \text{inr} \cdot h_2 \tag{19}
\]

The import of all this is that $+$ is left adjoint to the diagonal functor $\Delta$. 
4 Initial and final objects

An object is called initial if for each object \( B \in \mathcal{C} \) there is exactly one arrow from the initial object to \( B \). Any two initial objects are isomorphic, which is why we usually speak of the initial object. It is denoted 0, and the unique arrow from 0 to \( B \) is written \( i_B \) (pronounce “gnab”).

\[
0 \rightarrow i_B \rightarrow B
\]

The uniqueness can also be expressed as a universal property:

\[
f = i_B \iff \text{true},
\]

for all \( f : 0 \to B \). Instantiating \( f \) to the identity \( id_0 \), we obtain the reflection law: \( id_0 = i_B \). An arrow after a gnab can be fused into a single gnab.

\[
k \cdot i_{B'} = i_{B''},
\]

for all \( k : B' \to B'' \). The fusion law expresses that \( i_B \) is natural in \( B \).

Dually, 1 is a final object if for each object \( A \in \mathcal{C} \) there is a unique arrow from \( A \) to 1, denoted \( !_A \) (pronounce “bang”).

\[
A \rightarrow !_A \rightarrow 1
\]

5 Initial algebras and final coalgebras

Let \( F : \mathcal{C} \to \mathcal{C} \) be an endofunctor. An \( F \)-algebra is a pair \( \langle A, a \rangle \) consisting of an object \( A \in \mathcal{C} \) and an arrow \( a \in \mathcal{C}(FA, A) \). An \( F \)-homomorphism between algebras \( \langle A, a \rangle \) and \( \langle B, b \rangle \) is an arrow \( h \in \mathcal{C}(A, B) \) such that \( h \cdot a = b \cdot F h \).

\[
\begin{array}{ccc}
FA & \xrightarrow{Fh} & FB \\
\downarrow a & & \downarrow b \\
A & \xrightarrow{h} & B
\end{array}
\]

Identity is an \( F \)-homomorphism and \( F \)-homomorphisms compose. Consequently, the data defines a category, called \( \text{Alg}(F) \). The initial object in this category — if it exists — is the so-called initial \( F \)-algebra \( \langle \mu F, \in \rangle \). The import of initiality is that there is a unique arrow from \( \langle \mu F, \in \rangle \) to any other \( F \)-algebra \( \langle B, b \rangle \). This unique arrow is written \( \langle b \rangle \) and is called fold or catamorphism. Expressed in terms of the base category, it satisfies the following universal property.

\[
f = \langle b \rangle \iff f \cdot \in = b \cdot F f
\]

(20)
Like for products, the universal property has two immediate consequences. Substituting the left-hand side into the right-hand side gives the computation law:

$$\langle b \rangle \cdot \text{in} = b \cdot F \langle b \rangle.$$  

(21)

Setting $f := \text{id}$ and $b := \text{in}$, we obtain the reflection law:

$$\text{id} = \langle \text{in} \rangle.$$  

(22)

Since the initial algebra is an initial object, we also have a fusion law for fusing an arrow with a fold to form another fold.

$$k \cdot \langle b' \rangle = \langle b'' \rangle \iff k \cdot b' = b'' \cdot F k$$  

(23)

The proof is trivial if phrased in terms of the category $\text{Alg}(F)$. However, we can also execute the proof in the underlying category $C$.

\[
\begin{align*}
k \cdot \langle b' \rangle &= \langle b'' \rangle \\
\iff & \quad \text{universal property (20)} \} \\
& \quad k \cdot \langle b' \rangle \cdot \text{in} = b'' \cdot F (k \cdot \langle b' \rangle) \\
\iff & \quad \text{computation (21)} \} \\
& \quad k \cdot b' \cdot F \langle b' \rangle = b'' \cdot F \left( k \cdot \langle b' \rangle \right) \\
\iff & \quad \text{F functor} \} \\
& \quad k \cdot b' \cdot F \langle b' \rangle = b'' \cdot F k \cdot F \langle b' \rangle \\
\iff & \quad \text{cancel } \cdot \text{F} \langle b' \rangle \text{ on both sides} \} \\
& \quad k \cdot b' = b'' \cdot F k.
\end{align*}
\]

The fusion law states that $\langle - \rangle$ is natural in $\langle B, b \rangle$, that is, as an arrow in $\text{Alg}(F)$. This does not imply naturality in the underlying category $C$ (as an arrow in $C$ it is a strong dinatural transformation).

Using these laws we can show that $\mu F$ is indeed a fixed point of the functor: $F (\mu F) \cong \mu F$. The isomorphism is witnessed by the arrows $\text{in} \in C(F(\mu F), \mu F)$ and $\langle F \text{in} \rangle \in C(\mu F, F(\mu F))$. We calculate

$$\text{in} \cdot \langle F \text{in} \rangle = \text{id}$$

\[
\begin{align*}
\iff & \quad \text{reflection} \} \\
& \quad \text{in} \cdot \langle F \text{in} \rangle = \langle \text{in} \rangle \\
\iff & \quad \text{fusion (23)} \} \\
& \quad \text{in} \cdot F \text{in} = \text{in} \cdot F \text{in}
\end{align*}
\]

For the reverse direction, we reason

$$\langle F \text{in} \rangle \cdot \text{in}$$

\[
\begin{align*}
= & \quad \text{computation} \} \\
& \quad F \text{in} \cdot F \langle F \text{in} \rangle
\end{align*}
\]
\[
\begin{align*}
F id &= \{ \text{F functor}\} \\
F id &= \{ \text{see proof above}\} \\
F id &= \{ \text{F functor}\} \\
&= id.
\end{align*}
\]

Perhaps surprisingly, folds also enjoy a functor fusion law. To be able to formulate the law, we have to turn \(\mu\) into a higher-order functor of type \(C^C \to C\). The object part of this functor maps a functor to its initial algebra. The arrow part, which maps a natural transformation \(\alpha : F \to G\) to an arrow \(\mu \alpha \in C(\mu F, \mu G)\), is given by
\[
\mu \alpha = (\mu \cdot \alpha).
\] (24)

(To reduce clutter we have omitted the type argument of \(\alpha\) on the right-hand side, which should read \((\mu \cdot \alpha(\mu G))\). Like for products, we postpone the proof that \(\mu\) preserves identity and composition.

The functor fusion law states that we can fuse a fold after a map to form another fold:
\[
\langle b \cdot \alpha \rangle = \langle b \rangle \cdot \mu \alpha,
\] (25)

for all \(\alpha : F' \to F''\). To establish functor fusion we reason
\[
\begin{align*}
\langle b \cdot \alpha \rangle &= \langle b \rangle \cdot \mu \alpha \\
\iff & \{ \text{definition of } \mu (24) \} \\
\langle b \rangle \cdot \langle \mu \cdot \alpha \rangle &= \langle b \cdot \alpha \rangle \\
\iff & \{ \text{fusion (23)} \} \\
\langle b \rangle \cdot \mu \cdot \alpha &= b \cdot \alpha \cdot F'(\langle b \rangle) \\
\iff & \{ \text{computation (21)} \} \\
b \cdot F''\langle b \rangle \cdot \alpha &= b \cdot \alpha \cdot F'\langle b \rangle \\
\iff & \{ \text{naturality of } \alpha \} \\
b \cdot \alpha \cdot F'\langle b \rangle &= b \cdot \alpha \cdot F''\langle b \rangle.
\end{align*}
\]

Given these prerequisites, it is straightforward to show that \(\mu\) preserves identity
\[
\begin{align*}
\mu id &= \{ \text{definition of } \mu (24) \} \\
\langle in \cdot id \rangle &= \{ \text{identity and reflection (22)} \} \\
id
\end{align*}
\]
and composition
\[
\mu \beta \cdot \mu \alpha = \begin{cases} \text{definition of } \mu \end{cases} \\
\langle \text{in} \cdot \beta \rangle \cdot \mu \alpha = \begin{cases} \text{functor fusion (25)} \end{cases} \\
\langle \text{in} \cdot \beta \cdot \alpha \rangle = \begin{cases} \text{definition of } \mu \end{cases} \\
\mu (\beta \cdot \alpha).
\]

To summarise, functor fusion expresses that \(\langle - \rangle\) is natural in \(F\):
\[
\langle - \rangle : \forall F. C(F B, B) \to C(\mu F, B).
\]
The arrow \(\text{in} : F (\mu F) \to \mu F\) is natural in \(F\), as well. The arrow part of the higher-order functor \(A F \cdot F (\mu F)\) is \(\lambda \alpha. F'' (\mu \alpha) \cdot \alpha = \lambda \alpha. \alpha \cdot F' (\mu \alpha)\).
\[
\mu \alpha \cdot \text{in} = \text{in} \cdot \alpha \cdot F (\mu \alpha) \tag{26}
\]

We reason
\[
\mu \alpha \cdot \text{in} = \begin{cases} \text{definition of } \mu \end{cases} \\
\langle \text{in} \cdot \alpha \rangle \cdot \text{in} = \begin{cases} \text{computation (21)} \end{cases} \\
\text{in} \cdot \alpha \cdot F \langle \text{in} \cdot \alpha \rangle = \begin{cases} \text{definition of } \mu \end{cases} \\
\text{in} \cdot \alpha \cdot F (\mu \alpha).
\]

The development nicely dualises to \(F\)-coalgebras and unfolds. An \(F\)-coalgebra is a pair \(\langle A, a \rangle\) consisting of an object \(A \in C\) and an arrow \(a \in C(A, F A)\). An \(F\)-homomorphism between coalgebras \(\langle A, a \rangle\) and \(\langle B, b \rangle\) is an arrow \(h \in C(A, B)\) such that \(F h \cdot a = b \cdot h\). Identity is an \(F\)-homomorphism and \(F\)-homomorphisms compose. Consequently, the data defines a category, called \(\text{Coalg}(F)\). The final object in this category — if it exists — is the so-called final \(F\)-coalgebra \(\langle \nu F, \text{out} \rangle\). The import of finality is that there is a unique arrow to \(\langle \nu F, \text{out} \rangle\) from any other \(F\)-coalgebra \(\langle A, a \rangle\). This unique arrow is written \([a]\) and is called \(\text{unfold}\) or \(\text{anamorphism}\). Expressed in terms of the base category, it satisfies the following universal property.
\[
F g \cdot a = \text{out} \cdot g \iff [a] = g \tag{27}
\]

Like for initial algebras, the universal property implies computation, reflection, fusion and functor fusion laws. \(\text{Computation law:}\)
\[
F[a] \cdot a = \text{out} \cdot [a]. \tag{28}
\]
Reflection law:
\[ \{ \text{out} \} = \text{id}. \]  
(29)

Fusion law:
\[ \{ a' \} = \{ a'' \} \cdot h \iff F h \cdot a = a'' \cdot h. \]  
(30)
The object part of the functor \( \nu \) is defined
\[ \nu \alpha = \{ \alpha \cdot \text{out} \}. \]  
(31)

Functor fusion law:
\[ \nu \alpha \cdot \{ a \} = \{ \alpha \cdot a \} \]  
(32)
Finally, \( \text{out} \) is a natural transformation.
\[ \alpha \cdot F (\nu \alpha) \cdot \text{out} = \text{out} \cdot \nu \alpha \]

6 Adjunctions

We have noted in Sec. 3 that products and coproducts are part of an adjunction. In this section, we explore the notion of an adjunction in greater depth. Let \( \mathbb{C} \) and \( \mathbb{D} \) be categories. The functors \( L \) and \( R \) are adjoint, denoted \( L \dashv R \), if and only if there is a bijection
\[ \phi : \forall A B \cdot \mathbb{C}(L A, B) \cong \mathbb{D}(A, R B), \]
that is natural both in \( A \) and \( B \). The isomorphism \( \phi \) is called the adjoint transformation. It is also called the left adjunct with \( \phi^\circ \) being the right adjunct. That \( \phi \) and \( \phi^\circ \) are mutually inverse, can be captured using an equivalence.
\[ f = \phi^\circ g \iff \phi f = g \]  
(33)
(The left-hand side lives in \( \mathbb{C} \), and the right-hand side in \( \mathbb{D} \).) The formula is reminiscent of the universal property of products. That the latter indeed defines an adjunction can be seen more clearly if we re-formulate (2) in terms of the categories involved.
\[ f = \langle \text{outl}, \text{outr} \rangle \cdot \Delta g \iff \Delta f = g \]
The right part of the diagram below explicates the categories involved.
We actually have a double adjunction with + being left adjoint to \( \Delta \). Rewritten in terms of product categories, the universal property of coproducts (11) becomes

\[
  f = \nabla g \iff \Delta f \cdot \langle \text{inl}, \text{inr} \rangle = g.
\]

Initial objects and final objects also define (a rather trivial adjunction) between the category \( \mathbf{1} \) and \( \mathcal{C} \).

\[
\begin{array}{ccc}
\mathcal{C} & \Downarrow & \mathbf{0} \\
\Downarrow & \Delta & \Downarrow \\
\mathbf{1} & \Downarrow & \mathbf{1} \\
\end{array}
\begin{array}{ccc}
\Delta & \Arrow & \mathcal{C} \\
\Downarrow & \Downarrow & \Downarrow \\
\mathbf{1} & \Arrow & \mathbf{1} \\
\end{array}
\]

The category \( \mathbf{1} \) consists of a single object \(*\) and a single arrow \( \text{id}_* \). The diagonal functor is now defined \( \Delta A = * \) and \( \Delta f = \text{id}_* \). The objects 0 and 1 are seen as constant functors from \( \mathbf{1} \). (An object \( A \in \mathcal{C} \) seen as a functor \( A : \mathbf{1} \to \mathcal{C} \) maps \(*\) to \( A \) and \( \text{id}_* \) to \( \text{id}_A \).)

\[
  f = \text{id}_B \cdot 0 g \iff \Delta f \cdot \text{id}_* = g \tag{34}
\]
\[
  f = \text{id}_* \cdot \Delta g \iff \text{id} \cdot !_A = g \tag{35}
\]

The universal properties are a bit degenerated as the right-hand side of (34) and the left-hand side of (35) is vacuously true.

An adjunction can be defined in a variety of ways. An alternative approach makes use of two natural transformations: the counit \( \epsilon : L \circ R \to \text{Id} \) and the unit \( \eta : \text{Id} \to R \circ L \). For products, the counit is the pair of arrows \( \langle \text{outl}, \text{outr} \rangle \) and the unit is the diagonal arrow \( \delta = \text{id} \circ \text{id} \). The units must satisfy

\[
  (\epsilon \circ L) \cdot (L \circ \eta) = \text{id}_L \quad \text{and} \quad (R \circ \epsilon) \cdot (\eta \circ R) = \text{id}_R,
\]

where \( \circ \) denotes (horizontal) composition of a natural transformation with a functor: \( (F \circ \alpha) A = F (\alpha A) \) and \( (\alpha \circ F) A = \alpha (F A) \). It is useful to explicate the typing information.

You may want to think of \( L \) and \( R \) as closure operations. The unit laws express that going left-right-left is the same as going left once and likewise for going right.

All in all, an adjunction consists of six entities: two functors, two adjuncts, and two units. Every single of those can be defined in terms of the others:

\[
\begin{align*}
  \phi \circ g &= \epsilon \cdot L g & \epsilon &= \phi \circ \text{id} & L g &= \phi \circ (\eta \cdot g) \\
  \phi f &= R f \cdot \eta & \eta &= \phi \circ \text{id} & R f &= \phi \circ (f \cdot \epsilon).
\end{align*}
\]
In terms of programming language concepts, adjuncts correspond to introduction and elimination rules (\(\triangle\) introduces a pair, \(\triangledown\) eliminates a sum). The units can be seen as simple variants of these rules (\(\langle\text{outl}, \text{outr}\rangle\) eliminates a pair and \(\langle\text{inl}, \text{inr}\rangle\) introduces a sum). When we discussed products, we derived a variety of laws from the universal property. Table 1 re-formulates these laws using the new vocabulary. For instance, from the perspective of the right adjoint \(f = \phi \circ (\phi f)\) corresponds to a computation law or \(\beta\)-rule, viewed from the left it is an \(\eta\)-rule.\(^3\) The table merits careful study. Table 2 lists some examples of adjunctions.

Since the components of an adjunction are inter-definable, an adjunction can be specified by providing only part of the data. Surprisingly little is needed: for products only the functor \(L\) and the counit \(\epsilon\) were given, the other ingredients were derived from those. In the rest of this section, we replay the derivation in terms of adjunctions. Let \(L : \mathcal{D} \to \mathcal{C}\) be a functor, and let \(\epsilon \in \mathcal{C}(L(R B), B)\) be a universal arrow. Universality means that for each \(f \in \mathcal{C}(L A, B)\) there exists an

\(^3\) It is a coincidence that the same Greek letter is used both for extensionality (\(\eta\)-rule) and for the unit of an adjunction.
Table 2. Examples of adjunctions.

<table>
<thead>
<tr>
<th>adjunction</th>
<th>initial object</th>
<th>final object</th>
<th>coproduct</th>
<th>product</th>
<th>exponential</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>0</td>
<td>Δ</td>
<td>+</td>
<td>Δ</td>
<td>− × X</td>
</tr>
<tr>
<td>R</td>
<td>Δ</td>
<td>1</td>
<td>×</td>
<td>△</td>
<td>uncurry</td>
</tr>
<tr>
<td>φ</td>
<td>∇</td>
<td>△</td>
<td>△</td>
<td>outl, outr</td>
<td></td>
</tr>
<tr>
<td>ϵ</td>
<td>!</td>
<td>⟨inl, inr⟩</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>η</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>apply</td>
</tr>
</tbody>
</table>

arrow φf ∈ D(A, RB) such that

\[ f = \epsilon \cdot L \cdot g \iff \phi \cdot f = g, \]  
(36)

for all g ∈ D(A, RB). The formula suggests that \( \epsilon \cdot L \cdot g = \phi \circ g \). Computation law: substituting the right-hand side into the left-hand side, we obtain

\[ f = \epsilon \cdot L (\phi \cdot f), \]  
(37)

Reflection law: setting \( f := \epsilon \) and \( g := id \), yields

\[ \phi \cdot \epsilon = id \]  
(38)

Fusion law: to establish

\[ \phi (f \cdot L \cdot h) = \phi \cdot f \cdot h, \]  
(39)

we appeal to the universal property:

\[ f \cdot L \cdot h = \epsilon \cdot L (\phi \cdot f \cdot h) \iff \phi \cdot (f \cdot L \cdot h) = \phi \cdot f \cdot h. \]

To show the left-hand side, we calculate

\[ \epsilon \cdot L (\phi \cdot f \cdot h) \]
\[ = \{ L \text{ functor} \} \]
\[ \epsilon \cdot L (\phi \cdot f) \cdot L \cdot h \]
\[ = \{ \text{computation (37)} \} \]
\[ f \cdot L \cdot h. \]

The type constructor \( R \) can be turned into a functor whose action on arrows is defined \( Rf = \phi (f \cdot \epsilon) \). (The definition is suggested by combining reflection and functor fusion: \( Rf = R \cdot \phi \cdot \epsilon = \phi (f \cdot \epsilon). \) Functory fusion law:

\[ R k \cdot \phi f = \phi (k \cdot f). \]  
(40)
For the proof, we reason
\[
Rk \cdot \phi f = \{ \text{definition of } R \} \\
\phi (k \cdot \epsilon) \cdot \phi f = \{ \text{fusion (39)} \} \\
\phi (k \cdot \epsilon \cdot L(\phi f)) = \{ \text{computation (37)} \} \\
\phi (k \cdot f).
\]

**Functoriality:** $R$ preserves identity
\[
R \text{id} = \{ \text{definition of } R \} \\
\phi (\text{id} \cdot \epsilon) = \{ \text{identity and reflection (38)} \} \\
\text{id}
\]

and composition
\[
Rg \cdot Rf = \{ \text{definition of } R \} \\
Rg \cdot \phi (f \cdot \epsilon) = \{ \text{functor fusion (40)} \} \\
\phi (g \cdot f \cdot \epsilon) = \{ \text{definition of } R \} \\
R (g \cdot f).
\]

Fusion and functor fusion show that $\phi$ is natural both in $A$ and in $B$. Finally, the counit $\epsilon$ is natural in $B$.
\[
\epsilon \cdot L(Rk) = \{ \text{definition of } R \} \\
\epsilon \cdot L(\phi (k \cdot \epsilon)) = \{ \text{computation (37)} \} \\
k \cdot \epsilon
\]

Dually, a functor $R$ and a universal arrow $\eta \in C(A, R(LA))$ are sufficient.
\[
f = \phi \circ g \iff Rf \cdot \eta = g.
\]

Define $\phi f = Rf \cdot \eta$ and $Lg = \phi \circ (\eta \cdot g)$. 