Conditionals in distributive categories

Jeremy Gibbons

August 1996

Abstract

In a distributive category (a category in which the product distributes over the coproduct), coproducts can be used to model conditional expressions. We develop such a theory of conditionals.

1 Introduction

The category **Set** of sets and total functions is a *distributive category* [1], which is to say that the categorical product in **Set** (namely cartesian product) distributes over the coproduct (disjoint sum). In such a category, coproducts can be used to model conditionals. In this paper we show how this is done, developing a theory of conditionals as we go.

We write '$a \in A$' to denote that element a is in datatype A, and '$f : A \to B$' to denote that function f is from datatype A to datatype B. We write $f.a$ for the result of applying function f to element a; application associates to the right. Backwards composition of functions is written '$f \circ g$', so that $(f \circ g).a = f.g.a$. We use sans serif type for 'global' names, with the same meaning throughout the paper, and italic type for local names with limited scope.

2 Products and coproducts

In this section, we briefly describe the well-known categorical notions of product and coproduct. We stick to the categorical names 'product' and 'coproduct' rather than the more concrete 'cartesian product' and 'disjoint sum', because we will later refer to categories in which these do not coincide.

2.1 Product

The product $A \times B$ of two datatypes A and B is a datatype. There are two projections $\mathsf{fst} : A \times B \to A$ and $\mathsf{snd} : A \times B \to B$.

That $A \times B$ is a product is to say that, for fixed $f : C \to A$ and $g : C \to B$, there is a unique function h such that $\mathsf{fst} \circ h = f$ and $\mathsf{snd} \circ h = g$. This h is written '$f \vartriangle g$'; the uniqueness is expressed by the following universal property.

Definition 1.

$$h = f \vartriangle g \quad \equiv \quad \mathsf{fst} \circ h = f \text{ and } \mathsf{snd} \circ h = g$$

The product $f \times g$ of two functions $f : A \to B$ and $g : C \to D$ is a function of type $A \times C \to B \times D$, and is defined as follows.

Definition 2.

$$f \times g \quad = \quad (f \circ \mathsf{fst}) \vartriangle (g \circ \mathsf{snd})$$

Many properties of products follow directly from the universal property characterizing forks. Among them are the following, which we state without proof.

Theorem 3.

(i) any function returning a pair is a fork:

$$h \quad = \quad (\mathsf{fst} \circ h) \vartriangle (\mathsf{snd} \circ h)$$

(ii) projections eliminate forks:

$$\mathsf{fst} \circ (f \vartriangle g) \quad = \quad f \qquad \mathsf{snd} \circ (f \vartriangle g) \quad = \quad g$$

(iii) any function fuses with a fork:

$$(f \vartriangle g) \circ h \quad = \quad (f \circ h) \vartriangle (g \circ h)$$

(iv) product respects identity:

$$\mathsf{id} \times \mathsf{id} \quad = \quad \mathsf{id}$$

(v) product distributes over composition:

$$(f \circ g) \times (h \circ j) \quad = \quad (f \times h) \circ (g \times j)$$

(vi) projections promote through product:

$$\mathsf{fst} \circ (f \times g) \quad = \quad f \circ \mathsf{fst} \qquad \mathsf{snd} \circ (f \times g) \quad = \quad g \times \mathsf{snd}$$

(vii) product fuses with fork:

$$(f \times g) \circ (h \vartriangle j) \quad = \quad (f \circ h) \vartriangle (g \circ j)$$

We leave it to the reader to verify that, in the category **Set** of sets and total functions, cartesian product is a categorical product.

2.2 Coproduct

Coproducts are dual to products. The coproduct $A+B$ of two datatypes A and B is a datatype. There are two injections $\mathsf{inl} : A \to A+B$ and $\mathsf{inr} : B \to A+B$.

That $A+B$ is a coproduct is to say that, for fixed $f : A \to C$ and $g : B \to C$, there is a unique function h such that $h \circ \mathsf{inl} = f$ and $h \circ \mathsf{inr} = g$. This h is written '$f \triangledown g$'; the uniqueness is expressed by the following universal property.

Definition 4.

$$h = f \triangledown g \quad \equiv \quad h \circ \mathsf{inl} = f \text{ and } h \circ \mathsf{inr} = g$$

The coproduct $f+g$ of two functions $f : A \to B$ and $g : C \to D$ is a function of type $A+C \to B+D$, and is defined as follows.

Definition 5.

$$f+g \quad = \quad (\mathsf{inl} \circ f) \triangledown (\mathsf{inr} \circ g)$$

Coproducts too enjoy many properties, the duals of those relating to products.

Theorem 6.

(i) any function from a coproduct is a join:

$$h \quad = \quad (h \circ \mathsf{inl}) \triangledown (h \circ \mathsf{inr})$$

(ii) injections eliminate joins:

$$(f \triangledown g) \circ \mathsf{inl} \quad = \quad f \qquad (f \triangledown g) \circ \mathsf{inr} \quad = \quad g$$

(iii) any function fuses with a join:

$$h \circ (f \triangledown g) \quad = \quad (h \circ f) \triangledown (h \circ g)$$

(iv) coproduct respects identity:

$$\mathsf{id} + \mathsf{id} \quad = \quad \mathsf{id}$$

(v) coproduct distributes over composition:

$$(f \circ g) + (h \circ j) \quad = \quad (f+h) \circ (g+j)$$

(vi) injections promote through coproduct:

$$(f \times g) \circ \mathsf{inl} \quad = \quad \mathsf{inl} \circ f \qquad (f \times g) \circ \mathsf{inr} \quad = \quad \mathsf{inr} \circ g$$

(vii) join fuses with coproduct:

$$(f \triangledown g) \circ (h+j) \quad = \quad (f \circ h) \triangledown (g \circ j)$$

In **Set**, disjoint sum is a categorical coproduct.

2.3 The exchange law

An elegant law connecting products and coproducts is the *exchange law*, stating that a fork of joins is also a join of forks.

Theorem 7. *For any functions f, g, h, j,*

$$(f \vartriangle g) \triangledown (h \vartriangle j) \quad = \quad (f \triangledown h) \vartriangle (g \triangledown j)$$

Proof. There are two proofs, each the dual of the other. One proof uses the universal property of forks:

$$\begin{array}{ll}
& (f \vartriangle g) \triangledown (h \vartriangle j) = (f \triangledown h) \vartriangle (g \triangledown j) \\
\equiv & \quad \{ \text{ universal property of fork } \} \\
& \mathsf{fst} \circ ((f \vartriangle g) \triangledown (h \vartriangle j)) = f \triangledown h \;\text{ and }\; \mathsf{snd} \circ ((f \vartriangle g) \triangledown (h \vartriangle j)) = g \triangledown j \\
\equiv & \quad \{ \text{ join fusion } \} \\
& (\mathsf{fst} \circ (f \vartriangle g)) \triangledown (\mathsf{fst} \circ (h \vartriangle j)) = f \triangledown h \;\text{ and }\; \mathsf{snd} \circ ((f \vartriangle g) \triangledown (h \vartriangle j)) = g \triangledown j \\
\equiv & \quad \{ \text{ projection eliminates fork } \} \\
& \mathsf{true} \;\text{ and }\; \mathsf{snd} \circ ((f \vartriangle g) \triangledown (h \vartriangle j)) = g \triangledown j \\
\equiv & \quad \{ \text{ similarly on right-hand side } \} \\
& \mathsf{true}
\end{array}$$

The other proof uses the universal property of joins, and is left to the interested reader.
□

3 Distributive categories

Cartesian product of sets distributes over disjoint sum: the two sets $A \times (B + C)$ and $(A \times B) + (A \times C)$ are isomorphic, as are $(B + C) \times A$ and $(B \times A) + (C \times A)$. (Intuitively, given an A and either a B or a C, one can construct either an A and a B, or an A and a C; conversely, given either an A and a B or an A and a C, one can construct both an A and either a B or a C.) Therefore, in **Set**, we would expect the two objects $A \times (B + C)$ and $(A \times B) + (A + C)$ to be isomorphic, that is, for there to be functions $\mathsf{distl} : A \times (B + C) \to (A \times B) + (A \times C)$ distributing the left half of the product over the coproduct on the right, and $\mathsf{undistl} : (A \times B) + (A \times C) \to A \times (B + C)$ factoring or 'undistributing' the product back again, and for each to be the other's inverse.

It is easy to construct the function $\mathsf{undistl}$. In fact, it can be constructed in two different ways. Since its source type is a coproduct, it is a join; indeed, the definition below gives the required type.

Definition 8.

$$\mathsf{undistl} \quad = \quad (\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr})$$

Moreover, since its target type is a product, it is a fork, and it turns out that the expression

$$(\mathsf{fst} \triangledown \mathsf{fst}) \vartriangle (\mathsf{snd} + \mathsf{snd})$$

also has the required type. As we might expect, these two expressions are equal, on account of the exchange law.

Theorem 9.

$$(\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr}) \quad = \quad (\mathsf{fst} \triangledown \mathsf{fst}) \vartriangle (\mathsf{snd} + \mathsf{snd})$$

Proof.

$$\begin{array}{ll}
& (\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr}) \\
= & \quad \{ \text{ product as fork } \} \\
& (\mathsf{fst} \vartriangle (\mathsf{inl} \circ \mathsf{snd})) \triangledown (\mathsf{fst} \vartriangle (\mathsf{inr} \circ \mathsf{snd})) \\
= & \quad \{ \text{ exchange law } \} \\
& (\mathsf{fst} \triangledown \mathsf{fst}) \vartriangle ((\mathsf{inl} \circ \mathsf{snd}) \triangledown (\mathsf{inr} \circ \mathsf{snd})) \\
= & \quad \{ \text{ coproduct as join } \} \\
& (\mathsf{fst} \triangledown \mathsf{fst}) \vartriangle (\mathsf{snd} + \mathsf{snd})
\end{array}$$

□

The construction of undistl did not depend in any way on the particular category **Set**; the same construction works in any category with products and coproducts. However, an inverse function distl is not so easy to construct. The source type of such an inverse is a product, so the inverse is not a join, and its target type is a coproduct, so it is not a fork. Indeed, in some categories with products and coproducts, there is no inverse to undistl[1], so we should not expect to be able to construct one just from properties of products and coproducts.

We can do no better than to *assume* the existence of a function distl : $A \times (B + C) \to (A \times B) + (A \times C)$, and to assume that undistl and distl are each other's inverses. This is precisely what it means for the categorical setting to be a distributive category.

[1]We will see some counter-examples in Section 3.4.

3.1 Evaluation rules

Although we cannot give a point-free definition of distl, we can deduce a pointwise definition by pattern-matching from the definition of $\mathsf{undistl}$. (Indeed, it could be said that distributive categories are those settings which permit definition by pattern-matching.) It is easy to deduce the following evaluation rules for $\mathsf{undistl}$:

$$\mathsf{undistl}.\mathsf{inl}.(a,b) \;=\; (a,\mathsf{inl}.b) \qquad \mathsf{undistl}.\mathsf{inr}.(a,c) \;=\; (a,\mathsf{inr}.c)$$

Therefore, distl, being the inverse of $\mathsf{undistl}$, obeys the following evaluation rules:

$$\mathsf{distl}.(a,\mathsf{inl}.b) \;=\; \mathsf{inl}.(a,b) \qquad \mathsf{distl}.(a,\mathsf{inr}.c) \;=\; \mathsf{inr}.(a,c)$$

These rules completely determine the behaviour of distl, since the two patterns on the left-hand side are exhaustive and mutually exclusive.

3.2 Naturality

It is not hard to show that $\mathsf{undistl}$ is a natural transformation:

Theorem 10.

$$(f \times (g+h)) \circ \mathsf{undistl} \;=\; \mathsf{undistl} \circ ((f \times g) + (f \times h))$$

Naturality is closely related to so-called 'natural polymorphism'. Intuitively, the only way for $\mathsf{undistl}$ to commute with all functions 'in the obvious way' like this is for $\mathsf{undistl}$ to perform purely structural rearrangements, entirely independently of the values of the various components of these structures. Purely structural rearrangements are necessarily naturally polymorphic.

Proof (of Theorem 10).

$$\begin{array}{cl}
 & (f \times (g+h)) \circ \mathsf{undistl} \\
= & \quad \{\ \mathsf{undistl}\ \} \\
 & (f \times (g+h)) \circ ((\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr})) \\
= & \quad \{\ \text{join fusion}\ \} \\
 & ((f \times (g+h)) \circ (\mathsf{id} \times \mathsf{inl})) \triangledown ((f \times (g+h)) \circ (\mathsf{id} \times \mathsf{inr})) \\
= & \quad \{\ \text{product distributes over composition}\ \} \\
 & (f \times ((g+h) \circ \mathsf{inl})) \triangledown (f \times ((g+h) \circ \mathsf{inr})) \\
= & \quad \{\ \text{injections promote through coproduct}\ \} \\
 & (f \times (\mathsf{inl} \circ g)) \triangledown (f \times (\mathsf{inr} \circ h)) \\
= & \quad \{\ \text{product distributes over composition}\ \}
\end{array}$$

$$
\begin{array}{ll}
 & ((\mathsf{id} \times \mathsf{inl}) \circ (f \times g)) \triangledown ((\mathsf{id} \times \mathsf{inr}) \circ (f \times h)) \\
= & \quad \{ \text{ join fuses with coproduct } \} \\
 & ((\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr})) \circ ((f \times g) + (f \times h)) \\
= & \quad \{ \ \mathsf{undistl} \ \} \\
 & \mathsf{undistl} \circ ((f \times g) + (f \times h))
\end{array}
$$

□

A similar property therefore holds for distl too, because it is the inverse of undistl:

Theorem 11.

$$
\mathsf{distl} \circ (f \times (g + h)) \quad = \quad ((f \times g) + (f \times h)) \circ \mathsf{distl}
$$

Proof.

$$
\begin{array}{ll}
 & \mathsf{distl} \circ (f \times (g + h)) \\
= & \quad \{ \ \mathsf{undistl} \circ \mathsf{distl} = \mathsf{id} \ \} \\
 & \mathsf{distl} \circ (f \times (g + h)) \circ \mathsf{undistl} \circ \mathsf{distl} \\
= & \quad \{ \text{ naturality of } \mathsf{undistl} \ \} \\
 & \mathsf{distl} \circ \mathsf{undistl} \circ ((f \times g) + (f \times h)) \circ \mathsf{distl} \\
= & \quad \{ \ \mathsf{distl} \circ \mathsf{undistl} = \mathsf{id} \ \} \\
 & ((f \times g) + (f \times h)) \circ \mathsf{distl}
\end{array}
$$

□

Many properties of distl follow easily in this way from the corresponding property of undistl.

3.3 Distributing rightwards

Dual to undistl and distl are undistr : $(A \times C) + (B \times C) \to (A + B) \times C$ and distr : $(A + B) \times C \to (A \times C) + (B \times C)$, which factor and distribute the right-hand half of a product through a coproduct. The two ways of defining undistr are as follows.

Definition 12.

$$
\mathsf{undistr} \quad = \quad (\mathsf{inl} \times \mathsf{id}) \triangledown (\mathsf{inr} \times \mathsf{id}) \quad = \quad (\mathsf{fst} + \mathsf{fst}) \vartriangle (\mathsf{snd} \triangledown \mathsf{snd})
$$

The evaluation rules are

$$\begin{array}{rclcrcl} \mathsf{undistr}.\mathsf{inl}.(a,c) &=& (\mathsf{inl}.a,c) & \quad & \mathsf{distr}.(\mathsf{inl}.a,c) &=& \mathsf{inl}.(a,c) \\ \mathsf{undistr}.\mathsf{inr}.(b,c) &=& (\mathsf{inr}.b,c) & \quad & \mathsf{distr}.(\mathsf{inr}.b,c) &=& \mathsf{inr}.(b,c) \end{array}$$

and the naturality properties are given by the following theorem.

Theorem 13.

$$\begin{array}{rcl} ((f+g)\times h)\circ\mathsf{undistr} &=& \mathsf{undistr}\circ((f\times h)+(g\times h)) \\ ((f\times h)+(g\times h))\circ\mathsf{distr} &=& \mathsf{distr}\circ((f+g)\times h) \end{array}$$

3.4 Non-distributive categories

In a number of other categories modelling datatypes and mappings between them (for example, **Pfn**, where the mappings are partial functions; **Rel**, where the mappings are relations; and **Mfn**, where the mappings $f : A \to B$ are (total) functions taking an element of A to a subset of B, with Kleisli composition), the cartesian product is not a categorical product. With suitable definitions on mappings, the datatype $(A \times B) + A + B$ does form a categorical product, but this product does not distribute over disjoint sum, and so the categories are not distributive.

Still, with a couple of minor variations, much of the theory developed in this paper continues to apply in these categories too, using the cartesian product as a 'categorical pseudo-product'. One variation is that a couple of laws depend on determinism of a mapping, which can be encapsulated in the law

$$(f\circ h)\vartriangle(g\circ h) \quad = \quad (f\vartriangle g)\circ h$$

This law holds in **Pfn**, but not in general—only for deterministic h—in **Rel** or **Mfn**. Another variation is that a couple of laws require totality, which can be encapsulated by the law

$$\mathsf{fst}\circ(f\vartriangle g) \quad = \quad f$$

which does not hold in general—only for g whose domain includes the domain of f—in any of **Pfn**, **Rel** or **Mfn**. (Indeed, the failure of this law is essentially the reason why cartesian product is not a categorical product in these categories.)

4 Booleans and guards

The datatype Bool of booleans can be represented by $\mathbf{1}+\mathbf{1}$, where $\mathbf{1}$ is the unit type consisting of precisely one value, written (). The intention is that $\mathsf{inl}.()$ represents true and $\mathsf{inr}.()$ represents false. Negation is just a matter of swapping round the components of the coproduct; if the polymorphic function $\mathsf{swap} : A + B \to B + A$ is defined by

$$\mathsf{swap} \;\; = \;\; \mathsf{inr} \triangledown \mathsf{inl}$$

then not is just a monomorphic swap with $A = B = \mathbf{1}$.

Boolean conjunction and disjunction are more difficult, and require at least one of distl or distr, and hence a distributive category. We leave it to the reader to verify that the following two definitions, both of type $\mathsf{Bool} \times \mathsf{Bool} \to \mathsf{Bool}$, have the required behaviour:

$$\begin{array}{rcl} \mathsf{and} & = & (\mathsf{snd} \triangledown \mathsf{const.false}) \circ \mathsf{distr} \\ \mathsf{or} & = & (\mathsf{const.true} \triangledown \mathsf{snd}) \circ \mathsf{distr} \end{array}$$

Here, $\mathsf{const}.a$ is the function that returns a for any argument.

In a setting with partial functions and non-strict semantics, these definitions are strict in the first argument and non-strict in the second. Dual versions constructed from distl will have dual strictness properties. Completely strict versions can be constructed using both distl and distr, exploiting the fact that in a distributive category the datatypes $\mathsf{Bool} \times \mathsf{Bool}$ and $\mathbf{1} + \mathbf{1} + \mathbf{1} + \mathbf{1}$ are isomorphic and using a 'definition by truth table' approach.

4.1 Guards

A predicate on a type A is just a function of type $A \to \mathsf{Bool}$ (recall that $\mathsf{Bool} = \mathbf{1}+\mathbf{1}$). For predicate $p : A \to \mathsf{Bool}$, we define the *guard* $p? : A \to A + A$ as follows.

Definition 14.

$$p? \;\; = \;\; (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle p)$$

Intuitively, $p?.a$ returns $\mathsf{inl}.a$ if p holds of a, and $\mathsf{inr}.a$ otherwise.

Guards satisfy a number of properties, some of which are given below. Proofs of a number of them are simplified with the use of the following lemma.

Lemma 15.

$$(f + f) \circ p? \;\; = \;\; (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (f \vartriangle p)$$

Proof. We have:

$$\begin{array}{cl} & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (f \vartriangle p) \\ = & \quad \{ \text{ product fuses with fork } \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (f \times (\mathsf{id} + \mathsf{id})) \circ (\mathsf{id} \vartriangle p) \\ = & \quad \{ \text{ naturality } \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ ((f \times \mathsf{id}) + (f \times \mathsf{id})) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle p) \end{array}$$

$$
\begin{array}{ll}
= & \quad \{ \text{ pairs } \} \\
 & (f + f) \circ (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle p) \\
= & \quad \{ \text{ guards } \} \\
 & (f + f) \circ p?
\end{array}
$$

□

A guard can be promoted through a (deterministic) function:

Theorem 16.

$$
p? \circ f \quad = \quad (f + f) \circ (p \circ f)?
$$

Proof. We have:

$$
\begin{array}{ll}
 & (f + f) \circ (p \circ f)? \\
= & \quad \{ \text{ Lemma 15 } \} \\
 & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (f \vartriangle (p \circ f)) \\
= & \quad \{ \; f \text{ is deterministic } \} \\
 & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle p) \circ f \\
= & \quad \{ \text{ guards } \} \\
 & p? \circ f
\end{array}
$$

Note that determinism of f is used; this property does not hold in general in **Rel** or **Mfn**.

□

A total guard can be inverted, by discarding the information about whether it holds:

Theorem 17. *For total p,*

$$
(\mathsf{id} \triangledown \mathsf{id}) \circ p? \quad = \quad \mathsf{id}
$$

Proof. We have:

$$
\begin{array}{ll}
 & (\mathsf{id} \triangledown \mathsf{id}) \circ p? \\
= & \quad \{ \text{ guards } \} \\
 & (\mathsf{id} \triangledown \mathsf{id}) \circ (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle p) \\
= & \quad \{ \text{ pairs } \} \\
 & \mathsf{fst} \circ (\mathsf{id} \triangledown \mathsf{id}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle p) \\
= & \quad \{ \; (\mathsf{id} \triangledown \mathsf{id}) \circ \mathsf{distl} = \mathsf{id} \times (\mathsf{id} \triangledown \mathsf{id}) \text{ (see below) } \} \\
 & \mathsf{fst} \circ (\mathsf{id} \times (\mathsf{id} \triangledown \mathsf{id})) \circ (\mathsf{id} \vartriangle p) \\
= & \quad \{ \text{ pairs; } p \text{ is total } \} \\
 & \mathsf{id}
\end{array}
$$

(Note that totality of p is used; this law does not hold in general in any of **Pfn**, **Rel** or **Mfn**.) For the intermediate proof obligation, we have

$$\begin{array}{ll} & (\mathsf{id} \times (\mathsf{id} \triangledown \mathsf{id})) \circ \mathsf{undistl} \\ = & \quad \{ \ \mathsf{undistl} \ \} \\ & (\mathsf{id} \times (\mathsf{id} \triangledown \mathsf{id})) \circ ((\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr})) \\ = & \quad \{ \ \text{join fusion} \ \} \\ & ((\mathsf{id} \times (\mathsf{id} \triangledown \mathsf{id})) \circ (\mathsf{id} \times \mathsf{inl})) \triangledown ((\mathsf{id} \times (\mathsf{id} \triangledown \mathsf{id})) \circ (\mathsf{id} \times \mathsf{inr})) \\ = & \quad \{ \ \text{pairs} \ \} \\ & \mathsf{id} \triangledown \mathsf{id} \end{array}$$

and so

$$(\mathsf{id} \triangledown \mathsf{id}) \circ \mathsf{distl} \quad = \quad \mathsf{id} \times (\mathsf{id} \triangledown \mathsf{id})$$

□

Guarding with the negation of a predicate p gives the opposite result from guarding with p:

Theorem 18.

$$(\mathsf{not} \circ p)? \quad = \quad \mathsf{swap} \circ p?$$

Proof. We have:

$$\begin{array}{ll} & (\mathsf{not} \circ p)? \\ = & \quad \{ \ \text{guards} \ \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle (\mathsf{not} \circ p)) \\ = & \quad \{ \ \text{product fuses with fork} \ \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \times \mathsf{not}) \circ (\mathsf{id} \vartriangle p) \\ = & \quad \{ \ \mathsf{distl} \circ (\mathsf{id} \times \mathsf{swap}) = \mathsf{swap} \circ \mathsf{distl} \text{ (see below)} \ \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{swap} \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle p) \\ = & \quad \{ \ \text{pairs: } (f + g) \circ \mathsf{swap} = \mathsf{swap} \circ (g + f) \ \} \\ & \mathsf{swap} \circ (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle p) \\ = & \quad \{ \ \text{guards} \ \} \\ & \mathsf{swap} \circ p? \end{array}$$

For the proof obligation, we have

$$\begin{array}{ll} & \mathsf{undistl} \circ \mathsf{swap} \\ = & \quad \{ \ \mathsf{swap} \ \} \end{array}$$

$$
\begin{aligned}
& \mathsf{undistl} \circ (\mathsf{inr} \triangledown \mathsf{inl}) \\
= \quad & \{ \text{ join fusion } \} \\
& (\mathsf{undistl} \circ \mathsf{inr}) \triangledown (\mathsf{undistl} \circ \mathsf{inl}) \\
= \quad & \{ \ \mathsf{undistl} \ \} \\
& (((\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr})) \circ \mathsf{inr}) \triangledown (\mathsf{undistl} \circ \mathsf{inl}) \\
= \quad & \{ \text{ injections eliminate joins } \} \\
& (\mathsf{id} \times \mathsf{inr}) \triangledown (\mathsf{undistl} \circ \mathsf{inl}) \\
= \quad & \{ \ \mathsf{swap} \circ \mathsf{inl} = \mathsf{inr} \ \} \\
& (\mathsf{id} \times (\mathsf{swap} \circ \mathsf{inl})) \triangledown (\mathsf{undistl} \circ \mathsf{inl}) \\
= \quad & \{ \text{ similarly on the right-hand side } \} \\
& (\mathsf{id} \times (\mathsf{swap} \circ \mathsf{inl})) \triangledown (\mathsf{id} \times (\mathsf{swap} \circ \mathsf{inr})) \\
= \quad & \{ \text{ join fusion } \} \\
& (\mathsf{id} \times \mathsf{swap}) \circ ((\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr})) \\
= \quad & \{ \ \mathsf{undistl} \ \} \\
& (\mathsf{id} \times \mathsf{swap}) \circ \mathsf{undistl}
\end{aligned}
$$

and therefore

$$
\mathsf{swap} \circ \mathsf{distl} \quad = \quad \mathsf{distl} \circ (\mathsf{id} \times \mathsf{swap})
$$

□

A constantly-true guard is just an injection:

Theorem 19.

$$
\mathsf{const.true}? \quad = \quad \mathsf{inl}
$$

Proof. We have

$$
\begin{aligned}
& \mathsf{const.true}? \\
= \quad & \{ \text{ guards } \} \\
& (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle \mathsf{const.true}) \\
= \quad & \{ \ \mathsf{true} = \mathsf{inl}.(); \ \mathsf{const}.f.a = f \circ \mathsf{const}.a \ \} \\
& (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \times \mathsf{inl}) \circ (\mathsf{id} \vartriangle \mathsf{const}.()) \\
= \quad & \{ \ \mathsf{id} \times \mathsf{inl} = \mathsf{undistl} \circ \mathsf{inl} \ \} \\
& (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ \mathsf{undistl} \circ \mathsf{inl} \circ (\mathsf{id} \vartriangle \mathsf{const}.()) \\
= \quad & \{ \ \mathsf{distl} \circ \mathsf{undistl} = \mathsf{id}; \text{ pairs } \} \\
& \mathsf{inl}
\end{aligned}
$$

□

Applying a deterministic predicate twice in succession gives the same result both times:

Theorem 20. *For deterministic p,*

$$(p? + p?) \circ p? \quad = \quad (\mathsf{inl} + \mathsf{inr}) \circ p?$$

Proof. We note first that

$$p \quad = \quad (\mathsf{const}.() + \mathsf{const}.()) \circ p?$$

Now,

$$\begin{array}{cl} & (p? + p?) \circ p? \\ = & \quad \{ \text{ Lemma 15 } \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (p? \vartriangle p) \\ = & \quad \{ \text{ observation above } \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (p? \vartriangle ((\mathsf{const}.() + \mathsf{const}.()) \circ p?)) \\ = & \quad \{ \ p? \text{ is deterministic } \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle (\mathsf{const}.() + \mathsf{const}.())) \circ p? \\ = & \quad \{ \ \mathsf{distl} \circ (\mathsf{id} \vartriangle (f + g)) = (\mathsf{inl} \vartriangle f) + (\mathsf{inr} \vartriangle g) \text{ (see below) } \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ ((\mathsf{inl} \vartriangle \mathsf{const}.()) + (\mathsf{inr} \vartriangle \mathsf{const}.())) \circ p? \\ = & \quad \{ \text{ pairs } \} \\ & (\mathsf{inl} + \mathsf{inr}) \circ p? \end{array}$$

For the proof obligation, we have

$$\begin{array}{cl} & \mathsf{undistl} \circ ((\mathsf{inl} \vartriangle f) + (\mathsf{inr} \vartriangle g)) \\ = & \quad \{ \ \mathsf{undistl} \ \} \\ & ((\mathsf{id} \times \mathsf{inl}) \triangledown (\mathsf{id} \times \mathsf{inr})) \circ ((\mathsf{inl} \vartriangle f) + (\mathsf{inr} \vartriangle g)) \\ = & \quad \{ \text{ join fuses with coproduct } \} \\ & (\mathsf{inl} \vartriangle (\mathsf{inl} \circ f)) \triangledown (\mathsf{inr} \vartriangle (\mathsf{inr} \circ g)) \\ = & \quad \{ \text{ exchange law } \} \\ & (\mathsf{inl} \triangledown \mathsf{inr}) \vartriangle ((\mathsf{inl} \circ f) \triangledown (\mathsf{inr} \circ g)) \\ = & \quad \{ \text{ pairs } \} \\ & \mathsf{id} \vartriangle (f + g) \end{array}$$

and so

$$\mathsf{distl} \circ (\mathsf{id} \vartriangle (f + g)) \quad = \quad (\mathsf{inl} \vartriangle f) + (\mathsf{inr} \vartriangle g)$$

□

5 Conditionals

Guards are the basic building block for defining *conditionals*, a higher-level construct more suitable for programming.

Definition 21.

$$\textsf{if } p \textsf{ then } f \textsf{ else } g \quad = \quad (f \triangledown g) \circ p?$$

Conditionals too enjoy a number of properties, most following easily from corresponding properties of guards. Among them are the following. This list includes all of the laws of conditionals from [2].

A conditional fuses with a function, in both directions:

Theorem 22.

$$(\textsf{if } p \textsf{ then } f \textsf{ else } g) \circ h \quad = \quad \textsf{if } (p \circ h) \textsf{ then } (f \circ h) \textsf{ else } (g \circ h)$$

Proof. Follows from Theorem 16.
□

Theorem 23.

$$h \circ (\textsf{if } p \textsf{ then } f \textsf{ else } g) \quad = \quad \textsf{if } p \textsf{ then } (h \circ f) \textsf{ else } (h \circ g)$$

Proof. Follows from properties of coproducts.
□

A conditional based on the negation of a predicate is equivalent to one based on the original predicate, but with the branches swapped:

Theorem 24.

$$\textsf{if } (\textsf{not} \circ p) \textsf{ then } f \textsf{ else } g \quad = \quad \textsf{if } p \textsf{ then } g \textsf{ else } f$$

Proof. Follows from Theorem 18.
□

A conditional with two equal branches and a total predicate acts like one of the branches:

Theorem 25. *For total p,*

$$\textsf{if } p \textsf{ then } f \textsf{ else } f \quad = \quad f$$

Proof. Follows from Theorem 17.
□

A conditional with a constantly-true predicate acts like its first branch:

Theorem 26.

$$\mathsf{if}\ \mathsf{const.true}\ \mathsf{then}\ f\ \mathsf{else}\ g \quad = \quad f$$

Proof. Follows from Theorem 19.
□

Nested conditionals with total deterministic predicates 'abide' with each other:

Theorem 27. *For total and deterministic p and q,*

$$\begin{array}{l} \mathsf{if}\ p\ \mathsf{then}\ (\mathsf{if}\ q\ \mathsf{then}\ f\ \mathsf{else}\ g)\ \mathsf{else}\ (\mathsf{if}\ q\ \mathsf{then}\ h\ \mathsf{else}\ j) \\ \qquad = \ \mathsf{if}\ q\ \mathsf{then}\ (\mathsf{if}\ p\ \mathsf{then}\ f\ \mathsf{else}\ h)\ \mathsf{else}\ (\mathsf{if}\ p\ \mathsf{then}\ g\ \mathsf{else}\ j) \end{array}$$

Proof. We have

$$\begin{array}{cl}
 & \mathsf{if}\ p\ \mathsf{then}\ (\mathsf{if}\ q\ \mathsf{then}\ f\ \mathsf{else}\ g)\ \mathsf{else}\ (\mathsf{if}\ q\ \mathsf{then}\ h\ \mathsf{else}\ j) \\
= & \quad \{\ \text{conditionals}\ \} \\
 & (((f \triangledown g) \circ q?) \triangledown ((h \triangledown j) \circ q?)) \circ p? \\
= & \quad \{\ \text{pairs}\ \} \\
 & ((f \triangledown g) \triangledown (h \triangledown j)) \circ (q? + q?) \circ p? \\
= & \quad \{\ \text{Lemma 15}\ \} \\
 & ((f \triangledown g) \triangledown (h \triangledown j)) \circ (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (q? \vartriangle p) \\
= & \quad \{\ \text{Theorem 17; } q \text{ is total}\ \} \\
 & ((f \triangledown g) \triangledown (h \triangledown j)) \circ (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (q? \vartriangle (p \circ (\mathsf{id} \triangledown \mathsf{id}) \circ q?)) \\
= & \quad \{\ q? \text{ is deterministic}\ \} \\
 & ((f \triangledown g) \triangledown (h \triangledown j)) \circ (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle (p \circ (\mathsf{id} \triangledown \mathsf{id}))) \circ q? \\
= & \quad \{\ \text{guards}\ \} \\
 & ((f \triangledown g) \triangledown (h \triangledown j)) \circ (p \circ (\mathsf{id} \triangledown \mathsf{id}))? \circ q? \\
= & \quad \{\ (p \circ (\mathsf{id} \triangledown \mathsf{id}))? = ((\mathsf{inl} + \mathsf{inl}) \triangledown (\mathsf{inr} + \mathsf{inr})) \circ (p? + p?)\ \text{(see below)}\ \} \\
 & ((f \triangledown g) \triangledown (h \triangledown j)) \circ ((\mathsf{inl} + \mathsf{inl}) \triangledown (\mathsf{inr} + \mathsf{inr})) \circ (p? + p?) \circ q? \\
= & \quad \{\ \text{pairs}\ \} \\
 & ((f \triangledown h) \triangledown (g \triangledown j)) \circ (p? + p?) \circ q? \\
= & \quad \{\ \text{conditionals}\ \} \\
 & \mathsf{if}\ q\ \mathsf{then}\ (\mathsf{if}\ p\ \mathsf{then}\ f\ \mathsf{else}\ h)\ \mathsf{else}\ (\mathsf{if}\ p\ \mathsf{then}\ g\ \mathsf{else}\ j)
\end{array}$$

(Note that totality and determinism of p have not been used. In fact, the above calculation is valid even for partial and non-deterministic p and q, but the steps are a little harder to justify in that case.)

For the intermediate proof obligation, we have to show that

$$(p \circ (\mathsf{id} \triangledown \mathsf{id}))? \quad = \quad ((\mathsf{inl} + \mathsf{inl}) \triangledown (\mathsf{inr} + \mathsf{inr})) \circ (p? + p?)$$

The right-hand side of this is equal to

$$((\mathsf{inl} + \mathsf{inl}) \circ p?) \triangledown ((\mathsf{inr} + \mathsf{inr}) \circ p?)$$

because join fuses with coproduct, so by the universal property of joins it suffices to show that

$$\begin{array}{rcl} (p \circ (\mathsf{id} \triangledown \mathsf{id}))? \circ \mathsf{inl} & = & (\mathsf{inl} + \mathsf{inl}) \circ p? \\ (p \circ (\mathsf{id} \triangledown \mathsf{id}))? \circ \mathsf{inr} & = & (\mathsf{inr} + \mathsf{inr}) \circ p? \end{array}$$

For the first of these, we have:

$$\begin{array}{cl} & (p \circ (\mathsf{id} \triangledown \mathsf{id}))? \circ \mathsf{inl} \\ = & \quad \{ \text{ guards } \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{id} \vartriangle (p \circ (\mathsf{id} \triangledown \mathsf{id}))) \circ \mathsf{inl} \\ = & \quad \{ \ \mathsf{inl} \text{ is deterministic; pairs } \} \\ & (\mathsf{fst} + \mathsf{fst}) \circ \mathsf{distl} \circ (\mathsf{inl} \vartriangle p) \\ = & \quad \{ \text{ Lemma 15 } \} \\ & (\mathsf{inl} + \mathsf{inl}) \circ p? \end{array}$$

The second is symmetric.
□

6 Conclusions

Distributive categories are well-known in the category theory literature, as is the fact that coproducts in a distributive category can model conditionals. What is new in this paper, to the best of our knowledge, is the exploration and proof of the laws enjoyed by coproducts in this formulation.

One thing we have discovered in the process of writing this paper is that calculations involving distl are awkward, since they have to be performed indirectly in terms of undistl. There is probably no way around this, because there is no simpler point-free characterization of distl.

7 Acknowledgements

The author would like to thank Lambert Meertens and Robin Cockett for helpful discussions about distributive categories, conditionals and comonads, and Sue for patience while this paper was being born.

References

[1] Robin Cockett, *An Introduction to Distributive Categories.* Mathematical Structures in Computer Science, **1** (1991) p1–20.

[2] C. A. R. Hoare *et al*, *The Laws of Programming.* Communications of the ACM, **30** (1987) p672–686.