BRICS Summer School
on
Logical Methods

Categorical Logic and Type Theory

Roy L. Crole

University of Leicester, 2001

Contents

List of Tables

Preface

These notes accompanied five lectures given at the *EEF Foundations Summer School on Logical Methods* which took place at BRICS, Aarhus, Denmark, during June and July 2001. The notes provide an introduction to some aspects of categorical logic and type theory which may be of interest to (theoretical) computer scientists.

Categorical logic is the study of connections between formal logic and category theory. Categorical type theory is the study of connections between formal type systems and category theory. Logic concerns the study of reasoning, to produce consistent sets of propositions, often from some given assumptions. Formal logic concerns the use of mathematical languages to define both propositions, and methods for reasoning about propositions. In fact formal logic can be used to describe mathematical objects in terms of their constituent parts or internal structure; for example it provides a language for set theory, and we can describe a set by naming its elements. The notion of a type system pervades many programming languages. A type is essentially a collection of programs which have a similar structure or behaviour. A type system is a mathematical formalization of types, associated programs, and rules for reasoning about program behaviours. Category theory provides a very general language within which one can study mathematical objects via transformations of the objects, rather than the internal structure of the objects themselves; for example we might describe a set simply by stating that it is in bijection with another set. Here the "transformation" is a (particular kind of) function. Although category theory, logic, and type theory appear to be rather different subjects at first sight, they are in fact intimately connected. These notes give an introduction to some of the connections, very much geared towards computing science.

The notes are based around my book *Categories for Types*, although there is some new material here. In Part I, we cover background material which is used in Part II. We study order theory, category theory, and induction. Readers are assumed to be mathematically sophisticated, and have a rudimentary knowledge of (theoretical) computer science. Any reader familiar with these subjects may wish to go directly to Part II, referencing Part I as need arises. It is Part II which actually concerns the (main) subject matter of these notes, namely *Categorical Logic* and *Categorical Type Theory*. In these notes we barely scratch the surface of these subjects. However, there is probably plenty of material for five lectures! The logic we study is intuitionistic propositional calculus. This is a good starting point, and the categorical models are quite simple. From here, readers can go on to learn about the categorical treatment of predicate logics, although this is rather more difficult than the categorical treatment of propositional logic. The type theory we study is also quite elementary, as it does not involve any complicated quantifiers, and types are not dependent. Thus the semantics can be given in categories with quite simple structure (rather than requiring indexed or fibered categories). In

the final chapter on applications, we use categorical semantics to prove results about both the logic and the type theory which we have studied. We conclude with some suggestions for further reading.

©*Roy L. Crole, University of Leicester, UK, 2001.*

Part I

Preliminary Mathematics

1

Preorders and Posets

1.1 Ordered Sets

Discussion 1.1.1 *We begin with a summary of basic naive set theory. If A and X are sets, we write $A \subseteq X$ to mean A is a subset of X. A* ***total function*** *between a set X and a set Y is a subset $f \subseteq X \times Y$ for which given any $x \in X$ there is a unique $y \in Y$ such that $(x,y) \in f$. Given $x \in X$ we write $f(x)$ for the unique y such that $(x,y) \in f$. It will often be convenient to write $x \mapsto f(x)$ to indicate that $(x, f(x)) \in f$; for example, if $\mathbb{R}$ is the set of real numbers, then the function f between $\mathbb{R}$ and $\mathbb{R}$, given by $r \mapsto r^2$, is formally the subset*

$$\{(r,r^2) \mid r \in \mathbb{R}\} \subseteq \mathbb{R} \times \mathbb{R}.$$

Often we shall say that f is a function $X \to Y$ and write $f : X \to Y$ in place of $f \subseteq X \times Y$. We shall say (informally) that X and Y are the ***source*** *and* ***target*** *of the function f. A function $f : X \to X$ with identical source and target is called an* ***endofunction*** *on X. Given functions $f : X \to Y$ and $g : Y \to Z$, we write gf or $g \circ f$ for the function $X \to Z$ defined by $x \mapsto g(f(x))$. A* ***partial function*** *between X and Y is a subset $f \subseteq X \times Y$ such that given any elements $(x,y) \in f$ and $(x,y') \in f$ then $y = y'$. If $(x,y) \in f$, we write $f(x)$ for y. If $f : X \to Y$ is a partial function, and given $x \in X$ there is no $y \in Y$ for which $(x,y) \in f$, then we say that f is undefined at x, or sometimes simply say that $f(x)$ is undefined. If $f : X \to Y$ is a function and $S \subseteq X$ is a subset of X, then we shall sometimes use the notation $f(S)$ to represent the set $\{f(s) \mid s \in S\}$. If X and Y are any two sets, then the set $X \setminus Y \stackrel{\text{def}}{=} \{x \in X \mid x \notin Y\}$ is the* ***set difference*** *of Y from X. If X is a set, then $|X|$ will denote the* ***cardinality*** *(size) of X. A* ***binary relation*** *R on a set X is any subset $R \subseteq X \times X$. If $x,y \in X$, then we will write xRy for $(x,y) \in R$. R is* ***reflexive*** *if for any $x \in X$ we have xRx;* ***symmetric*** *if whenever $x,y \in X$ then xRy implies yRx;* ***transitive*** *if for any $x,y,z \in X$, whenever we have xRy and yRz then xRz; and* ***anti-symmetric*** *if whenever $x,y \in X$, xRy and yRx imply x and y are identical. R is an* ***equivalence relation*** *if it is reflexive, symmetric and transitive. Given an equivalence relation R on X, the* ***equivalence class*** *of $x \in X$ is the set $[x] \stackrel{\text{def}}{=} \{y \mid y \in X, xRy\}$. We write X/R for the set of equivalence classes $\{[x] \mid x \in X\}$. This completes the summary, and we now move on to the definition of ordered sets.*

A ***preorder*** *on a set X is a binary relation $\leq$ on X which is reflexive and transitive. The relation $\leq$ will sometimes be referred to informally as the* ***order relation*** *on the set X. It will sometimes be convenient to write $x \geq y$ for $y \leq x$. If at least one of $x \leq y$ and $y \leq x$ holds, then x and y are said to be* ***comparable****. If neither relation holds, then x and y are* ***incomparable****. A* ***preordered*** *set $(X, \leq)$ is a set equipped with a preorder, that is to say we are given a set (in this case X) along with a preorder $\leq$ on the set X; the set X is*

sometimes called the ***underlying*** *set of the preorder* $(X, \leq)$*. Where confusion cannot result, we refer to the preordered set* X*, or sometimes just the preorder* X*. The preorder* X *is said to be* ***discrete*** *if any two distinct elements of* X *are incomparable. If* $x \leq y$ *and* $y \leq x$ *then we shall write* $x \cong y$ *and say that* x *and* y *are* ***isomorphic*** *elements. Note that we can regard* $\cong$ *as a relation on* X*, which is in fact an equivalence relation. If* $(X, \leq_X)$ *is a preorder, we shall write* $S \subseteq X$ *to mean that the set* S *is a subset of the underlying set of* X*. Of course, we can regard* S *as a preordered set* $(S, \leq_S)$ *by restricting the order relation on* X *to* S*; more precisely, if* $s, s' \in S$*, then* $s \leq_S s'$ *iff* $s \leq_X s'$*. We shall then say that* S *has the* ***restriction order*** *inherited from* X*. However, we shall limit the force of the judgement* $S \subseteq X$ *to mean that* S *is simply a subset of the underlying set of* X*. The notation* $x \leq S$ *will mean that for each* $s \in S$*,* $x \leq s$*.*

A ***partial order*** *on a set* X *is a binary relation* $\leq$ *which is reflexive, transitive and anti-symmetric. A set* X *equipped with a partial order is called a* ***partially ordered*** *set, or sometimes a* ***poset****. If* $x, y \in X$*, where* X *is a poset, then we shall write* $x < y$ *to mean that* $x \leq y$ *and* $x \neq y$*. Given a preorder* X *then the set of equivalence classes* $X/\cong$ *can be given a partial ordering by setting* $[x] \leq [y]$ *iff* $x \leq y$ *for all* $x, y \in X$*. The poset* $X/\cong$ *is called the* ***poset reflection*** *of* X*.*

Examples 1.1.2

(1) The set of natural numbers, $\mathbb{N}$*, with the usual increasing order is a poset. We will refer to this poset as the* ***vertical natural numbers****.*

(2) The set $\{A \mid A \subseteq X\}$ *of subsets of a set* X *is often written as* $\mathcal{P}(X)$ *and is called the* ***powerset*** *of* X*. The powerset is a poset with order given by inclusion of subsets,* $A \subseteq B$*. The order is certainly anti-symmetric, for if* A *and* A' *are subsets of* X *where* $A \subseteq A'$ *and* $A' \subseteq A$*, then* $A = A'$*. Reflexivity and transitivity are clear.*

(3) Given preorders X *and* Y*, their* ***cartesian product*** *has underlying set*

$$X \times Y \stackrel{\text{def}}{=} \{(x, y) \mid x \in X, y \in Y\}$$

with order given ***pointwise****, that is* $(x, y) \leq (x', y')$ *iff* $x \leq x'$ *and* $y \leq y'$*.*

(4) If X *is a preorder, then* X^{op} *is the preorder with underlying set* X *and order given by* $x \leq^{op} y$ *iff* $y \leq x$ *where* $x, y \in X$*. We usually call* X^{op} *the* ***opposite*** *preorder of* X*. Of course any poset is certainly also a preorder.*

Discussion 1.1.3 *We now give some more definitions. Suppose that* X *is a preorder and* A *is a subset of* X*. An element* $x \in X$ *is an* ***upper bound*** *for* A *if for every* $a \in A$ *we have* $a \leq x$ *(or we can just write* $A \leq x$*, using the informal notation given in Discussion 1.1.1). An element* $x \in X$ *is a* ***lower bound*** *for* A *if* $x \leq A$*. An element* $x \in X$ *is a* ***greatest element*** *of* A *if it is an upper bound of* A *which belongs to* A*;* x *is a* ***least element*** *of* A *if it is a lower bound of* A *and belongs to* A*. We can prove a useful little result about greatest and least elements:*

Proposition 1.1.4 *Let X be a preordered set and A a subset of X. Then greatest and least elements of A are unique up to isomorphism if they exist.*

Proof Let a and a' be greatest elements of A. By definition, a is an upper bound of A, and also $a' \in A$. Hence $a' \leq a$. Similarly $a \leq a'$. Hence $a \cong a'$. The proof for least elements is essentially the same. □

Discussion 1.1.5 *The notions of upper bound, greatest element and so on give us mathematical tools for the description of the structure of preordered sets. The reader is probably familiar with the everyday notions of maximum and minimum, and our definitions of greatest and least elements correspond to such notions. Unfortunately, such ideas are not quite general enough for our purposes. We shall now define the concepts of meet and join which refine those of greatest and least elements.*

*Let X be a preordered set and $A \subseteq X$. A **join** of A, if such exists, is a least element in the set of upper bounds for A. A join is sometimes called the **least upper bound** or a **supremum**. A **meet** of A, if it exists, is a greatest element in the set of lower bounds for A. A meet is sometimes called the **greatest lower bound** or **infimum**. Note that meets and joins are defined as greatest and least elements; so from Proposition 1.1.4 we know that meets and joins are determined up to isomorphism if they exist. If the subset A has at least one join, then we will write $\bigvee A$ for a choice of one of the joins of A. Similarly, if the subset A has at least one meet, then we will write $\bigwedge A$ for a choice of one of the meets of A. If we wish to draw attention to the ordered set with respect to which a join and meet are being taken (in this case X) we shall write $\bigvee_X A$ and $\bigwedge_X A$ respectively. Note that the join is characterised by the property that for every $x \in X$ we have $\bigvee A \leq x$ iff $A \leq x$; this amounts to a formal statement that a join is by definition a least element in a set of upper bounds. Some points deserve special attention.*

• *Let X be a non-empty discrete preorder X, and $A \subseteq X$ a non-empty subset. Then A only has a meet or join if A is a singleton set. Clearly, for any $x \in X$, we have $\bigwedge\{x\} = x$ and $\bigvee\{x\} = x$.*

• *Consider the empty set, $\varnothing \subseteq X$. Then $\bigvee \varnothing$, if such exists, is written $\bot$ and is called a **bottom** of X. Note that a bottom element satisfies the property that for any $x \in X$ we have $\bot \leq x$. Similarly, $\bigwedge \varnothing$, if such exists, is written $\top$ and is called the **top** of X; it satisfies $x \in X$ implies $x \leq \top$.*

• *Consider a two element subset $\{a,b\} \subseteq X$. Write $a \vee b$ for $\bigvee\{a,b\}$ and call this a (binary) join of a and b. Similarly $a \wedge b$ is a (binary) meet of a and b. If we unravel the definitions, it can be seen that binary joins are characterised by the property that for every $x \in X$ we have $a \vee b \leq x$ iff $a \leq x$ and $b \leq x$; and binary meets by asking that for any $x \in X$ we have $x \leq a \wedge b$ iff $x \leq a$ and $x \leq b$.*

• *More generally, we may write $a_1 \wedge a_2 \wedge \ldots \wedge a_n$ for $\bigwedge\{a_1, a_2, \ldots, a_n\}$ when each $a_i \in X$, with a similar notation for joins.*

Exercise 1.1.6 *Make sure you understand the definition of meet and join in a preorder X. Think of some simple finite preordered sets in which meets and joins do not exist. Now suppose that X is a poset (and thus also a preorder). Show that meets and joins in a poset are unique if they exist.*

Example 1.1.7 *Given a set X, the powerset poset $\mathcal{P}(X)$ has all meets and joins. Meets are given by intersections and joins by unions. The top element is of course X and the bottom element $\varnothing$.*

Discussion 1.1.8 *We now turn our attention to notions of relations between preordered sets, and in particular to functional relations. If we talk of a function between the preordered sets X and Y we shall simply mean that we are given a function between the underlying sets. Such a function is said to be* ***monotone*** *if for $x,y \in X$ we have $x \leq y$ implies $f(x) \leq f(y)$; and* ***antitone*** *if $x \leq y$ implies $f(y) \leq f(x)$. We often refer to such a monotone function as a* ***homomorphism of preorders****. Roughly one thinks of a homomorphism as a function which preserves structure; in the case of a preorder, this structure is just the order relation. A monotone function may alternatively be called an* ***order preserving*** *function. f is said to* ***reflect order*** *if given any $x,y \in X$, $f(x) \leq f(y)$ implies $x \leq y$. The posets X and Y are* ***isomorphic*** *if there are monotone functions $f : X \to Y$ and $g : Y \to X$ for which $gf = id_X$ and $fg = id_Y$. The monotone function g is an* ***inverse*** *for f; and likewise f is an inverse for g. We say that f is an* ***isomorphism*** *if such an inverse g exists. The set $X \Rightarrow Y$ is defined to have elements the monotone functions with source X and target Y, that is functions $X \to Y$. This set can be regarded as a preorder by defining a relation $f \leq g$ iff given any $x \in X$ we have $f(x) \leq g(x)$, where $f,g : X \to Y$. This ordering is often referred to as the* ***pointwise*** *order. We have the following proposition:*

Proposition 1.1.9 *The identity function on any preordered set is monotone, and the composition of two monotone functions is another monotone function. Now let X, Y and Z be preordered sets. The composition function*

$$\circ : (Y \Rightarrow Z) \times (X \Rightarrow Y) \to (X \Rightarrow Z)$$

sending the pair $(g, f) \in (Y \Rightarrow Z) \times (X \Rightarrow Y)$ to $gf \in X \Rightarrow Z$ is itself a monotone function between preordered sets. Finally, any function $f : X \times Y \to Z$ is monotone iff it is monotone in each variable separately, which is to say that given any $x,x' \in X$ and $y,y' \in Y$, then

$$x \leq x' \text{ implies } f(x,y) \leq f(x',y)$$

and

$$y \leq y' \text{ implies } f(x,y) \leq f(x,y').$$

Proof Follows by a routine manipulation of the definitions. □

Examples 1.1.10

(1) Let $f : X \to Y$ be a set-theoretic function between sets X and Y. Then there is a monotone function $f^{-1} : \mathcal{P}(Y) \to \mathcal{P}(X)$ where given $B \subseteq Y$ we define

$$f^{-1}(B) \stackrel{\text{def}}{=} \{x \in X \mid f(x) \in B\}$$

(2) Take $\mathbb{R}$ to be the set of reals with their usual ordering. Then the function $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x) = x^3$ is monotone.

Exercises 1.1.11

(1) Complete the proof of Proposition 1.1.9.

(2) Let X and Y be preorders and $X \times Y$ their cartesian product. Check that there are monotone functions $\pi_X : X \times Y \to X$, $(x,y) \mapsto x$ and $\pi_Y : X \times Y \to Y$, $(x,y) \mapsto y$ where $(x,y) \in X \times Y$. Now verify that given monotone functions $f : Z \to X$ and $g : Z \to Y$ where Z is any given preorder, there is a unique monotone function $m : Z \to X \times Y$ for which $f = \pi_X m$ and $g = \pi_Y m$.

(3) Find a counterexample to the following statement. A monotone function $f : X \to Y$ between posets X and Y which is a bijection is necessarily an isomorphism.

(4) Let X be a poset and define a relation on the set X by saying that $x \prec y$ just in case $x < y$ and there is no $z \in X$ for which $x < z < y$. Now let X be any set and Y be a poset. Let $X \Rightarrow Y$ be the poset of functions $X \to Y$ ordered pointwise. Show that $f \prec g$ (where $f, g \in X \Rightarrow Y$) iff

(a) There is $\hat{x} \in X$ for which $f(\hat{x}) \prec g(\hat{x})$ in Y, and
(b) $f(x) = g(x)$ for each $x \in X \setminus \{\hat{x}\}$.

*Now let X be a finite poset, and $X \Rightarrow Y$ the poset of **monotone** functions $X \to Y$. Show that $f \prec g$ iff (a) and (b) remains true, with this new definition of $X \Rightarrow Y$.*

1.2 Basic Lattice and Prelattice Theory

Remark 1.2.1 *Throughout the remainder of this chapter, note that all definitions, results and proofs are presented for structured ordered sets in which the order is a partial order. If any occurrence of an equality sign $=$ is replaced with an isomorphism sign $\cong$ then a definition, result or proof remains valid for the same structure but now with a preorder. Further, anything which is unique with respect to a partial order will be unique up to isomorphism with respect to a preorder.*

Discussion 1.2.2 *In this section we shall describe some examples of posets which have additional structure and which feature in concrete examples of models of logics and type theories. For each kind of poset we shall give its formal definition, some examples, and certain elementary theorems which will give further examples.*

*A **lattice** is a poset which has finite meets and joins, that is, meets and joins of finite subsets. A **complete lattice** is a poset which has arbitrary meets and joins. A **prelattice** is a preordered set which has finite meets and joins. A **complete prelattice** is a preordered set which has arbitrary meets and joins.*

From now, definitions are stated only in terms of lattices—bear in mind Remark 1.2.1.

*We shall want to consider functional relations between such structures; those we consider will usually preserve structure and are known in general as **homomorphisms**. Thus a **homomorphism of lattices** is a function $f : X \to Y$ (with X and Y lattices) which preserves finite meets and joins, that is*

$$f(\bigwedge\{x_1,\ldots,x_n\}) = \bigwedge\{f(x_1),\ldots,f(x_n)\}$$

and

$$f(\bigvee\{x_1,\ldots,x_n\}) = \bigvee\{f(x_1),\ldots,f(x_n)\}$$

and also $f(\top) = \top$ and $f(\bot) = \bot$. Note that such a function is automatically monotone.

Remark 1.2.3 *Note that both lattices and complete lattices have top and bottom elements (being the meet and join of the empty subset respectively). Also, having finite meets and joins is equivalent to having binary meets and joins and top and bottom elements—see Lemma 1.2.5. Finally, because $x \wedge y = x$ and $x \vee y = y$ just in case $x \leq y$ in a poset X, one only needs to check the existence of binary meets and joins of non-comparable pairs of elements of X to see that X is a lattice (as well as checking that X has top and bottom elements).*

Examples 1.2.4

(1) If we adjoin a top and bottom to the set $\mathbb{Q}$ of rational numbers to obtain $\mathbb{Q}^* \stackrel{\text{def}}{=} (\mathbb{Q} \cup \{\infty\} \cup \{-\infty\}, \leq)$ where for every $q \in \mathbb{Q}$ we have $-\infty \leq q \leq \infty$, then $\mathbb{Q}^*$ is a lattice, but not a complete lattice. If we likewise add a top and bottom to the set $\mathbb{R}$ of reals to obtain $\mathbb{R}^* \stackrel{\text{def}}{=} (\mathbb{R} \cup \{\infty\} \cup \{-\infty\}, \leq)$ where for every $r \in \mathbb{R}$ we have $-\infty \leq r \leq \infty$, then $\mathbb{R}^*$ is a complete lattice due to the completeness axiom for $\mathbb{R}$. Similarly any closed interval $[r_1, r_2] \subseteq \mathbb{R}$ is a complete lattice when its order is inherited from $\mathbb{R}$.

(2) The power set $\mathcal{P}(X)$ of a set X is a complete lattice with the inclusion order, with meets and joins given by intersection and union. The top element is X and the bottom element is $\varnothing$.

(3) The topped vertical natural numbers $\mathbb{N} \cup \{\infty\}$ form a complete lattice.

(4) If X and Y are (complete) lattices, their binary ***cartesian product*** *is the set $X \times Y$ of pairs of elements, ordered pointwise. Then $X \times Y$ is a (complete) lattice with meets and joins calculated pointwise. In a similar way we can consider the cartesian product of a finite number (say n) of lattices $X_1 \times \ldots \times X_n$.*

(5) Let S be any set and X a poset. Then the set of functions $S \Rightarrow X$ with source S and target (the underlying set of) X is a (complete) lattice whenever X is a (complete) lattice, with $S \Rightarrow X$ ordered pointwise. It is easy to see that meets and joins are calculated ***pointwise****. For example, if $f, g \in S \Rightarrow X$ then for any $x \in X$ we have $(f \wedge g)(x) = f(x) \wedge g(x)$.*

The next lemma lists some useful elementary facts about lattices.

Lemma 1.2.5 *Any lattice X satisfies the following laws, where x, y and z are taken to be any elements of X.*

(i) $x \wedge x = x = x \vee x$ (idempotency).

(ii) $x \wedge y = y \wedge x$ and $x \vee y = y \vee x$ (commutativity).

(iii) $x \wedge (y \wedge z) = (x \wedge y) \wedge z$ and $x \vee (y \vee z) = (x \vee y) \vee z$ (associativity).

(iv) The existence of binary meets (binary joins) is the same as the existence of (non-empty) finite meets (finite joins). In particular,

$$\bigwedge \{x_1, \ldots, x_n\} = (\ldots((x_1 \wedge x_2) \wedge x_3)\ldots) \wedge x_n$$

with a similar statement for joins.

(v) $x \wedge (x \vee y) = x \vee (x \wedge y) = x$ (absorption).

(vi) $x \leq y$ iff either $x \wedge y = x$ or $x \vee y = y$.

(vii) If $y \leq z$ then $x \wedge y \leq x \wedge z$ and $x \vee y \leq x \vee z$ (monotonicity).

(viii) $(x \wedge y) \vee (x \wedge z) \leq x \wedge (y \vee z)$ and $x \vee (y \wedge z) \leq (x \vee y) \wedge (x \vee z)$.

(ix) $x \leq z$ implies $x \vee (y \wedge z) \leq (x \vee y) \wedge z$.

(x) $x \cong x'$ and $y \cong y'$ implies $x \wedge y \cong x' \wedge y'$.

Proof The proof is an easy application of the properties of meets and joins. For example, to prove (viii), note that $y \leq x \vee y$ implies $y \wedge z \leq (x \vee y) \wedge z$; and $x \leq x \vee y$ and the hypothesis imply $x \leq (x \vee y) \wedge z$. □

Discussion 1.2.6 *We end this section by giving some simple results concerning the distributive lattices. These lattices have pleasant properties which aid the manipulation of meets and joins. Let X be a lattice. Then X is* ***distributive*** *if it satisfies $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$ for all x, y, z in X.*

Remark 1.2.7 *From Lemma 1.2.5, parts (vii) and (viii), we see that in order to check that a lattice is distributive it is only necessary to check that an inequality holds.*

Lemma 1.2.8 *For any lattice X the following conditions are equivalent:*

(i) For every $x, y, z \in X$ we have $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$,

(ii) For every $x, y, z \in X$ we have $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$,

(iii) For every $x, y, z \in X$ we have $x \wedge (y \vee z) \leq (x \wedge y) \vee z$.

Proof Throughout we use the results of Lemma 1.2.5. If (i) holds then

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z) \leq (x \wedge y) \vee z$$

which is (iii). If (iii) holds, then we have

$$x \wedge (y \vee z) \leq x \wedge [(x \wedge y) \vee z] \leq (x \wedge z) \vee (x \wedge y), \qquad (*)$$

namely one half of (i). To see $(*)$, note that by using monotonicity of $x \wedge (-)$ on the hypothesis (iii) we obtain the first inequality. We can also use (iii) to deduce that

$$\text{For every } x, y, z \in X \text{ we have } x \wedge (y \vee z) \leq (x \wedge z) \vee y.$$

and then use this deduction to obtain the second inequality of $(*)$. The other half of (i) holds via Lemma 1.2.5. That (ii) iff (iii) is a similar argument. □

Examples 1.2.9

(1) The poset $\mathcal{P}(X)$ is distributive. For if A, B and C are subsets of X, it follows from simple naive set theory that $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

(2) The complete lattice $(\mathbb{N}, |)$ is distributive.

(3) If X is a distributive lattice then $S \Rightarrow X$, the functions from S to X with the pointwise order, is a distributive lattice.

1.3 Boolean and Heyting Lattices and Prelattices

Discussion 1.3.1 *In this section we introduce the notions of Boolean and Heyting lattices (and prelattices). A Boolean lattice is, roughly, a distributive lattice for which there is an operation on elements which mimics the notion of complementation in a powerset (recall that if $A \in \mathcal{P}(X)$ then the complement of A in $\mathcal{P}(X)$ is the set difference $X \setminus A$ of A from X). Readers may have met the notion of Boolean lattice from undergraduate or even school circuit theory, but this will not concern us here.*

Take a lattice X and let $x \in X$. An element $a \in X$ satisfying $a \wedge x = \bot$ and $a \vee x = \top$ is called a ***complement*** *of x. The next proposition shows that, in a distributive lattice, such complements are unique.*

Proposition 1.3.2 *Let X be a distributive lattice and $x, y, z \in X$. Then there is at most one $a \in X$ for which $x \wedge a = y$ and $x \vee a = z$.*

Proof Suppose that $a' \in X$ also satisfies the hypotheses. We have

$$\begin{aligned} a &= a \wedge (a \vee x) \\ &= a \wedge z \\ &= a \wedge (x \vee a') \\ &= (a \wedge x) \vee (a \wedge a') \\ &= y \vee (a \wedge a'). \end{aligned}$$

But $y \leq a$ and $y \leq a'$ and so $a = a \wedge a'$. Similarly $a' = a \wedge a'$ and so $a = a'$. □

Discussion 1.3.3 *A* ***Boolean lattice*** *X is a distributive lattice, which also has complements of all elements. We write $\neg x$ for the complement of x where $x \in X$. Note that it makes sense to talk of* ***the*** *complement of x, because Proposition 1.3.2 implies that complements are unique if they exist. The next lemma states some laws which are true of all Boolean lattices, and are analogues of the well known De Morgan rules which hold for powersets.*

Lemma 1.3.4 *In a Boolean lattice X, we have for all $x, y \in X$,*

(i) $\neg\neg x = x$,

(ii) $\neg(x \wedge y) = \neg x \vee \neg y$, *and*

(iii) $\neg(x \vee y) = \neg x \wedge \neg y$.

Proof Note that both $\neg\neg x$ and x are complements of the element $\neg x$ in X. Then use Proposition 1.3.2 to deduce that (i) holds. There is of course a well defined function $\neg : X \to X$ (and (i) shows that it is a bijection). In fact this function is antitone. To

prove this, one shows that if $x \leq y$ in X then $\neg y = \neg y \wedge \neg x$, by showing that $\neg y \wedge \neg x$ is a complement of y and thus equals $\neg y$. But $\neg y = \neg y \wedge \neg x$ implies $\neg y \leq \neg x$, as required. Then (ii) and (iii) follow from the fact that $\neg$ is an antitone bijective endofunction on X. □

Examples 1.3.5

(1) Let X be a set and define, for $A \subseteq X$, $\neg A \stackrel{\text{def}}{=} X \setminus A$. Then $\mathcal{P}(X)$ is a Boolean lattice. (Recall that $X \setminus A$ is the set difference of A from X).

(2) The collection of finite and cofinite subsets of a set X,

$$\{A \subseteq X \mid A \text{ is finite or } X \setminus A \text{ is finite}\}$$

is a Boolean lattice in which $\neg A \stackrel{\text{def}}{=} X \setminus A$.

(3) The subsets of a topological space X which are both open and closed is a Boolean lattice in which $\neg A \stackrel{\text{def}}{=} X \setminus A$, where A is a closed and open subset of X.

*(4) Let us write $\mathbb{B}$ for the poset with a two point underlying set $\{\bot, \top\}$ for which $\bot \leq \top$. We call this the **Sierpinski** poset. Then the cartesian product of a finite number of copies of $\mathbb{B}$, $\mathbb{B} \times \ldots \times \mathbb{B}$, is a Boolean lattice.*

Discussion 1.3.6 *A **Heyting lattice** X is a lattice in which for each pair of elements $y, z \in X$ there is an element $y \Rightarrow z \in X$ such that*

$$x \leq y \Rightarrow z \qquad \textit{iff} \qquad x \wedge y \leq z.$$

*We call $y \Rightarrow z$ the **Heyting implication** of y and z.*

Lemma 1.3.7 *In a Heyting lattice X, the Heyting implication of y and z is unique.*

Proof Suppose that a and a' are two candidates for the element $y \Rightarrow z \in X$. Then $a \leq a$ implies $a \wedge y \leq z$ implies $a \leq a'$; the converse is similar. □

Proposition 1.3.8 *Every Boolean lattice is a Heyting lattice.*

Proof Let X be a Boolean lattice, $x, y \in X$, and define $x \Rightarrow y \stackrel{\text{def}}{=} \neg x \vee y$. Then $z \leq \neg x \vee y$ implies

$$\begin{aligned} z \wedge x &\leq (\neg x \vee y) \wedge x \\ &= (\neg x \wedge x) \vee (y \wedge x) \\ &= \bot \vee (y \wedge x) \\ &\leq y, \end{aligned}$$

and $z \wedge x \leq y$ implies $\neg x \vee y \geq \neg x \vee (z \wedge x) \geq (\neg x \vee z) \wedge (\neg x \vee x) \geq z$. □

Proposition 1.3.9

(i) A Heyting lattice X is distributive.

(ii) Any finite distributive lattice X is a Heyting lattice.

(iii) A Heyting lattice X is a Boolean lattice iff for all $x \in X$, $\neg\neg x = x$, where we define $\neg x \stackrel{\text{def}}{=} x \Rightarrow \bot$.

Proof

(i) Recall Lemma 1.2.5 which tells us that $(x \wedge y) \vee (x \wedge z) \leq x \wedge (y \vee z)$. Then note that

$$x \Rightarrow ((x \wedge y) \vee (x \wedge z)) \geq (x \Rightarrow (x \wedge y)) \vee (x \Rightarrow (x \wedge z)) \geq y \vee z$$

which amounts to $(x \wedge y) \vee (x \wedge z) \geq x \wedge (y \vee z)$, and so X is distributive.

(ii) Take x and y and define $x \Rightarrow y \stackrel{\text{def}}{=} \bigvee \{u \in X \mid u \wedge x \leq y\}$. Then $z \wedge x \leq y$ implies $z \leq x \Rightarrow y$ is immediate. If $z \leq x \Rightarrow y$ then

$$z \wedge x \leq \bigvee \{u \wedge x \mid u \wedge x \leq y\} \leq y$$

follows from distributivity (and the finiteness of X).

(iii) $(\Rightarrow)$ This way is part (i) of Lemma 1.3.4.

$(\Leftarrow)$ Conversely, we have just seen that X is distributive, and

$$x \wedge \neg x = x \wedge (x \Rightarrow \bot) = \bot$$

is trivial. There is a well defined function $\neg : X \to X$ given by $x \mapsto \neg x$ and it is easy to check that it is antitone and bijective. Thus it follows that De Morgan's rules (ii) and (iii) of Proposition 1.3.4 hold (but here in the Heyting lattice X). Hence $\top = \neg(x \wedge \neg x) = \neg x \vee x$ as required.

□

Examples 1.3.10

(1) Let X be a chain with top and bottom elements—by definition a chain is a poset in which any two elements are comparable. Thus X is a lattice in which meet and join are given by greatest and least elements. Then X is Heyting with

$$x \Rightarrow y \;=\; \begin{cases} \top \text{ if } x \leq y \\ y \text{ otherwise} \end{cases}$$

where $x, y \in X$. Note that X is not a Boolean lattice, for $\neg\neg x = \top$ for all $x \in X$: see Proposition 1.3.9.

Exercises 1.3.11

(1) Let X be any poset with finite meets. Prove that X is a Boolean lattice iff for all $x \in X$, there is $\overline{x} \in X$ such that for all $y \in X$ we have

$$x \leq y \qquad \textit{iff} \qquad x \wedge \overline{y} = b$$

for some fixed $b \in X$.

(2) Let X be a Heyting lattice, and for each $x \in X$ make the definition $\neg x \stackrel{\text{def}}{=} x \Rightarrow \bot$. Prove that for any $x,y \in X$, $\neg(x \vee y) = \neg x \wedge \neg y$ and $\neg\neg(x \wedge y) = \neg\neg x \wedge \neg\neg y$.

2

Category Theory

2.1 Introduction

Discussion 2.1.1 *A category consists of a pair of collections, namely a collection of "structures" together with a collection of "relations between the structures." Let us illustrate this with some informal examples of categories.*

• *The collection of all sets (thus each set is an example of one of the structures referred to above), together with the collection of all set-theoretic functions (the functions are the relations between the structures).*

• *The collection of all posets (each poset is a structure), together with all monotone functions (the monotone functions are the relations between the structures).*

• *The collection of all finite dimensional vector spaces, together with all linear maps.*

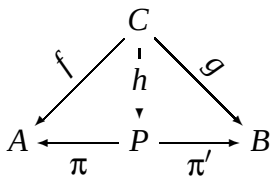
• *The set of real numbers $\mathbb{R}$ (in this case each structure is just a real number $r \in \mathbb{R}$), together with the relation of order $\leq$ on the set $\mathbb{R}$. Thus given two structures $r, r' \in \mathbb{R}$, there is a relation between them just in case $r \leq r'$.*

• *More generally, any set X, together with a preorder.*

All categories have this basic form, that is, consist of structures and relations between the structures: the structures are usually referred to as the objects of the category and the relations between the structures as morphisms. It is important to note that the objects of a category do not have to be sets (in the fourth example they are real numbers) and that the morphisms do not have to be functions (in the fourth example they are instances of the order relation $\leq$). Of course, there are some precise rules which define exactly what a category is, and we shall come to these shortly: the reader may care to look at the definition of a category given in Discussion 2.2.1 while also reading the remainder of this introduction. For the time being we continue with a broad discussion of the aims of category theory, that is, the general study of categories. Category theory looks at properties which are common to different categories. It is often the case that the specification of a property of a category can be set out in very general terms, but that the implementation of this property in particular categories varies greatly. Let us look at an example. We have said that the collection of all sets and functions forms a category; consider the following property of this "category:"

(Property CP*) Given any two sets A and B, then there is a set P and functions $\pi : P \to A$, $\pi' : P \to B$ such that the following condition holds: given any functions $f : C \to A$, $g : C \to B$*

with C any set, then there is a unique function $h : C \to P$ making the diagram



commute. End of definition of (Property CP*).*

Let us investigate an instance of (Property CP*) in the case of two given sets A and B. Suppose that $A \stackrel{\text{def}}{=} \{a,b\}$ and $B \stackrel{\text{def}}{=} \{c,d,e\}$. Let us take P to be $A \times B \stackrel{\text{def}}{=} \{(x,y) \mid x \in A, y \in B\}$ and the functions π and π' to be coordinate projection to A and B respectively, and see if (P,π,π') makes the instance of (Property* CP*) for the given A and B hold. Let C be any other set and $f : C \to A$ and $g : C \to B$ be any two functions. Define the function $h : C \to P$ by $z \mapsto (f(z),g(z))$. We leave the reader to verify that indeed $f = \pi h$ and $g = \pi' h$, and that h is the only function for which these equations hold with the given f and g. Now define $P' \stackrel{\text{def}}{=} \{1,2,3,4,5,6\}$ along with functions $p : P' \to A$ and $q : P' \to B$ where*

$$\begin{array}{llll} p(1), & p(2), & p(3) = a & \qquad\qquad q(1), \quad q(4) = c \\ p(4), & p(5), & p(6) = b & \qquad\qquad q(2), \quad q(5) = d \\ & & & \qquad\qquad q(3), \quad q(6) = e \end{array}$$

In fact (P',p,q) also makes the instance of (Property CP*) for the given A and B hold true. To see this, one can check by enumerating six cases that there is a unique function $h : C \to P'$ for which $f = ph$ and $g = qh$ (for example, if $x \in C$ and $f(x) = a$ and $g(x) = d$ then we must have $h(x) = 2$, and this is one case).*

Now notice that there is a bijection between P (the cartesian product

$$\{(a,c),(a,d),(a,e),(b,c),(b,d),(b,e)\}$$

of A and B) and P'. In fact any choices for the set P can be shown to be bijective. It is very often useful to determine sets up to bijection rather than worry about their elements or "internal make up," so we might consider taking (Property CP*) as a definition of cartesian product of two sets and think of the P and P' in the example above as two implementations of the notion of cartesian product of the sets A and B. Of course (Property* CP*) only makes sense when talking about the collection of sets and functions; we can give a definition of cartesian product for an arbitrary category which becomes (Property* CP*) for the "category" of sets and functions.*

Category theory looks at properties enjoyed by categories which may be described using an abstract definition of a category rather than particular examples of categories. Such a property is called a categorical property. Very roughly, such properties depend on the external behaviour of objects in categories, and not the internal make up of the objects. For example, if we take an instance of (Property CP*) for sets A and B as the new definition of the*

cartesian product of A and B, then we have shifted our viewpoint away from the structures of our category (sets) and towards the relations between the structures (functions). The traditional definition of cartesian product is given in terms of the elements of the sets A and B, and the new definition is given solely in terms of functions involving A and B. It is the emphasis of relations between objects in categories, rather than the objects themselves, which allows us to make definitions which do not (explicitly) depend on the internal make up of the objects.

Now that we have painted an informal picture of a category, and described the idea of a categorical property, we can move on to an account of the contents of Chapter 2. We begin with a formal definition of a category and give a number of examples of the concept. Next, the notion of functor is given, which is a mapping between categories. We give examples of functors and introduce a little more notation. This is followed by the definition of natural transformation; such a gadget can be thought of as a mapping between functors. The concepts of category, functor and natural transformation are the three most basic notions of category theory. Using these three ideas, we give an account of ways of regarding two categories as being "essentially the same," namely isomorphism and equivalence. These notions of similarity are based on the idea that two bijective sets are similar, as are two isomorphic groups. We follow this with the Yoneda lemma, which will turn out to be a very useful tool indeed when discussing the semantics of type theories. We give an account of cartesian closed categories, which have some structure similar to that found in the category of sets and functions, and which will give us categorical models of type theory.

2.2 Categories and Examples

Discussion 2.2.1 *We begin with a definition of a category. A* ***category*** $\mathcal{C}$ *is specified by the following data:*

• *A collection* $ob\,\mathcal{C}$ *of entities called* ***objects****. An object will often be denoted by a capital letter such as A, B, C...*

• *A collection* $mor\,\mathcal{C}$ *of entities called* ***morphisms****. A morphism will often be denoted by a small letter such as* f*,* g*,* h*...*

• *Two operations assigning to each morphism* f *its* ***source*** $src(f)$ *which is an object of* $\mathcal{C}$ *and its* ***target*** $tar(f)$ *also an object of* $\mathcal{C}$*. We shall write* $f : src(f) \longrightarrow tar(f)$ *to indicate this, or perhaps* $f : A \to B$ *where* $A = src(f)$ *and* $B = tar(f)$*. Sometimes we shall just say "let* $f : A \to B$ *be a morphism of* $\mathcal{C}$*" to mean* f *is a morphism of* $\mathcal{C}$ *with source A and target B.*

• *Morphisms* f *and* g *are* ***composable*** *if* $tar(f) = src(g)$*. There is an operation assigning to each pair of composable morphisms* f *and* g *their* ***composition*** *which is a morphism denoted by* $g \circ f$ *or just* gf *and such that* $src(gf) = src(f)$ *and* $tar(gf) = tar(g)$*. So for example, if* $f : A \to B$ *and* $g : B \to C$*, then there is a morphism* $gf : A \to C$*. There is also*

an operation assigning to each object A of $\mathcal{C}$ an ***identity*** *morphism $id_A : A \to A$. These operations are required to be* ***unitary***

$$\begin{aligned} id_{tar(f)} \circ f &= f \\ f \circ id_{src(f)} &= f \end{aligned}$$

and ***associative****, that is given morphisms $f : A \to B$, $g : B \to C$ and $h : C \to D$ then*

$$(hg)f = h(gf).$$

It is time to give some examples of categories. We adopt the convention of using calligraphic letters to denote categories, using an abbreviation of the names of the objects of the category, or occasionally the morphisms of a category. This convention should become clear with the following examples.

Examples 2.2.2

(1) The category of sets and total functions, $\mathcal{S}et$. The objects of the category are sets and the morphisms are triples (A, f, B) where A and B are sets and $f \subseteq A \times B$ is a subset of the cartesian product of A and B giving rise to a total function. The source and target operations are defined by $src(A, f, B) \stackrel{\text{def}}{=} A$ and $tar(A, f, B) \stackrel{\text{def}}{=} B$. Suppose that we have another morphism (B, g, C). Then $tar(A, f, B) = src(B, g, C)$, and the composition is given by

$$(B, g, C) \circ (A, f, B) = (A, gf, C)$$

where gf is the usual composition of the functions f and g. Finally, if A is any set, the identity morphism assigned to A is given by (A, id_A, A) where $id_A \subseteq A \times A$ is the identity function. We leave the reader to check that composition is an associative operation and that composition by identities is unitary. Informally, the morphisms of $\mathcal{S}et$ are functions in the usual set theoretic sense together with a ***specified*** *source and target. From now on we shall not give such a formal account of our examples of categories.*

(2) The category of sets and partial functions, $\mathcal{P}art$. The objects are sets and the morphisms are partial functions equipped with a specified source and target. The definition of composition is the expected one, namely given $f : A \to B$, $g : B \to C$, then for each element a of A, $gf(a)$ is defined with value $g(f(a))$ if both $f(a)$ and $g(f(a))$ are defined, and is otherwise not defined.

(3) Any preordered set $(X, \leq)$ may be viewed as a category. Recall from Section 1.1 that a preorder $\leq$ on a set X is a reflexive, transitive relation on X. The objects are the elements of the set X and the morphisms instances of the order relation; formally the collection of morphisms is the set of pairs of the form (x, y) where $x, y \in X$ and $x \leq y$. $(X, \leq)$ forms a category with identity morphisms (x, x) for each object x (because $\leq$ is reflexive) and composition $(y, z) \circ (x, y) \stackrel{\text{def}}{=} (x, z)$ (because $\leq$ is transitive). Note for x and y elements of X, there is at most one morphism from x to y according to whether $x \leq y$ or not.

(4) A ***discrete*** *category is one for which the only morphisms are identities. So a very simple example of a discrete category is given by regarding any set as a category in which the objects are the elements of the set, there is an identity morphism for each element, and there are no other morphisms.*

(5) Given a category $\mathcal{C}$, we may define the ***opposite category*** *$\mathcal{C}^{op}$. The collection of objects of $\mathcal{C}^{op}$ is the same as the collection of objects of $\mathcal{C}$. The collection of morphisms of $\mathcal{C}^{op}$ is the same as the collection of morphisms of $\mathcal{C}$. If f is a morphism of $\mathcal{C}^{op}$ (and thus by definition a morphism of $\mathcal{C}$), then the source $src(f)$ of f in $\mathcal{C}^{op}$ is defined to be the target $tar(f)$ of f in $\mathcal{C}$. Also, the target of f in $\mathcal{C}^{op}$ is the source of f in $\mathcal{C}$. (Thus $f : A \to B$ is a morphism in $\mathcal{C}^{op}$ just in case $f : B \to A$ is a morphism in $\mathcal{C}$). The identity on an object A in $\mathcal{C}^{op}$ is defined to be id_A in $\mathcal{C}$. Finally we need to define composition in $\mathcal{C}^{op}$. If $f : A \to B$ and $g : B \to C$ are morphisms in $\mathcal{C}^{op}$, then $f : B \to A$ and $g : C \to B$ are morphisms in $\mathcal{C}$. Hence f and g are composable in $\mathcal{C}$, with composition $f \circ g : C \to A$. We define the composition of f and g in $\mathcal{C}^{op}$ to be the morphism $f \circ g$.*

(6) If we are given a preorder X which we regard as a category, then the opposite category X^{op} is precisely the opposite preorder of X.

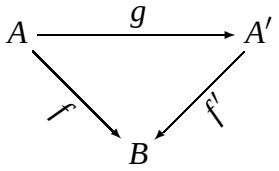
(7) The category $\mathcal{P}re\mathcal{S}et$ has objects preorders and morphisms the monotone functions; the category $\mathcal{P}O\mathcal{S}et$ has objects posets and morphisms the monotone functions.

(8) The category of lattices $\mathcal{L}at$ has objects lattices and morphisms the lattice homomorphisms.

(9) The category $\mathcal{H}Lat$ has objects the Heyting lattices and morphisms Heyting lattice homomorphisms. A Heyting lattice homomorphism $f : X \to Y$ between Heyting lattices is a function preserving finite meets and joins and preserving Heyting implications. So if $x, x' \in X$, then

$$f(x \Rightarrow x') = f(x) \Rightarrow f(x').$$

(10) Let $\mathcal{C}$ be a category and B an object of $\mathcal{C}$. The ***slice of*** *$\mathcal{C}$* ***by*** *B, denoted by $\mathcal{C}/B$, is the category whose objects are morphisms $f : A \to B$ in $\mathcal{C}$, and whose morphisms $g : f \to f'$ are those morphisms $g : src(f) \to src(f')$ in $\mathcal{C}$ for which $f = f'g$. It is often helpful to view such objects and morphisms as a commutative diagram:*



Similarly we can define the ***coslice****, $B/\mathcal{C}$. This has objects which are morphisms $f : B \to A$ in $\mathcal{C}$ and morphisms $g : f \to f'$ are morphisms $g : tar(f) \to tar(f')$ in $\mathcal{C}$ for which $f' = gf$.*

(11) Suppose that $\mathcal{C}$ and $\mathcal{D}$ are categories. Then the ***product*** *of $\mathcal{C}$ and $\mathcal{D}$, $\mathcal{C} \times \mathcal{D}$, has objects pairs (C, D) where C is an object of $\mathcal{C}$ and D an object of $\mathcal{D}$. The morphisms*

$(C,D) \to (C',D')$ are pairs of morphisms (f,g) where $f : C \to C'$ and $g : D \to D'$ with composition given (as expected) coordinatewise.

(12) Any monoid (M,b,e) yields a category where there is one object M, the morphisms are elements of the monoid, and composition and identity arise from the monoid operations. So if m and m' are elements of M and hence morphisms $m,m' : M \to M$, then their composition is $b(m,m') : M \to M$.

(13) The category $\mathcal{M}$on has objects consisting of all monoids, and morphisms which are functions preserving the monoid multiplication. Thus a monoid morphism f between monoids M and M' is a function $f : M \to M'$ between the underlying sets, for which $f(e) = e$ and $f(mn) = f(m)f(n)$ where m and n are elements of M.

Exercise 2.2.3 *Understand how each of the informal descriptions of categories given in Examples 2.2.2 can be formalised to fit the definition of a category given in Discussion 2.2.1.*

2.3 Functors and Examples

Discussion 2.3.1 *A function $f : X \to Y$ can be thought of as a relation between two sets. We can also think of the function f as specifying an element of Y for each element of X; from this point of view, f is rather like a program which outputs a value $f(x) \in Y$ for each $x \in X$. We could say that a functor is to a pair of categories as a function is to a pair of sets. Roughly, a functor from a category $\mathcal{C}$ to a category $\mathcal{D}$ is an assignment which sends each object of $\mathcal{C}$ to an object of $\mathcal{D}$, and each morphism of $\mathcal{C}$ to a morphism of $\mathcal{D}$. This assignment has to satisfy some rules. For example, the identity on an object A of $\mathcal{C}$ is sent to the identity in $\mathcal{D}$ on the object FA, where the functor sends the object A in $\mathcal{C}$ to FA in $\mathcal{D}$. Further, if two morphisms in $\mathcal{C}$ compose, then their images under the functor must compose in $\mathcal{D}$. Very informally, we might think of the functor as "preserving the structure" of $\mathcal{C}$. Let us move to the formal definition of a functor.*

*A **functor** F between categories $\mathcal{C}$ and $\mathcal{D}$, written as $F : \mathcal{C} \to \mathcal{D}$, is specified by*

- *an operation taking objects A in $\mathcal{C}$ to objects FA in $\mathcal{D}$, and*
- *an operation sending morphisms $f : A \to B$ in $\mathcal{C}$ to morphisms $Ff : FA \to FB$ in $\mathcal{D}$,*

for which $F(id_A) = id_{FA}$, and whenever the composition of morphisms gf is defined in $\mathcal{C}$ we have $F(gf) = Fg \circ Ff$. Note that $Fg \circ Ff$ is defined in $\mathcal{D}$ whenever gf is defined in $\mathcal{C}$, that is, Ff and Fg are composable in $\mathcal{D}$ whenever f and g are composable in $\mathcal{C}$.

Remark 2.3.2 *Sometimes we shall give the specification of a functor F by writing the operation on an object A as $A \mapsto FA$ and the operation on a morphism f, where $f : A \to B$,*

as $f : A \to B \mapsto Ff : FA \to FB$. *Provided that everything is clear, we shall sometimes even say "the functor* $f : \mathcal{C} \to \mathcal{D}$ *is defined by an assignment*

$$f : A \longrightarrow B \quad \mapsto \quad Ff : FA \longrightarrow FB$$

where $f : A \to B$ *is any morphism of* $\mathcal{C}$*." We shall refer informally to* $\mathcal{C}$ *as the source of the functor* F*, and to* $\mathcal{D}$ *as the target of* F.

Examples 2.3.3

(1) Let $\mathcal{C}$ *be a category. The* ***identity*** *functor* $id_{\mathcal{C}}$ *is defined by* $id_{\mathcal{C}}(A) \stackrel{\text{def}}{=} A$ *where* A *is an object of* $\mathcal{C}$ *and* $id_{\mathcal{C}}(f) \stackrel{\text{def}}{=} f$ *where* f *is a morphism of* $\mathcal{C}$.

(2) The Kleene closure A^* *of a set* A *gives rise to a monoid via list concatenation. Hence we may define a functor* $F : \mathcal{S}et \to \mathcal{M}on$ *by taking the operation on objects to be* $FA \stackrel{\text{def}}{=} A^*$ *and an operation on morphisms* $Ff \stackrel{\text{def}}{=} map(f)$*, where* $map(f) : A^* \to B^*$ *is defined by*

$$map(f)([a_1, \ldots, a_n]) = [f(a_1), \ldots, f(a_n)],$$

with $[a_1, \ldots, a_n]$ *any element of* A^**. Being our first example of a functor, we give explicit details of the verification that* F *is indeed a functor. To see that* $F(id_A) = id_{A^*}$ *note that*

$$\begin{aligned} F(id_A)([a_1, \ldots, a_n]) &\stackrel{\text{def}}{=} map(id_A)([a_1, \ldots, a_n]) \\ &= id_{A^*}([a_1, \ldots, a_n]) \\ &\stackrel{\text{def}}{=} id_{FA}([a_1, \ldots, a_n]), \end{aligned}$$

and to see that $F(gf) = Fg \circ Ff$ *where* $A \xrightarrow{f} B \xrightarrow{g} C$ *note that*

$$\begin{aligned} F(gf)([a_1, \ldots, a_n]) &\stackrel{\text{def}}{=} map(gf)([a_1, \ldots, a_n]) \\ &= [gf(a_1), \ldots, gf(a_n)] \\ &= map(g)([f(a_1), \ldots, f(a_n)]) \\ &= map(g)(map(f)([a_1, \ldots, a_n])) \\ &= Fg \circ Ff([a_1, \ldots, a_n]). \end{aligned}$$

(3) Given a set A*, recall that the powerset* $\mathcal{P}(A)$ *is the set of subsets of* A*. We can define a functor* $\mathcal{P} : \mathcal{S}et \to \mathcal{S}et$ *which is given by*

$$f : A \to B \quad \mapsto \quad f_* : \mathcal{P}(A) \to \mathcal{P}(B),$$

where $f : A \to B$ *is a function and* f_* *is defined by* $f_*(A') \stackrel{\text{def}}{=} \{f(a') \mid a' \in A'\}$ *where* $A' \in \mathcal{P}(A)$*. We call* $P : \mathcal{S}et \to \mathcal{S}et$ *the* ***covariant powerset*** *functor.*

(4) We can define a functor $\mathcal{P} : \mathcal{S}et^{op} \to \mathcal{S}et$ by setting

$$f : B \to A \quad \mapsto \quad f^{-1} : \mathcal{P}(B) \to \mathcal{P}(A),$$

*where $f : A \to B$ is a function in $\mathcal{S}et$, and the function f^{-1} is defined by $f^{-1}(B') \stackrel{\text{def}}{=} \{a \in A \mid f(a) \in B'\}$ where $B' \in \mathcal{P}(B)$. Note that the source of the functor $\mathcal{P}$ is the **opposite** of the category of sets and functions; we refer to $\mathcal{P}$ as the **contravariant powerset** functor.*

*(5) Given functors $F : \mathcal{C} \to \mathcal{C}'$ and $G : \mathcal{D} \to \mathcal{D}'$, the **product functor***

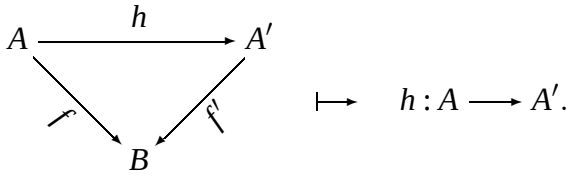
$$F \times G : \mathcal{C} \times \mathcal{D} \to \mathcal{C}' \times \mathcal{D}'$$

is defined in the expected coordinatewise manner.

(6) The functors between two preorders A and B regarded as categories are precisely the monotone functions from A to B.

(7) A functor between monoids is a monoid homomorphism, where we are regarding a monoid as a category with one object.

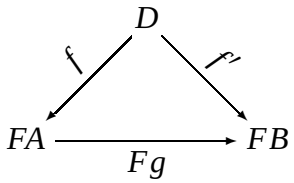
(8) Given a slice category $\mathcal{C}/B$, we may define a functor $U : \mathcal{C}/B \to \mathcal{C}$ as follows. Suppose that $f : A \to B$ is an object of the slice category, and that $h : f \to f'$ is a morphism of the slice. We set $Uf \stackrel{\text{def}}{=} A$ and $Uh \stackrel{\text{def}}{=} h$:



(9) Let $F : \mathcal{C} \to \mathcal{D}$ be any functor. Then we can define a functor $F^{op} : \mathcal{C}^{op} \to \mathcal{D}^{op}$ which sends an object A of $\mathcal{C}^{op}$ to FA in $\mathcal{D}^{op}$, and a morphism $f : A' \to A$ in $\mathcal{C}^{op}$ to $Ff : FA' \to FA$ in $\mathcal{D}^{op}$.

*(10) Given categories $\mathcal{C}$ and $\mathcal{D}$ and an object D of $\mathcal{D}$, the **constant** functor $\tilde{D} : \mathcal{C} \to \mathcal{D}$ sends any object A of $\mathcal{C}$ to D and any morphism $f : A \to B$ of $\mathcal{C}$ to $id_D : D \to D$.*

*(11) Let $F : \mathcal{C} \to \mathcal{D}$ be a functor and D an object of $\mathcal{D}$. Then the **under-cone** category $(D \downarrow F)$ has objects (f, A) where A is an object of $\mathcal{C}$ and $f : D \to FA$ is a morphism of $\mathcal{D}$. A morphism $g : (f, A) \to (f', B)$ is a morphism $g : A \to B$ for which the following diagram commutes:*



*The **over-cone** category $(F \downarrow D)$ is defined similarly.*

(12) Suppose that $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{C}' \to \mathcal{D}$ are functors. Then we define the ***comma*** *category $(F \downarrow G)$ to have objects which are triples (A, f, A') where A and A' are objects of $\mathcal{C}$ and $\mathcal{C}'$ respectively and $f : FA \to GA'$ is a morphism of $\mathcal{D}$. A morphism $(A, f, A') \to (B, f', B')$ is a pair (g, h) where $g : A \to B$ in $\mathcal{C}$ and $h : A' \to B'$ in $\mathcal{C}'$ for which the following diagram commutes:*

$$\begin{array}{ccc} FA & \xrightarrow{Fg} & FB \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f'} \\ GA' & \xrightarrow[Gh]{} & GB' \end{array}$$

(13) Let $\mathcal{G}rp$ be the category of groups and group homomorphisms. The ***forgetful*** *functor $U : \mathcal{G}rp \to \mathcal{S}et$ sends a group to its underlying set and a group homomorphism to its underlying function.*

Exercises 2.3.4

(1) Check that the definitions given in Examples 2.3.3 make sense and do indeed define functors between categories.

(2) Let us say that a category $\mathcal{C}$ is ***tiny*** *if the collection of objects forms a set and $\mathcal{C}$ is discrete; prove that a category $\mathcal{C}$ is tiny iff given any category $\mathcal{D}$ with a set of objects $ob\,\mathcal{D}$ and any set function $f : ob\,\mathcal{C} \to ob\,\mathcal{D}$, then f extends uniquely to a functor $F : \mathcal{C} \to \mathcal{D}$. (Extends means that if A is an object of $\mathcal{C}$, then $FA = f(A) \in ob\,\mathcal{D}$.)*

Discussion 2.3.5 *We shall end this section with a few definitions. A morphism $f : A \to A$ with equal source and target is called an* ***endomorphism****. A pair of morphisms f and g for which $src(f) = src(g)$ and $tar(f) = tar(g)$ are said to be* ***parallel*** *and this will be written $f, g : A \to B$. A functor with common source and target categories is called an* ***endofunctor****. Now let $F : \mathcal{C} \to \mathcal{D}$ be any functor. We say that F is* ***faithful*** *if given a parallel pair of morphisms $f, g : A \to B$ in $\mathcal{C}$ for which $Ff = Fg$, then $f = g$. We say that F is* ***full*** *if given objects A and B in $\mathcal{C}$ and a morphism $g : FA \to FB$ in $\mathcal{D}$, then there is some $f : A \to B$ in $\mathcal{C}$ for which $Ff = g$.*

As an example, consider the functor $U : \mathcal{M}on \to \mathcal{S}et$ which takes monoids and monoid homomorphisms to their underlying sets and set functions respectively. Then U is faithful but not full. The functor $U : \mathcal{G}rp \to \mathcal{M}on$ which takes groups and group homomorphisms to their underlying monoids is both full and faithful.

2.4 Natural Transformations and Examples

Not content with just the notion of a relation between categories, we now consider the notion of a relation between functors.

Let $\mathcal{C}$ and $\mathcal{D}$ be categories and $F, G : \mathcal{C} \to \mathcal{D}$ be functors. Then a **natural transformation** α from F to G, written $\alpha : F \to G$, is specified by an operation which assigns to each object A in $\mathcal{C}$ a morphism $\alpha_A : FA \to GA$ in $\mathcal{D}$, such that for any morphism $f : A \to B$ in $\mathcal{C}$, we have $Gf \circ \alpha_A = \alpha_B \circ Ff$, that is, the following diagram commutes:

$$\begin{array}{ccc} FA & \xrightarrow{\alpha_A} & GA \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FB & \xrightarrow[\alpha_B]{} & GB \end{array}$$

The morphism α_A is called the **component** of the natural transformation α at A. We shall also write $\alpha : F \to G : \mathcal{C} \to \mathcal{D}$ to indicate that α is a natural transformation between the functors $F, G : \mathcal{C} \to \mathcal{D}$. If we are given such a natural transformation, we shall refer to the above commutative square by saying "consider naturality of α in A at $f : A \to B$."

Examples 2.4.1

(1) Recall the functor $F : \mathcal{S}et \to \mathcal{M}on$ (see page 21) which takes a set to its Kleene closure. We can define a natural transformation $rev : F \to F$ which has components $rev_A : A^* \to A^*$ defined by

$$rev_A([a_1, \ldots, a_n]) \stackrel{\text{def}}{=} [a_n, \ldots, a_1]$$

where A is a set and $[a_1, \ldots, a_n] \in A^*$. It is trivial to see that this does define a natural transformation:

$$Ff \circ rev_A([a_1, \ldots, a_n]) = [f(a_n), \ldots, f(a_1)] = rev_B \circ Ff([a_1, \ldots, a_n]).$$

(2) Take a fixed set X and define a functor $F_X : \mathcal{S}et \to \mathcal{S}et$ by the operation $F_X(A) \stackrel{\text{def}}{=} (X \Rightarrow A) \times X$ on objects and the operation $F_X(f) \stackrel{\text{def}}{=} (f \circ -) \times id_X$ on morphisms where A is any set and f is any function. Here, $X \Rightarrow A$ is the set of functions from X to A, $(X \Rightarrow A) \times X$ is a cartesian product of sets, and $(f \circ -) \times id_X$ denotes a cartesian product of functions. Then we can define a natural transformation $ev : F_X \to id_{\mathcal{S}et}$ by setting $ev_A(g, x) \stackrel{\text{def}}{=} g(x)$ where $(g, x) \in (X \Rightarrow A) \times X$. To see that we have defined a natural transformation ev with components $ev_A : (X \Rightarrow A) \times X \to A$ let $f : A \to B$ be a set function, $(g, x) \in (X \Rightarrow A) \times X$ and note that

$$\begin{aligned} (id_{\mathcal{S}et}(f) \circ ev_A)(g, x) &= f(ev_A(g, x)) \\ &= f(g(x)) \\ &= ev_B(fg, x) \\ &= ev_B(F_X(f)(g, x)) \\ &= (ev_B \circ F_X(f))(g, x). \end{aligned}$$

(3) Let $\mathcal{V}ec$ be the category of vector spaces over a (fixed) field K. Write V^* for the set of linear maps from V into K. Then there is a functor $(-)^* : \mathcal{V}ec^{op} \to \mathcal{V}ec$, which is defined by

$$f : U \longrightarrow V \quad \longmapsto \quad f^* : U^* \xrightarrow{\theta \mapsto \theta f} V^*$$

where $f : U \to V$ is any morphism of $\mathcal{V}ec^{op}$ and $\theta \in U^*$ is any linear map. Thus there is a functor $(-)^{**} : \mathcal{V}ec \to \mathcal{V}ec$ which is defined by

$$f : V \longrightarrow U \quad \longmapsto \quad f^{**} : V^{**} \xrightarrow{(\chi \mapsto (\theta \mapsto \chi(f^*(\theta))))} U^{**}$$

where $\chi \in V^{**}$ is any linear map. For each vector space V there is a linear map $\alpha_V : V \to V^{**}$ given by $\alpha_V(v)(\theta) \stackrel{\text{def}}{=} \theta(v)$ where $v \in V$ and $\theta \in V^*$. It is easy to check that the diagram

$$\begin{array}{ccc} V & \xrightarrow{\alpha_V} & V^{**} \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle f^{**}} \\ U & \xrightarrow[\alpha_U]{} & U^{**} \end{array}$$

commutes, where $v \in V$:

$$\begin{array}{ccccc} v & \longmapsto & & & (\theta \mapsto \theta(v)) \\ \downarrow & & & & \downarrow \\ f(v) & \longmapsto & (\theta \mapsto \theta(f(v))) & = & (\theta \mapsto f^*(\theta)(v)) \end{array}$$

and hence that the α_V define a natural transformation $\alpha : id_{\mathcal{V}ec} \to (-)^{**}$.

Discussion 2.4.2 *It will be convenient to introduce some notation for dealing with methods of "composing" functors and natural transformations. Let $\mathcal{C}$ and $\mathcal{D}$ be categories and let F, G, H be functors from $\mathcal{C}$ to $\mathcal{D}$. Also let $\alpha : F \to G$ and $\beta : G \to H$ be natural transformations. We can define a natural transformation $\beta\alpha : F \to H$ by setting the components to be $(\beta\alpha)_A \stackrel{\text{def}}{=} \beta_A\alpha_A$. For clarity we will sometimes write $\beta \circ \alpha$ instead of $\beta\alpha$. This yields a category $[\mathcal{C}, \mathcal{D}]$ with objects functors from $\mathcal{C}$ to $\mathcal{D}$, morphisms natural transformations between such functors, and composition as given above. $[\mathcal{C}, \mathcal{D}]$ is called the* ***functor category*** *of $\mathcal{C}$ and $\mathcal{D}$.*

We end this discussion with a little more notation. Suppose that $\alpha : F \to G : \mathcal{D} \to \mathcal{E}$ is a natural transformation, and that $I : \mathcal{C} \to \mathcal{D}$ and $J : \mathcal{E} \to \mathcal{F}$ are functors. Then we can define a natural transformation $\alpha_I : FI \to GI : \mathcal{C} \to \mathcal{E}$ by setting components to be $(\alpha_I)_C \stackrel{\text{def}}{=} \alpha_{IC}$ where C is an object of $\mathcal{C}$, and a natural transformation $J\alpha : JF \to JG : \mathcal{D} \to \mathcal{F}$ by setting components $(J\alpha)_D \stackrel{\text{def}}{=} J(\alpha_D)$ where D is an object of $\mathcal{D}$.

Example 2.4.3 *Note that we may define a functor*

$$Ev : \mathcal{C} \times [\mathcal{C}, \mathcal{S}et] \longrightarrow \mathcal{S}et$$

where $Ev(A,F) \stackrel{\text{def}}{=} FA$ *and* $Ev(g,\mu) \stackrel{\text{def}}{=} \mu_{A'} \circ Fg = F'g \circ \mu_A$ *with* $g : A \to A'$ *and* $\mu : F \to F'$. *Such a functor* Ev *is usually referred to as an* ***evaluation*** *functor.*

Exercises 2.4.4

(1) Given categories $\mathcal{C}$ *and* $\mathcal{D}$, *verify that the functor category* $[\mathcal{C}, \mathcal{D}]$ *as defined in Discussion 2.4.2 is indeed a category.*

(2) Verify that the definitions given in Discussion 2.4.2 do indeed yield natural transformations. Further, given a diagram of categories and functors

$$\mathcal{C} \xrightarrow{\ I\ } \mathcal{D} \xrightarrow{\ F,G,H\ } \mathcal{E} \xrightarrow{\ J\ } \mathcal{F}$$

and natural transformations $\alpha : F \to G$ *and* $\beta : G \to H$, *show that* $J(\beta \circ \alpha) = J\beta \circ J\alpha$ *and* $(\beta \circ \alpha)_I = \beta_I \circ \alpha_I$. *Note: make sure you understand in which categories the compositions are defined.*

2.5 Isomorphisms and Equivalences

Discussion 2.5.1 *Of course, the basic idea of isomorphism is that isomorphic objects are "essentially the same." In the case of* $\mathcal{S}et$, *two sets* X *and* Y *are isomorphic just in case there is a bijection between them. This means that either there is a function* $f : X \to Y$ *which is injective and surjective, or, equivalently, there are functions* $f : X \to Y$ *and* $g : Y \to X$ *which are mutually inverse. We can use the idea that a pair of mutually inverse functions in the category* $\mathcal{S}et$ *gives rise to bijective sets to define the notion of isomorphism in an arbitrary category.*

A morphism $f : A \to B$ *in a category* $\mathcal{C}$ *is said to be an* ***isomorphism*** *if there is some* $g : B \to A$ *for which* $fg = id_B$ *and* $gf = id_A$. *We shall say that* g *is an* ***inverse*** *for* f *and that* f *is an inverse for* g. *Given objects* A *and* B *in* $\mathcal{C}$, *we say that* A *is* ***isomorphic*** *to* B *and write* $A \cong B$ *if such a mutually inverse pair of morphisms exists, and we say that the pair of morphisms* ***witnesses*** *the fact that* $A \cong B$. *Note that there may be many such pairs. In the category determined by a partially ordered set, the only isomorphisms are the identities, and in a preorder* X *with* $x, y \in X$ *we have* $x \cong y$ *iff* $x \leq y$ *and* $y \leq x$. *Note that in this case there can be only one pair of mutually inverse morphisms witnessing the fact that* $x \cong y$.

An isomorphism in a functor category is referred to as a ***natural isomorphism***. *If there is a natural isomorphism between the functors* F *and* G, *then we shall say that* F *and* G *are* ***naturally isomorphic***.

Exercises 2.5.2

*(1) Let $\mathcal{C}$ be a category and let $f : A \to B$ and $g, h : B \to A$ be morphisms. If $fh = id_B$ and $gf = id_A$ show that $g = h$. Deduce that any morphism f has a **unique** inverse if such exists.*

(2) Let $\mathcal{C}$ be a category and $f : A \to B$ and $g : B \to C$ be morphisms. If f and g are isomorphisms, show that gf is too. What is its inverse?

Lemma 2.5.3 *Let $\alpha : F \to G : \mathcal{C} \to \mathcal{D}$ be a natural transformation. Then α is a natural isomorphism just in case each component α_C is an isomorphism in $\mathcal{D}$. More precisely, if we are given a natural isomorphism α in $[\mathcal{C}, \mathcal{D}]$ with inverse β, then each β_C is an inverse for α_C in $\mathcal{D}$; and if given a natural transformation α in $[\mathcal{C}, \mathcal{D}]$ for which each component α_C has an inverse (say β_C) in $\mathcal{D}$, then the β_C are the components of a natural transformation β which is the inverse of α in $[\mathcal{C}, \mathcal{D}]$.*

Proof Direct calculations from the definitions. □

Exercise 2.5.4 *Do the proof of Lemma 2.5.3. Be careful to note precisely what the lemma is saying.*

Discussion 2.5.5 *We emphasised at the start of this chapter that we were interested in the way structures behaved and not so much in their internal make-up. To this end we might expect that the judgement "A and B are isomorphic in $\mathcal{C}$" is more important than the judgement "A and B are equal in $\mathcal{C}$." Indeed, the development of category theory has shown this to be the case. This leads us to the notion of equivalence of categories. If two categories are equivalent, then the rough idea is that we can write down a one to one correspondence between isomorphism classes of objects obtained from the categories. Let us give the precise definition.*

*Two categories $\mathcal{C}$ and $\mathcal{D}$ are **equivalent** if there are functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ together with natural isomorphisms $\varepsilon : FG \cong id_{\mathcal{D}}$ and $\eta : id_{\mathcal{C}} \cong GF$. We say that F is an **equivalence** with an **inverse equivalence** G and denote the equivalence by $F : \mathcal{C} \simeq \mathcal{D} : G$.*

Example 2.5.6 *Let us look at an example of equivalent categories. Recall $\mathcal{P}art$, the category of sets and partial functions, and write 1 for a singleton set. Recall also the definition of a coslice category. Then it is the case that $\mathcal{P}art \simeq 1/\mathcal{S}et$. Note that an object of $1/\mathcal{S}et$ is a function $f : 1 \to A$ where A is a set (and hence in particular A is non-empty); this amounts to giving a pair (A, a) where $a \in A$, and a is sometimes referred to as the distinguished element of A. A morphism $g : (A, a) \to (B, b)$ amounts to a function $g : A \to B$ for which $b = g(a)$, that is g preserves the distinguished element of A.*

We pause to consider the intuition behind the equivalence. Any partial function $f : A \to B$ can be regarded as a total function $\overline{f} : A \cup \{*\} \to B \cup \{*\}$ (choose $*$ such that $* \notin A$ and

$* \notin B$) *where*

$$\overline{f}(\xi) \;=\; \begin{cases} f(\xi) \text{ if } \xi \in A \text{ and } f(\xi) \text{ is defined} \\ * \text{ otherwise} \end{cases}$$

for any $\xi \in A \cup \{*\}$*; and of course* $A \cup \{*\}$ ($B \cup \{*\}$) *can be viewed as a set with a distinguished element. Conversely, any total function whose source is a set with a distinguished element can be seen to give rise to a partial function.*

Now we look more formally at the equivalence of $\mathcal{P}art$ *and* $1/\mathcal{S}et$*. Let* (X,x_0) *and* (Y,y_0) *be objects of* $1/\mathcal{S}et$*, and* $f : (X,x_0) \to (Y,y_0)$ *a morphism. Define a functor* $F : 1/\mathcal{S}et \to \mathcal{P}art$ *by setting* $F(X,x_0) \stackrel{\text{def}}{=} X \setminus \{x_0\}$ *and*

$$F f(x) \;\stackrel{\text{def}}{=}\; \begin{cases} f(x) \text{ provided } f(x) \neq y_0 \\ \text{undefined otherwise} \end{cases}$$

where of course $x \in X \setminus \{x_0\}$ *and* $F f : X \setminus \{x_0\} \to Y \setminus \{y_0\}$*.*

Let $g : X \to Y$ *be a partial function. Define a functor* $G : \mathcal{P}art \to 1/\mathcal{S}et$ *by setting* $GX \stackrel{\text{def}}{=} (X \cup \{X\}, X)$ *and*

$$Gg(\xi) \;\stackrel{\text{def}}{=}\; \begin{cases} g(\xi) \text{ if } \xi \neq X \text{ and } g(\xi) \in Y \text{ is defined} \\ Y \text{ otherwise} \end{cases}$$

where of course $Gg : X \cup \{X\} \to Y \cup \{Y\}$ *and* $\xi \in X \cup \{X\}$*. Then one can check that this does give rise to an equivalence of categories.*

Exercises 2.5.7

(1) Investigate the notion of equivalence of preorders regarded as categories. Draw some pictures of equivalent preorders.

(2) Complete a detailed verification of the equivalence given in Example 2.5.6.

(3) The slice category $\mathcal{S}et/B$ *is often referred to as the category of B-indexed families of sets with functions preserving the indexing. To see this, note that a function* $f : X \to B$ *gives rise to the family of sets* $(f^{-1}(b) \mid b \in B)$*, and the family of sets* $(X_b \mid b \in B)$ *gives rise to the function*

$$f : \{(x,b) \mid x \in X_b, b \in B\} \to B$$

where $f(x,b) \stackrel{\text{def}}{=} b$*. Note that we can regard the set B as a discrete category; then there is an equivalence between the functor category* $[B, \mathcal{S}et]$ *and the slice* $\mathcal{S}et/B$*. Formulate this equivalence carefully and prove that your definitions really do give an equivalence.*

2.6 Products and Coproducts

Discussion 2.6.1 *The notion of "product of two objects in a category" can be viewed as an abstraction of the idea of a cartesian product of two sets. As we mentioned in Section 2.1,*

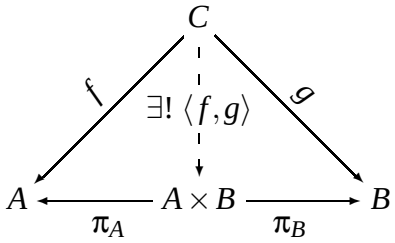
the definition of a cartesian product of sets is an "internal" one; we specify the elements of the product in terms of the elements of the sets from which the product is composed. However, as we have seen, there is a property of the cartesian product which characterises it ***up to set-theoretic bijection****. The internal make up of two bijective sets may be very different, but their properties may well be very similar. We can now give the formal definition of a binary product of two objects of a category.*

A ***binary product*** *of objects A and B in a category $\mathcal{C}$ is specified by*

- *an object $A \times B$ of $\mathcal{C}$, together with*
- *two* ***projection*** *morphisms $\pi_A : A \times B \to A$ and $\pi_B : A \times B \to B$,*

for which given any object C and morphisms $f : C \to A$, $g : C \to B$, there is a unique morphism $\langle f, g \rangle : C \to A \times B$ for which $\pi_A \langle f, g \rangle = f$ and $\pi_B \langle f, g \rangle = g$.

We shall refer simply to a binary product $A \times B$ instead of the triple $(A \times B, \pi_A, \pi_B)$, without explicit mention of the projection morphisms, much as it is common practice to speak of a group G rather than $(G, \circ, e)$. The data for a binary product is more readily understood as a commutative diagram, where we have written $\exists!$ to mean "there exists a unique":

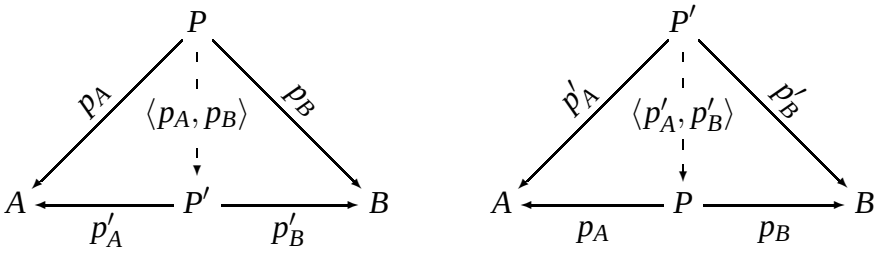


Given a binary product $A \times B$ and morphisms $f : C \to A$ and $g : C \to B$, the unique morphism $\langle f, g \rangle : C \to A \times B$ (making the above diagram commute) is called the ***mediating*** *morphism for f and g. We shall say that the category $\mathcal{C}$* ***has binary products*** *if there is a product in $\mathcal{C}$ of any two objects A and B, and that $\mathcal{C}$ has* ***specified*** *binary products if there is a given canonical choice of binary product for each pair of objects. For example, $\mathcal{S}et$ has specified binary products by setting $A \times B \stackrel{\text{def}}{=} \{ (a, b) \mid a \in A, b \in B \}$ with projections given by the usual set-theoretic projection functions. Talking of* ***specified*** *binary products is a reasonable thing to do, by virtue of the next result.*

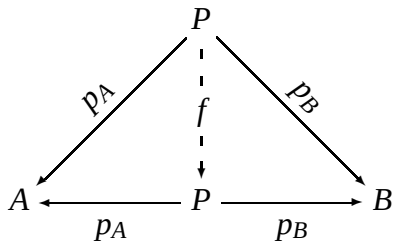
Lemma 2.6.2 *A binary product of A and B in a category $\mathcal{C}$ is unique up to isomorphism if it exists.*

Proof Suppose that (P, p_A, p_B) and (P', p'_A, p'_B) are two candidates for the binary product. Then we have $\langle p_A, p_B \rangle : P \to P'$ by applying the defining property of (P', p'_A, p'_B) to the morphisms $p_A : P \to A$, $p_B : P \to B$, and $\langle p'_A, p'_B \rangle : P' \to P$ exists from a similar

argument. So we have diagrams of the form



But then $f \stackrel{\text{def}}{=} \langle p'_A, p'_B \rangle \langle p_A, p_B \rangle : P \to P$ and one can check that $p_A f = p_A$ and that $p_B f = p_B$, that is f is a mediating morphism for the binary product (P, p_A, p_B); we can picture this as the following commutative diagram:



But it is trivial that id_P is also such a mediating morphism, and so uniqueness implies $f = id_P$. Similarly one proves that $\langle p_A, p_B \rangle \langle p'_A, p'_B \rangle = id_{P'}$, to deduce $P \cong P'$ witnessed by the morphisms $\langle p_A, p_B \rangle$ and $\langle p'_A, p'_B \rangle$. □

Remark 2.6.3 *We sometimes refer to a property which involves the "existence of a unique morphism (functor)" leading to a structure which is determined up to isomorphism (equivalence) as a* ***universal*** *property.*

Discussion 2.6.4 *The notion of binary product has an extension to set-indexed families of objects.*

Given a family of objects $(A_i \mid i \in I)$ in a category $\mathcal{C}$ where I is a set, a ***product*** *of the family of objects is specified by*

- *an* ***object*** $\Pi_{i \in I} A_i$ *in* $\mathcal{C}$*, and*
- *for every $j \in I$, a morphism $\pi_j : \Pi_{i \in I} A_i \to A_j$ in $\mathcal{C}$ called the jth* ***product projection****,*

such that for any object C and family of morphisms $(f_i : C \to A_i \mid i \in I)$ there is a unique morphism

$$\langle f_i \mid i \in I \rangle : C \to \Pi_{i \in I} A_i$$

for which given any $j \in I$, we have $\pi_j \langle f_i \mid i \in I \rangle = f_j$. We shall say that $\langle f_i \mid i \in I \rangle$ is the ***mediating*** *morphism for the family $(f_i \mid i \in I)$.*

Remark 2.6.5 *In the definition of product, for any family $(A_i \mid i \in I)$ we speak of* ***a*** *product and not* ***the*** *product. This is because a product is unique only up to isomorphism, and the proof that this is the case is similar to that for binary products (Lemma 2.6.2). We shall sometimes speak of a product $\Pi_{i\in I}A_i$ without explicit mention of the product projections.*

Examples 2.6.6

(1) A ***terminal object*** *1 of a category $\mathcal{C}$ has the property that for any object C of the ambient category, there is a unique morphism $!_C : C \to 1$. Note that such an object is unique up to isomorphism. For suppose that both 1 and $1'$ are terminal objects of $\mathcal{C}$. Then there is a unique morphism $!_{1'} : 1' \to 1$, and a unique morphism $!_1 : 1 \to 1'$. Thus by composition there is a morphism $!_{1'} \circ !_1 : 1 \to 1$. But 1 is a terminal object, and so we must have $!_{1'} \circ !_1 = id_1$, because there has to be a unique morphism $!_1 : 1 \to 1$ and this must therefore be the identity. Similarly $!_1 \circ !_{1'} = id_{1'}$ and thus $1 \cong 1'$ as required. Note that this kind of argument using uniqueness properties is prevalent in category theory.*

(2) A product of an empty family of objects (that is a family indexed by the empty set) in a category $\mathcal{C}$ is given by a terminal object. The first clause of the definition of product says we are given an object of $\mathcal{C}$, say 1, and the remainder of the definition reduces to saying that for each object C there is a unique morphism $C \to 1$. A singleton set is an example of a terminal object in the category $\mathcal{S}et$.

(3) A ***global element*** *of an object A in a category $\mathcal{C}$ is any morphism $1 \to A$. What are global elements in the category $\mathcal{S}et$?*

(4) Now let $I = \{1,2\}$. A product of a family (A_1, A_2) of two objects is just a binary product of A_1 and A_2 as defined on page 29. We usually denote the product $\Pi_{i\in\{1,2\}}A_i$ by $A_1 \times A_2$ and write $\langle f_1, f_2\rangle$ for any mediating morphism $\langle f_i \mid i \in \{1,2\}\rangle$.

(5) When I is a finite set we shall speak of a finite product. If $I = \{1,\ldots,n\}$ we shall sometimes write $A_1 \times \ldots \times A_n$ or $\Pi_1^n A_i$ for the product of $(A_i \mid i \in I)$. We shall sometimes abuse notation and allow n to be 0 in the notation $\Pi_1^n A_i$. Thus $\Pi_1^0 A_i$ will indicate a product of no objects, that is, a terminal object.

Discussion 2.6.7 *A category $\mathcal{C}$* ***has*** *products of set-indexed families of objects if there is always a product for any such family. $\mathcal{C}$ has* ***specified*** *products of set-indexed families of objects if it has products and there is always a specified canonical choice. We shall say that $\mathcal{C}$ has* ***finite*** *products if it has products of finite families of objects. We shall see that such categories with finite products arise naturally in the semantics of algebraic type theory. It will be useful to have a little additional notation for dealing with such categories. Let $\mathcal{C}$ be a category with finite products and take morphisms $f : A \to B$ and $f' : A' \to B'$. We write $f \times f' : A \times A' \to B \times B'$ for the morphism $\langle f\pi, f'\pi'\rangle$ where $\pi : A \times A' \to A$ and $\pi' : A \times A' \to A'$. The uniqueness condition of mediating morphisms means that in general one has*

$$id_A \times id_{A'} = id_{A\times A'} \qquad \text{and} \qquad (g \times g')(f \times f') = gf \times g'f',$$

where $g : B \to C$ and $g' : B' \to C'$.

Exercises 2.6.8

(1) Show that a category $\mathcal{C}$ has finite products just in case it has binary products and a terminal object.

(2) Let $\mathcal{C}$ be a category with finite products and let

$$\begin{array}{lll} l : X \to A & f : A \to B & g : A \to C \\ h : B \to D & k : C \to E & \end{array}$$

be morphisms of $\mathcal{C}$. Show that $(h \times k) \circ \langle f, g \rangle = \langle hf, kg \rangle$ and $\langle f, g \rangle \circ l = \langle fl, gl \rangle$.

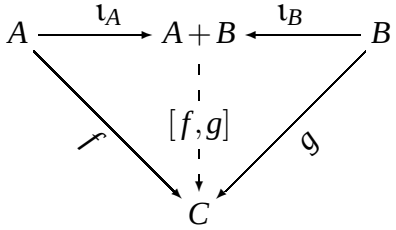
(3) Investigate the notion of a binary product in a category $\mathcal{C}^{op}$.

Discussion 2.6.9 *A coproduct is a dual notion of product. For example, a binary coproduct of a pair of objects in a category $\mathcal{C}$ is given by the binary product of the objects in the opposite category $\mathcal{C}^{op}$. With this definition, readers should provide a definition of binary coproduct for themselves, and check their conclusion with the following definition.*

*A **binary coproduct** of objects A and B in a category $\mathcal{C}$ is specified by*

- *an object $A + B$ of $\mathcal{C}$, together with*
- *two **insertion** morphisms $\iota_A : A \to A + B$ and $\iota_B : B \to A + B$,*

such that for each pair of morphisms $f : A \to C$, $g : B \to C$ there exists a unique morphism $[f, g] : A + B \to C$ for which $[f, g]\iota_A = f$ and $[f, g]\iota_B = g$. We can picture this definition through the following commutative diagram:



*We can extend this definition to the notion of a coproduct of a set-indexed family of objects. Given a family of objects $(A_i \mid i \in I)$ in a category $\mathcal{C}$, where I is a set, a **coproduct** is specified by the following data:*

- *An object $\Sigma_{i \in I} A_i$ in $\mathcal{C}$, and*
- *for every $j \in I$, a morphism $\iota_j : A_j \to \Sigma_{i \in I} A_i$ called the **jth coproduct insertion**,*

such that for any object C and any family of morphisms $(f_i : A_i \to C \mid i \in I)$ there is a unique morphism $[f_i \mid i \in I] : \Sigma_{i \in I} A_i \to C$ for which given any $j \in I$, we have

$$[f_i \mid i \in I]\iota_j = f_j.$$

Let $\mathcal{C}$ be a category with finite coproducts and take morphisms $f : A \to B$ and $f' : A' \to B'$. We write $f + f' : A + A' \to B + B'$ for the morphism $[\iota_B \circ f, \iota_{B'} \circ f']$ where $\iota_B : B \to B + B'$ and $\iota_{B'} : B' \to B + B'$. The uniqueness condition of mediating morphisms means that one has

$$id_A + id_{A'} = id_{A+A'} \qquad \text{and} \qquad (g + g')(f + f') = gf + g'f',$$

where $g : B \to C$ and $g' : B' \to C'$.

Exercise 2.6.10 *Prove the coproduct of any set-indexed family of objects is unique up to isomorphism if it exists.*

Examples 2.6.11

(1) An object 0 of a category $\mathcal{C}$ is called ***initial*** *if there is a unique morphism $!_A : 0 \to A$ for each object A of $\mathcal{C}$. Note that such an object is unique up to isomorphism. Then a coproduct of an empty family of objects of $\mathcal{C}$ (that is a family indexed by the empty set) is an initial object for $\mathcal{C}$, and the reader should verify this.*

(2) In the category $\mathcal{S}et$, the binary coproduct of sets A and B is given by their disjoint union together with the obvious insertion functions. We can define the disjoint union $A \uplus B$ of A and B as the union $(A \times \{t\}) \bigcup (B \times \{t'\})$ (where t, t' are not elements of A or B) with the insertion functions

$$\iota_A : A \to A \uplus B \leftarrow B : \iota_B$$

where ι_A is defined by $a \mapsto (a, t)$ for all $a \in A$, and ι_B is defined analogously.

(3) Suppose that X is a lattice viewed as a category. Then the top of X is a terminal object, the bottom of X an initial object, and finite meets and joins are finite products and coproducts respectively.

(4) The category $\mathcal{POS}et$ has binary products through the pointwise order on the cartesian product of (the underlying sets of) posets X and Y, together with the monotone set-theoretic projection functions. Also, any one point poset is a terminal object.

Discussion 2.6.12 *In a category with finite products we can show that for any objects A, B and C that there is an isomorphism $A \times (B \times C) \cong (A \times B) \times C$. Moreover, we can show that both the left hand side and the right hand side of this isomorphism are products of the triple (A, B, C). One might loosely say that $A \times (B \times C)$, $(A \times B) \times C$ and $A \times B \times C$ each satisfy the same specification, but are implemented differently. In concrete examples, the internal make up of these objects will be different, but they will each have similar external properties.*

If $\mathcal{C}$ and $\mathcal{D}$ are categories with finite products, then the functor $F : \mathcal{C} \to \mathcal{D}$ ***preserves finite products*** *if for any finite family of objects $(A_1, \ldots, A_n)$ in $\mathcal{C}$ the morphism*

$$m \stackrel{\text{def}}{=} \langle F\pi_i \mid i \in I \rangle : F(A_1 \times \ldots \times A_n) \to FA_1 \times \ldots \times FA_n$$

is an isomorphism, where of course $F\pi_j : F(A_1 \times \ldots \times A_n) \to FA_j$ is F applied to the projection π_j for each $j \in I$. Note that there may be witnesses other than m to the above isomorphism; we refer to m as the ***canonical*** *isomorphism. In the case that $n = 0$ we say that F* ***preserves terminal objects****, that is the unique morphism $F(1_{\mathcal{C}}) \to 1_{\mathcal{D}}$ is an isomorphism where $1_{\mathcal{C}}$ and $1_{\mathcal{D}}$ are the terminal objects of $\mathcal{C}$ and $\mathcal{D}$ respectively. A finite product preserving functor F is* ***strict*** *if the above isomorphisms are identities.*

If $\mathcal{C}$ and $\mathcal{D}$ are categories with finite coproducts, then the functor $F : \mathcal{C} \to \mathcal{D}$ ***preserves finite coproducts*** *if for any finite family of objects $(A_1, \ldots, A_n)$ in $\mathcal{C}$ the morphism*

$$m \stackrel{\text{def}}{=} [F\iota_i \mid i \in I] : FA_1 + \ldots + FA_n \to F(A_1 + \ldots + A_n)$$

is an isomorphism, where of course $F\iota_j : FA_j \to F(A_1 + \ldots + A_n)$ is F applied to the insertion ι_j for each $j \in I$. In the case that $n = 0$ we say that F ***preserves initial objects****, that is the unique morphism $0_{\mathcal{D}} \to F(0_{\mathcal{C}})$ is an isomorphism where $0_{\mathcal{C}}$ and $0_{\mathcal{D}}$ are the initial objects of $\mathcal{C}$ and $\mathcal{D}$ respectively. A finite coproduct preserving functor F is* ***strict*** *if the above isomorphisms are identities.*

Exercise 2.6.13 *Find an example of a functor $F : \mathcal{C} \to \mathcal{D}$ for which*

$$F(A \times B) \cong FA \times FB$$

in $\mathcal{D}$ for all pairs of objects A and B in $\mathcal{C}$, but such that F does not preserve binary products. ***Hint: think about countably infinite sets.***

2.7 The Yoneda Lemma

Discussion 2.7.1 *We learn in a first course in set theory that the collection of all sets cannot be a set but is in fact a proper class. In category theory we have to deal with collections which are large in a very real sense. Questions of size are important, but we do not delve into formal details here. We shall, however, make the following definition.*

A category $\mathcal{C}$ is ***small*** *if its morphisms can be indexed by a set; note that this implies the collection of objects of $\mathcal{C}$ is a set. $\mathcal{C}$ is* ***locally small*** *if for any pair of objects A and B the collection of morphisms from A to B can be indexed by a set. This set is usually written $Hom_{\mathcal{C}}(A,B)$ or $\mathcal{C}(A,B)$. For example, if X is a preorder viewed as a category, then*

$$X(x,y) \quad = \quad \begin{cases} \{*\} \text{ if } x \leq y \\ \varnothing \text{ otherwise} \end{cases}$$

where $x, y \in X$ and $\{*\}$ is a one point set. Most categories that one meets in everyday use are indeed locally small. However, functor categories are not always so. If $\mathcal{C}$ is small and $\mathcal{D}$ is locally small, then $[\mathcal{C}, \mathcal{D}]$ is locally small. This is because the components of the natural transformations in $[\mathcal{C}, \mathcal{D}]$ are indexed by the objects of $\mathcal{C}$ and this collection is a set by

hypothesis. More precisely, to give a natural transformation $\alpha : F \to G$, we have to give a morphism $\alpha_C : FC \to GC$ in $\mathcal{D}$ for each object C of $\mathcal{C}$. Thus there is a candidate for such a natural transformation α for every function

$$ob\,\mathcal{C} \longrightarrow \bigcup\{\ \mathcal{D}(FC, GC)\ \mid\ C \in ob\,\mathcal{C}\ \}$$

and there is certainly a set of such functions. If $\mathcal{C}$ and $\mathcal{D}$ are locally small then $[\mathcal{C}, \mathcal{D}]$ need not be locally small.

Now it is time to introduce some more notation. Let A and B be objects of a locally small category $\mathcal{C}$. We can define a functor

$$\mathcal{C}(-,+) : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{S}et$$

by taking any morphism $(f,g) : (A,B) \to (A',B')$ in $\mathcal{C}^{op} \times \mathcal{C}$ to the set-theoretic function

$$\mathcal{C}(f,g) : \mathcal{C}(A,B) \to \mathcal{C}(A',B')$$

where $\mathcal{C}(f,g)(h) = ghf$ for each morphism $h : A \to B$. (Note that f is a morphism $A' \to A$ in $\mathcal{C}$). We can also define a functor

$$\mathcal{C}(A,+) : \mathcal{C} \to \mathcal{S}et.$$

As expected, this functor takes objects B of $\mathcal{C}$ to the set $\mathcal{C}(A,B)$, and if $g : B \to B'$ is a morphism of $\mathcal{C}$ then the functor $\mathcal{C}(A,+)$ takes $g : B \to B'$ to the function $\mathcal{C}(A,g) : \mathcal{C}(A,B) \to \mathcal{C}(A,B')$ defined by setting $\mathcal{C}(A,g)(h) \stackrel{\text{def}}{=} gh$, where $h : A \to B$. Similarly, we can define a functor $\mathcal{C}(-,B) : \mathcal{C}^{op} \to \mathcal{S}et$. We shall also write H^A for the functor $\mathcal{C}(A,+)$ and H_B for the functor $\mathcal{C}(-,B)$, when the category to which we are referring is clear. If $h : C \to C'$ is a morphism of $\mathcal{C}$ then $H^A h : H^A C \to H^A C'$ will be written as $H^A_h : H^A_C \to H^A_{C'}$, with a similar notation adopted for H_B. Finally, we shall write

$$H : \mathcal{C}^{op} \to [\mathcal{C}, \mathcal{S}et]$$

*for the functor which sends $f : A' \to A$ to $H^f : H^A \to H^{A'}$. The functor H is often called the **Yoneda embedding** of $\mathcal{C}^{op}$ into the functor category $[\mathcal{C}, \mathcal{S}et]$. There is also a functor $H : \mathcal{C} \to [\mathcal{C}^{op}, \mathcal{S}et]$ defined in a similar fashion, that is by the assignment*

$$f : A \longrightarrow A' \quad \mapsto \quad H_f : H_A \longrightarrow H_{A'}$$

where $f : A \to A'$ is a morphism of $\mathcal{C}$.

Now suppose that we are given functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ between locally small categories $\mathcal{C}$ and $\mathcal{D}$. Then there are functors

$$\begin{array}{l} \mathcal{D}^{op} \times \mathcal{C} \xrightarrow{id \times F} \mathcal{D}^{op} \times \mathcal{D} \xrightarrow{\mathcal{D}(-,+)} \mathcal{S}et \\ \mathcal{D}^{op} \times \mathcal{C} \xrightarrow{G^{op} \times id} \mathcal{C}^{op} \times \mathcal{C} \xrightarrow{\mathcal{C}(-,+)} \mathcal{S}et \end{array}$$

and the compositions will usually be written as $\mathcal{D}(-,F+)$, $\mathcal{C}(G-,+)$ respectively. See page 22 for the definition of G^{op}: the omission of the "op" in the notation $\mathcal{C}(G-,+)$ arises from the fact that the action of the functor G^{op} is essentially to "apply G." If A is an object of $\mathcal{D}$ and B an object of $\mathcal{C}$, then there are obvious functors $\mathcal{D}(A,F+) : \mathcal{C} \to \mathcal{S}et$ and $\mathcal{C}(G-,B) : \mathcal{D}^{op} \to \mathcal{S}et$.

Let us finish this discussion with the definition of a representable functor. Given a functor $F : \mathcal{C} \to \mathcal{S}et$ where $\mathcal{C}$ is locally small, we say that F is ***representable*** *if there is an object A of $\mathcal{C}$ for which we can find a natural isomorphism $H^A \cong F$. We say that a pair (a,A) where $a \in FA$ is a* ***representation*** *of F if the natural transformation $\Psi : H^A \to F$ which has components $\Psi_B : \mathcal{C}(A,B) \to FB$ given by $\Psi_B(f) \stackrel{\text{def}}{=} Ff(a)$ is a natural isomorphism, where $f : A \to B$ is any morphism of $\mathcal{C}$.*

Example 2.7.2 *To give some feel for the definition we look at a concrete example. Recall that the set of subsets of a given set X, $\mathcal{P}(X)$, is in bijection with the set of functions from X to a two point set $\{0,1\}$. For if we think of 1 as meaning true and 0 as meaning false, then given a subset X' of X and an element $x \in X$, we can ask whether it is true or not that $x \in X'$. Thus we can see that a two point set can represent (or code) information about subset membership. We can use the idea of a representable functor to formalise these ideas. In fact it is the case that the contravariant powerset functor $\mathcal{P} : \mathcal{S}et^{op} \to \mathcal{S}et$ is represented by $(\{1\},\{0,1\})$. The natural transformation $\Psi : H^{\{0,1\}} \to \mathcal{P}$ has components*

$$\Psi_X : \mathcal{S}et(X,\{0,1\}) \xrightarrow{\chi \mapsto \mathcal{P}(\chi)(\{1\}) \stackrel{\text{def}}{=} \chi^{-1}(\{1\})} \mathcal{P}(X)$$

where X is any set and $\chi : X \to \{0,1\}$ is a function. It is easy to check that Ψ is a natural isomorphism.

Discussion 2.7.3 *We are now in a position to state and prove the Yoneda lemma. On first reading, this material may feel rather heavy. However, the Yoneda lemma is an indispensable tool which every category theorist should carry in his kit and be able to use. We shall make use of the Yoneda lemma when discussing the semantics of datatypes in later chapters. Roughly, the Yoneda lemma says that for a locally small category $\mathcal{C}$ and a functor $F : \mathcal{C} \to \mathcal{S}et$, if we choose any object A of $\mathcal{C}$, there is a bijection between the elements of FA and natural transformations from H^A to F.*

Lemma 2.7.4 *Let $\mathcal{C}$ be a locally small category, $F : \mathcal{C} \to \mathcal{S}et$ a functor and A an object of $\mathcal{C}$. Then the collection $Nat(H^A,F)$ of natural transformations $H^A \to F$ is a set and so we can define a functor*

$$Nat(H^-,+) : \mathcal{C} \times [\mathcal{C},\mathcal{S}et] \longrightarrow \mathcal{S}et$$

as follows. The morphism $(g,\mu) : (A,F) \to (A',F')$ in $\mathcal{C} \times [\mathcal{C},\mathcal{S}et]$ is taken to the function

$$Nat(H^g,\mu) : Nat(H^A,F) \to Nat(H^{A'},F')$$

which is defined by $Nat(H^g,\mu)(\alpha) \stackrel{\text{def}}{=} \mu\circ\alpha\circ H^g$ *where* $\alpha : H^A \to F$ *is a natural transformation. Recall also (page 26) the evaluation functor*

$$Ev : \mathcal{C}\times[\mathcal{C},\mathcal{S}et] \longrightarrow \mathcal{S}et.$$

Then there is a natural isomorphism $\Phi : Nat(H^-,+) \cong Ev : \Psi$. *If* A *is an object of* $\mathcal{C}$, *this amounts to saying that there is an isomorphism (set-theoretic bijection)*

$$\Phi_{(A,F)} : Nat(H^A,F) \cong FA : \Psi_{(A,F)}$$

and this isomorphism is natural in (A,F).

Proof *In the first part of the proof, we show that for each* A *and* F *there is a bijective assignment between* $Nat(H^A,F)$ *and* FA, *establishing that* $Nat(H^A,F)$ *is indeed a set and thus* $Nat(H^-,+)$ *is well defined. We can define an assignment*

$$\Phi_{(A,F)} : Nat(H^A,F) \longrightarrow FA$$

by setting $\Phi_{(A,F)}(\alpha) \stackrel{\text{def}}{=} \alpha_A(id_A)$ *for* $\alpha : H^A \to F$ *a natural transformation, and define an assignment*

$$\Psi_{(A,F)} : FA \longrightarrow Nat(H^A,F)$$

by setting $\Psi_{(A,F)}(a)_B(f) \stackrel{\text{def}}{=} F f(a)$ *where* $a \in FA$, B *is an object of* $\mathcal{C}$, $\Psi_{(A,F)}(a)_B$ *is the component of the natural transformation* $\Psi_{(A,F)}(a)$ *at* B, *and* $f : A \to B$ *is a morphism in* $\mathcal{C}$, *where of course* $\Psi_{(A,F)}(a)_B : \mathcal{C}(A,B) \to FB$. *Note that one has to check that* $\Psi_{(A,F)}(a)$ *is a natural transformation. We urge the reader to do this.*

Now we check that assignments $\Phi_{(A,F)}$ *are isomorphisms (bijections) with inverses given by the* $\Psi_{(A,F)}$. *One way round, we check that* $\Psi_{(A,F)}\Phi_{(A,F)} = id_{Nat(H^A,F)}$. *To see this, first note that* $\Psi_{(A,F)}\Phi_{(A,F)}(\alpha) = \Psi_{(A,F)}(\alpha_A(id_A))$. *Then we have*

$$\begin{aligned}
\Psi_{(A,F)}(\alpha_A(id_A))_B(f) &= F f(\alpha_A(id_A)) \\
\textit{because } \alpha \textit{ is natural} \quad &= \alpha_B(H^A_f(id_A)) \\
&= \alpha_B(f\circ id_A) \\
&= \alpha_B(f).
\end{aligned}$$

Thus the components of $\Psi_{(A,F)}(\alpha_A(id_A))$ *are the same as those of* α *which is to say that the natural transformations* $\Psi_{(A,F)}\Phi_{(A,F)}(\alpha)$ *and* α *are identical, as required.*

For the other way we check that $\Phi_{(A,F)}\Psi_{(A,F)} = id_{FA}$. *Let* $a \in FA$; *then we have*

$$\begin{aligned}
\Phi_{(A,F)}\Psi_{(A,F)}(a) &= \Phi_{(A,F)}(\Psi_{(A,F)}(a)) \\
&= \Psi_{(A,F)}(a)_A(id_A) \\
&= (F(id_A))(a) \\
&= a.
\end{aligned}$$

Hence we have established a bijection between FA and the collection $Nat(H^A, F)$, implying that the latter is indeed a set.

In fact the functions $\Phi_{(A,F)}$ and $\Psi_{(A,F)}$ give rise to natural isomorphisms Φ and Ψ. We shall check that $\Phi : Nat(H^-, +) \longrightarrow Ev$ is a natural transformation, that is the diagram

$$\begin{array}{ccc} Nat(H^A, F) & \xrightarrow{\Phi_{(A,F)}} & FA \\ {\scriptstyle Nat(H^g,\mu)}\downarrow & & \downarrow{\scriptstyle F'g\circ\mu_A} \\ Nat(H^{A'}, F') & \xrightarrow[\Phi_{(A',F')}]{} & F'A' \end{array}$$

commutes, where $(g,\mu) : (A,F) \to (A',F')$ is a morphism in $\mathcal{C} \times [\mathcal{C}, \mathcal{S}et]$. Let α be a natural transformation from H^A to F. Then we have

$$\begin{aligned} (\Phi_{(A',F')} \circ Nat(H^g,\mu))(\alpha) &= \Phi_{(A',F')}(Nat(H^g,\mu)(\alpha)) \\ &= \Phi_{(A',F')}(\mu \circ \alpha \circ H^g) \\ \text{by definition of } \Phi_{(A',F')} &= (\mu \circ \alpha \circ H^g)_{A'}(id_{A'}) \\ &= (\mu_{A'} \circ \alpha_{A'} \circ H^g_{A'})(id_{A'}) \\ &= (\mu_{A'} \circ \alpha_{A'})(H^g_{A'}(id_{A'})) \\ &= (\mu_{A'} \circ \alpha_{A'})(H^A_g(id_A)) \\ &= (\mu_{A'} \circ \alpha_{A'} \circ H^A_g)(id_A) \\ \text{naturality of } \alpha &= (\mu_{A'} \circ Fg \circ \alpha_A)(id_A) \\ \text{naturality of } \mu &= (F'g \circ \mu_A \circ \Phi_{(A,F)})(\alpha). \end{aligned}$$

But now we are done, for Lemma 2.5.3 implies that Ψ must be a natural transformation which is an inverse for Φ. □

Exercises 2.7.5

(1) Verify that Ψ in Example 2.7.2 is a natural isomorphism.

(2) Verify that the Yoneda embedding $H : \mathcal{C} \to [\mathcal{C}^{op}, \mathcal{S}et]$ is a full and faithful functor for any locally small category $\mathcal{C}$.

(3) Let X be a preorder and let $F : X \to \mathcal{S}et$ be a functor where we will write $x \mapsto Fx$ for the operation on objects and $x \leq y \quad \mapsto \quad f_{x,y} : Fx \to Fy$ for the operation on morphisms.

(a) If Fx is the empty set $\varnothing$, what can we say about $x \in X$?

(b) Let $a \in X$. Show that to give a natural transformation $\alpha : H^a \to F$ is to give an element $e_x \in Fx$ for each $x \in X$ satisfying $a \leq x$, such that $f_{x,y}(e_x) = e_y$ whenever $y \in X$ and $x \leq y$.

(c) Investigate the Yoneda lemma in this situation.

2.8 (Bi)Cartesian Closed Categories

Discussion 2.8.1 *We begin by giving some intuition behind the idea of cartesian closed categories. Consider the category $\mathcal{S}et$. Given sets A and B, the collection of functions $A \to B$ is again a set, written $\mathcal{S}et(A,B)$ using the notation of the previous section. Thus $\mathcal{S}et(A,B)$ is actually an object of the ambient category $\mathcal{S}et$. In a cartesian closed category $\mathcal{C}$, for any objects A and B, there is an object $A \Rightarrow B$ of $\mathcal{C}$ which has properties making it "resemble" the collection of morphisms $A \to B$ in $\mathcal{C}$. Also, in the category $\mathcal{S}et$, for every pair of sets A and B, there is a function $ev : (A \Rightarrow B) \times A \to B$ defined by $ev(f,a) \stackrel{\text{def}}{=} f(a)$, for any $f : A \to B$ and $a \in A$, which " evaluates a function at an argument." Bearing this in mind, let us now give the formal definition.*

A category $\mathcal{C}$ is a ***cartesian closed category*** *if it has finite products, and for any objects B and C there is an object $B \Rightarrow C$ and morphism*

$$ev : (B \Rightarrow C) \times B \to C$$

such that for any $f : A \times B \to C$ there is a unique morphism $\lambda(f) : A \to (B \Rightarrow C)$ such that $f = ev \circ (\lambda(f) \times id_B)$. This is another example of a universal property. In this definition, the object $B \Rightarrow C$ is called the ***exponential*** *of B and C and $\lambda(f)$ is the* ***exponential mate*** *of f. We shall also write $g^\star \stackrel{\text{def}}{=} ev \circ (g \times id_B)$ for any morphism $g : A \to (B \Rightarrow C)$.*

Examples 2.8.2

(1) The category $\mathcal{S}et$ is a cartesian closed category. A specified choice of terminal object is $\{\varnothing\}$ and (specified) binary products are given by set-theoretic cartesian product. The exponential of A and B is the set of functions from A to B. The function $ev : (A \Rightarrow B) \times A \to B$ is given by $ev(f,a) = f(a)$, where $a \in A$ and $f : A \to B$ is a function. Then given any $f : X \times A \to B$ we may define $\lambda(f) : X \to (A \Rightarrow B)$ by letting $\lambda(f)(x)(a) = f(x,a)$ for each $x \in X$ and $a \in A$. It is easy to check that $f = ev(\lambda(f) \times id)$ and that $\lambda(f)$ is the unique function which satisfies this equation. Now we can begin to see the connection with functional type theory. In $\mathcal{S}et$, $\lambda(f)$ may be regarded as a "curried" version of the function f.

(2) The category $\mathcal{C}at$ of small categories is cartesian closed, with the exponential of $\mathcal{C}$ and $\mathcal{D}$ being given by the functor category $[\mathcal{C},\mathcal{D}]$. Note that $[\mathcal{C},\mathcal{D}]$ is another small category.

(3) A Heyting lattice viewed as a category is indeed cartesian closed, with Heyting implications as exponentials. In fact such a lattice also has finite coproducts.

(4) Any category $[\mathcal{C},\mathcal{S}et]$ is cartesian closed. All constructions are defined pointwise. For example, the product of F,F' in $[\mathcal{C},\mathcal{S}et]$ is given by the functor $F \times F'$ where we define $(F \times F')C \stackrel{\text{def}}{=} FC \times F'C$. The details are an exercise.

Discussion 2.8.3 *In the case that $\mathcal{C}$ is a locally small category we can give a slightly different definition of the notion of being cartesian closed. Let us introduce some more*

notation, and then present the equivalent definition as Proposition 2.8.4. Let $\mathcal{C}$ be a locally small category with finite products, let A, A', B and C be objects of $\mathcal{C}$ and $g : A' \to A$ a morphism. Define a functor $F : \mathcal{C}^{op} \to \mathcal{S}et$ as follows, where $f : A \times B \to C$ is any morphism in $\mathcal{C}$:

$$g : A' \longrightarrow A \quad \longmapsto \quad Fg : \mathcal{C}(A \times B, C) \xrightarrow{f \mapsto f(g \times id_B)} \mathcal{C}(A' \times B, C).$$

Note that we shall often write $\mathcal{C}(- \times B, C) : \mathcal{C}^{op} \to \mathcal{S}et$ for this functor. Recall that the functor $\mathcal{C}(- \times B, C)$ is representable if there is an object $B \Rightarrow C$ in $\mathcal{C}^{op}$ for which there is a natural isomorphism

$$\mathcal{C}(- \times B, C) \cong \mathcal{C}^{op}(B \Rightarrow C, -) \stackrel{\text{def}}{=} \mathcal{C}(-, B \Rightarrow C).$$

Now we have the proposition

Proposition 2.8.4 *Suppose that $\mathcal{C}$ is a locally small category with finite products and that B and C are objects of $\mathcal{C}$. If the functor $\mathcal{C}(- \times B, C)$ is representable by some object $B \Rightarrow C$ for all such B and C, then $\mathcal{C}$ is a cartesian closed category with exponentials given by $B \Rightarrow C$. Conversely, if $\mathcal{C}$ is a locally small cartesian closed category, the functor $\mathcal{C}(- \times B, C)$ is well defined and represented by the exponential $B \Rightarrow C$ for all such B and C.*

Proof

($\Rightarrow$) Suppose that $\mathcal{C}(- \times B, C)$ is represented by $B \Rightarrow C$ (for every pair of objects B and C of $\mathcal{C}$), where, say, we have a natural isomorphism denoted by

$$\Phi : \mathcal{C}(- \times B, C) \cong \mathcal{C}(-, B \Rightarrow C) : \Psi.$$

We write

$$\lambda(-) : \mathcal{C}(A \times B, C) \rightleftarrows \mathcal{C}(A, B \Rightarrow C) : (-)^{\star}$$

for the components of the natural isomorphisms at A, that is we write $\lambda(-)$ for Φ_A and $(-)^{\star}$ for Ψ_A. Thus given a morphism $f : A \times B \to C$ in $\mathcal{C}$ there is a morphism $\lambda(f) : A \to (B \Rightarrow C)$ in $\mathcal{C}$. Unraveling the definition of naturality in A at $\lambda(f)$ we get the following commutative diagram:

$$\begin{array}{ccc} \mathcal{C}(B \Rightarrow C, B \Rightarrow C) & \xrightarrow{(-)^{\star}} & \mathcal{C}((B \Rightarrow C) \times B, C) \\ \Big\downarrow{\scriptstyle \mathcal{C}(\lambda(f), B \Rightarrow C)} & & \Big\downarrow{\scriptstyle \mathcal{C}(\lambda(f) \times B, C)} \\ \mathcal{C}(A, B \Rightarrow C) & \xrightarrow[(-)^{\star}]{} & \mathcal{C}(A \times B, C) \end{array}$$

Setting $ev \stackrel{\text{def}}{=} (id_{B \Rightarrow C})^{\star}$, and using commutativity, we get

$$[\mathcal{C}(\lambda(f), B \Rightarrow C)(id_{B \Rightarrow C})]^{\star} = \mathcal{C}(\lambda(f) \times B, C)(ev)$$

that is $\lambda(f)^\star = f = ev(\lambda(f) \times id_B)$. Thus $\mathcal{C}$ is a cartesian closed category provided $\lambda(f)$ is the **unique** morphism $A \to (B \Rightarrow C)$ satisfying $f = ev(\lambda(f) \times id_B)$ for each $f : A \times B \to C$. To see that this is the case, first note that for any morphism $g : A \to (B \Rightarrow C)$ we have $g^\star = ev(g \times id_B)$, which follows from naturality of $\lambda(-)$. Now suppose also $h : A \to (B \Rightarrow C)$ is another candidate for $\lambda(f)$. Then $h = \lambda(h^\star) = \lambda(ev(h \times id_B)) = \lambda(f)$ showing uniqueness of $\lambda(f)$.

($\Leftarrow$) Now suppose that $\mathcal{C}$ is a locally small cartesian closed category. Then we claim that $\mathcal{C}(- \times B, C)$ is represented by the object $B \Rightarrow C$. Let us define a natural isomorphism

$$\Phi : \mathcal{C}(- \times B, C) \cong \mathcal{C}^{op}(B \Rightarrow C, -) \stackrel{\text{def}}{=} \mathcal{C}(-, B \Rightarrow C) : \Phi^{-1}$$

by appealing to Lemma 2.5.3, that is, we shall define isomorphisms (bijections)

$$\lambda(-) : \mathcal{C}(A \times B, C) \rightleftarrows \mathcal{C}(A, B \Rightarrow C) : (-)^\star$$

which are the components of Φ and Φ^{-1}. In fact $\lambda(-)$ is given by exponential mate in $\mathcal{C}$, and if $g : A \to (B \Rightarrow C)$ then $g^\star \stackrel{\text{def}}{=} ev(g \times id_B)$. From the universal property of exponential mates, it is easy to see that we have defined a bijection of sets, for example note that

$$g^\star = ev(g \times id_B) = ev(\lambda(g^\star) \times id_B)$$

implying that $\lambda(g^\star) = g$, and if $f : A \times B \to C$ we can show $\lambda(f)^\star = f$ similarly. We also need to see that the bijection is natural in A, which in the case of $\lambda(-) \stackrel{\text{def}}{=} \Phi_A$ amounts to the diagram

$$\begin{array}{ccc} \mathcal{C}(A \times B, C) & \xrightarrow{\lambda(-)} & \mathcal{C}(A, B \Rightarrow C) \\ \Big\downarrow{\scriptstyle \mathcal{C}(g \times B, C)} & & \Big\downarrow{\scriptstyle \mathcal{C}(g, B \Rightarrow C)} \\ \mathcal{C}(A' \times B, C) & \xrightarrow[\lambda(-)]{} & \mathcal{C}(A', B \Rightarrow C) \end{array}$$

commuting, where $g : A' \to A$ in $\mathcal{C}$. To see this, take $f : A \times B \to C$ and note that

$$\mathcal{C}(g \times B, C)(f) \stackrel{\text{def}}{=} f(g \times id_B) = ev(\lambda(f) \times id_B)(g \times id_B) = ev(\lambda(f)g \times id_B)$$

implying that $\lambda(f)g = \lambda(f(g \times id_B))$ from the universal property. Thus we have shown $\mathcal{C}(g, B \Rightarrow C)(\lambda(f)) = \lambda(\mathcal{C}(g \times B, C)(f))$ as required. □

Discussion 2.8.5 *We end this section with some notation which will prove useful when giving semantics to type theories. Let $\mathcal{C}$ be a cartesian closed category. We can define a functor $(-) \Rightarrow (+) : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$ by the assignment*

$$(f, g) : (A, B) \longrightarrow (A', B') \quad \mapsto \quad f \Rightarrow g : (A \Rightarrow B) \longrightarrow (A' \Rightarrow B')$$

where $(f,g):(A,B)\longrightarrow(A',B')$ *is of course a morphism of* $\mathcal{C}^{op}\times\mathcal{C}$*,* $A\Rightarrow B$ *and* $A'\Rightarrow B'$ *are exponentials in* $\mathcal{C}$*, and we define* $f\Rightarrow g \stackrel{\text{def}}{=} \lambda(g\circ ev\circ(id_{A\Rightarrow B}\times f))$*. Given an object* A *of* $\mathcal{C}$ *we can similarly define functors* $A\Rightarrow(-):\mathcal{C}\to\mathcal{C}$ *and* $(-)\Rightarrow A:\mathcal{C}^{op}\to\mathcal{C}$*.*

Let $F:\mathcal{C}\to\mathcal{D}$ *be a functor between cartesian closed categories which preserves finite products. We say that* F ***preserves*** *exponentials if the exponential mate of*

$$F(A\Rightarrow B)\times FA \stackrel{\cong}{\longrightarrow} F((A\Rightarrow B)\times A)\stackrel{F(ev)}{\longrightarrow} FB$$

is an isomorphism $\lambda(F(ev)\circ\cong):F(A\Rightarrow B)\to(FA\Rightarrow FB)$ *and we call this witness the* ***canonical*** *isomorphism. We shall also refer to such an* F *as a* ***cartesian closed*** *functor. A cartesian closed functor* F *is* ***strict*** *if (it is a finite product preserving functor) and* $\lambda(F(ev)\circ\cong)=id_{FA\Rightarrow FB}$*.*

Exercises 2.8.6

(1) Formulate precisely the definitions of the functors $A\Rightarrow(-):\mathcal{C}\to\mathcal{C}$ *and* $(-)\Rightarrow A:\mathcal{C}^{op}\to\mathcal{C}$*, where* A *is an object of a cartesian closed category* $\mathcal{C}$*.*

(2) Let $\mathcal{C}$ *be a cartesian closed category and* $f:A\to B$ *and* $g:B\to(C\Rightarrow D)$ *be morphisms of* $\mathcal{C}$*. Show that* $(gf)^\star=g^\star(f\times id_C)$*.*

(3) Let $f:A\times B\to C$ *and* $g:C\to D$ *be morphisms of a cartesian closed category. Show that* $\lambda(gf)=(id_B\Rightarrow g)\lambda(f)$*.*

(4) Let A *be an object of a cartesian closed category* $\mathcal{C}$*. Show that* $A\Rightarrow(-)$ *preserves finite products.*

(5) Formulate the notion of a finite coproduct preserving functor and show that $A\times(-):\mathcal{C}\to\mathcal{C}$ *is such a functor provided that* $\mathcal{C}$ *is cartesian closed.*

Discussion 2.8.7 *A category* $\mathcal{C}$ *is a* ***bicartesian closed category*** *if it is a cartesian closed category which has finite coproducts. A functor* $F:\mathcal{C}\to\mathcal{D}$ *is said to be* ***bicartesian closed*** *if it is cartesian closed and also preserves finite coproducts. We shall also call such a functor a* ***morphism*** *of bicartesian closed categories. A category with finite products and coproducts is said to be* ***distributive*** *if for all objects* A,B,C*, the mediating morphisms*

$$[id_A\times i, id_A\times j]:(A\times B)+(A\times C)\stackrel{\cong}{\longrightarrow}A\times(B+C)$$

and $!_{A\times 0}:0\stackrel{\cong}{\longrightarrow}A\times 0$ *are isomorphisms. In fact any bicartesian closed category is automatically distributive.*

Theorem 2.8.8 *Let* $\mathcal{C}$ *be a bicartesian closed category. Then* $\mathcal{C}$ *is distributive.*

Proof We show that there is a coproduct diagram

$$id_A \times i : A \times B \longrightarrow A \times (B+C) \longleftarrow A \times C : id_A \times j$$

that is, given any $f : A \times B \to D$ and $g : A \times C \to D$, there is a unique morphism $m : A \times (B+C) \to D$ such that $m \circ (id_A \times i) = f$ and $m \circ (id_A \times j) = g$. (Draw a picture!)

We have $\lambda(f) : B \to A \Rightarrow D$ and $\lambda(g) : C \to A \Rightarrow D$ and thus a unique morphism

$$[\lambda(f), \lambda(g)] : B+C \to A \Rightarrow D$$

for which $[\lambda(f), \lambda(g)] \circ i = \lambda(f)$ and $[\lambda(f), \lambda(g)] \circ j = \lambda(g)$.

Thus we have

$$\begin{array}{rcl} f & = & ev \circ (id_A \times \lambda(f)) \\ & = & ev \circ (id_A \times [\lambda(f), \lambda(g)]) \circ (id_A \times i) \\ & = & [\lambda(f), \lambda(g)]^\star \circ (id_A \times i) \end{array}$$

with a similar equation for g. Hence we can take m to be $[\lambda(f), \lambda(g)]^\star$. It is left as an *exercise* to check that if m' also makes the relevant diagram commute, then $\lambda(m') \circ i = \lambda(f)$ and $\lambda(m') \circ j = \lambda(g)$ implying that $m = m'$.

Having established the universal property of $A \times (B+C)$ it is also an *exercise* to use the property to verify that

$$[id_A \times i, id_A \times j] : (A \times B) + (A \times C) \xrightarrow{\cong} A \times (B+C)$$

In a similar fashion, one can show that $!_{A \times 0}$ is an isomorphism, with inverse π_0. □

Example 2.8.9 *The category $\mathcal{S}et$ is bicartesian closed, as is any Heyting prelattice which is regarded as a category. Any category $[\mathcal{C}, \mathcal{S}et]$ is bicartesian closed; coproducts are defined pointwise.*

3

Abstract Syntax and Rule Induction

3.1 Introduction

Discussion 3.1.1 *In this chapter we introduce the notions of abstract syntax tree, and rule induction. The two topics are not directly related. However, the applications of rule induction which occur throughout the remainder of the notes often concern definitions which are based on abstract syntax.*

3.2 Abstract Syntax Trees

Discussion 3.2.1 *Consider a conditional expression* `if true then 2 else 3` *which is a text string in a program file. Such a string does not directly give a computer the information that we have a conditional which is built out of three pieces of data, namely the Boolean and the two numbers. In a real language, it is the job of the compiler to extract such information out of textual strings. We want to represent "program expressions" in a way which makes this information explicit, without "compilation". Let us look at another example where l denotes a list*

$$\mathsf{if}\ \ \mathsf{elist}(l)\ \ \mathsf{then}\ \ \underline{0}\ \ \mathsf{else}\ \ \mathsf{hd}(l)+\mathsf{sum}(\mathsf{tl}(l))$$

and think about the structure of the expression. It has the form

$$\mathsf{if}\ B\ \mathsf{then}\ E_1\ \mathsf{else}\ E_2$$

where, for example, B is $\mathsf{elist}(l)$*. The conditional (if-then-else) expression requires three arguments, B,* E_1 *and* E_2*, and to make this clear it is helpful to write it as*

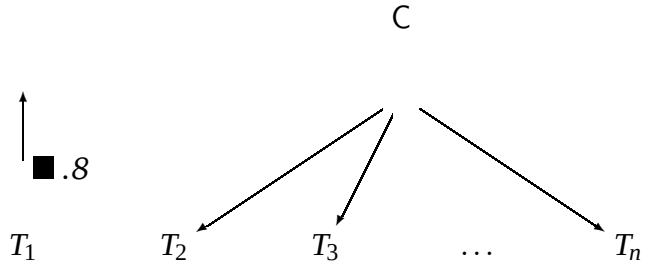
$$\mathsf{cond}(\mathsf{elist}(l)\ ,\ \underline{0}\ ,\ \mathsf{hd}(l)+\mathsf{sum}(\mathsf{tl}(l)))\quad (*)$$

and think of the cond*itional as a constructor which acts on three arguments, to "construct" a new program (you might like to think of a constructor as a function). Now we look at a sub-part of the program body,* $\mathsf{hd}(l)+\mathsf{sum}(\mathsf{tl}(l))$*. We can think of* $+$ *as a constructor which acts on two arguments, and to make this visually clear, it is convenient to write the latter expression as*

$$+(\mathsf{hd}(l)\ ,\ \mathsf{sum}(\mathsf{tl}(l))).$$

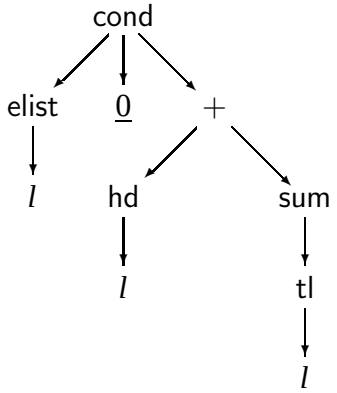
Finally, looking at one of the sub-parts of this expression, namely $\mathsf{hd}(l)$*, we can think of* $\mathsf{hd}(l)$ *as a constructor* hd *acting on a single argument, l.*

Let us make this a little clearer. We shall adopt the following notation for finite trees: If T_1, T_2, T_3 and so on to T_n is a (finite) sequence of finite trees, then we shall write $\mathsf{C}(T_1, T_2, T_3, \ldots, T_n)$ for the finite tree which has the form



*Each T_i is itself of the form $\mathsf{C}'(T'_1, T'_2, T'_3, \ldots, T'_m)$. We call C a **constructor** and say that C takes n arguments. Any constructor which takes 0 arguments is a **leaf** node. We call C the **root** node of the tree. The roots of the trees T_i are called the **children** of C. The constructors are **labels** for the nodes of the tree. Each of the T_i above is a **subtree** of the whole tree—in particular, any leaf node is a subtree.*

If we say that cond *is a constructor which takes three arguments, $+$ a constructor which takes two arguments, and so on, then $(*)$ denotes the tree*



*Note that in this (finite) tree, we regard each node as a constructor. To do this, we can think of both l and $\underline{0}$ as constructors which take no arguments!!. These form the leaves of the tree. We call the root of the tree the **outermost** constructor, and refer to trees of this kind as **abstract syntax** trees. We often refer to an abstract syntax tree by its outermost constructor—the tree above is a "conditional".*

In Part II of this course we shall be giving definitions of syntax. These definitions will be of logical propositions, and of type theory expressions. Throughout Part II, logical propositions and type theory expressions will all denote abstract syntax trees. However, we will not always use the formal notation $\mathsf{C}(T_1, T_2, T_3, \ldots, T_n)$, but shall employ so-called syntactic sugar to make things more readable. This is best explained by example. Rather than write

$$\mathsf{cond}(B\ ,\ E_1\ ,\ E_2)$$

we shall actually write

$$\mathsf{if}\ B\ \mathsf{then}\ E_1\ \mathsf{else}\ E_2$$

where it will be understood that the latter denotes the former. We say that the latter expression is ***syntactic sugar*** *for the former.*

3.3 Inductively Defined Sets

Discussion 3.3.1 *In this section we introduce a method for defining sets. Any such set will be known as an inductively defined set. Let us first introduce some notation. We shall let U be any set. A* ***rule*** *R is a pair (H,c) where $H \subseteq U$ is any finite set, and $c \in U$ is any element. Note that H might be $\varnothing$, in which case we say that R is a* ***base*** *rule. If H is non-empty we say R is an* ***inductive*** *rule. In the case that H is non-empty we might write $H = \{h_1, \ldots, h_k\}$ where $1 \leq k$. We can write down a base rule $R = (\varnothing, c)$ using the following notation*

Base

$$\frac{}{c}\,(R)$$

and an inductive rule $R = (H,c) = (\{h_1, \ldots, h_k\}, c)$ as

Inductive

$$\frac{h_1 \quad h_2 \quad \ldots \quad h_k}{c}\,(R)$$

Given a set U and a set $\mathcal{R}$ of rules based on U, a ***derivation*** *is a finite tree with nodes labelled by elements of U such that*

- *each leaf node label c arises as a base rule $(\varnothing, c) \in \mathcal{R}$*
- *for any non-leaf node label c, if H is the set of children of c then $(H,c) \in \mathcal{R}$ is an inductive rule.*

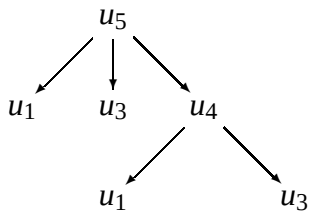
We then say that the set ***inductively defined*** *by $\mathcal{R}$ consists of those elements $u \in U$ which have a derivation with root node labelled by u.*

Examples 3.3.2

(1) Let U be the set $\{u_1, u_2, u_3, u_4, u_5, u_6\}$ and let $\mathcal{R}$ be the set of rules

$$\{R_1 = (\varnothing, u_1), R_2 = (\varnothing, u_3), R_3 = (\{u_1, u_3\}, u_4), R_4 = (\{u_1, u_3, u_4\}, u_5)\}$$

Then a derivation for u_5 is given by the tree



which is more normally written up-side down and in the following style

$$\dfrac{\dfrac{}{u_1}R_1 \qquad \dfrac{}{u_3}R_2 \qquad \dfrac{\dfrac{}{u_1}R_1 \qquad \dfrac{}{u_3}R_2}{u_4}R_3}{u_5}R_4$$

(2) A set $\mathcal{R}$ of rules for defining the set $E \subseteq \mathbb{N}$ of even numbers is $\mathcal{R} = \{R_1, R_2\}$ where

$$\frac{}{0}\,(R_1) \qquad\qquad \frac{e}{e+2}\,(R_2)$$

*Note that rule R_2 is, strictly speaking, a rule **schema**, that is e is acting as a variable. There is a "rule" for each instantiation of e. A derivation of 6 is given by*

$$\dfrac{\dfrac{\dfrac{\dfrac{}{0}\,(R_1)}{0+2}\,(R_2)}{2+2}\,(R_2)}{4+2}\,(R_2)$$

(3) The set I of integer multiples of 3 can be inductively defined by a set of rules $\mathcal{R} = \{a, b, c\}$ where

$$\frac{}{0}\,(a) \qquad \frac{i}{i+3}\,(b) \qquad \frac{i}{i-3}\,(c)$$

and informally you should think of the symbol i as a variable, that is, (b) and (c) are rule schemas.

*(4) Suppose that Σ is any set, which we think of as an **alphabet**. Each element l of Σ is called a **letter**. We inductively define the set Σ^* of **words** over the alphabet Σ by the set of rules $\mathcal{R} \stackrel{\text{def}}{=} \{1, 2\}$ (so 1 and 2 are just labels for rules!) given by*[1]

$$\frac{}{l}\,[l \in \Sigma]\ (1) \qquad\qquad \frac{w \quad w'}{ww'}\,(2)$$

[1] In rule (1), $[l \in \Sigma]$ is called a **side condition**. It means that in reading the rule, l can be any element of Σ.

Let I be inductively defined by a set of rules $\mathcal{R}$. Suppose we wish to show that a proposition $\phi(i)$ holds for all elements $i \in I$, that is, we wish to prove

$$\forall i \in I. \quad \boxed{\phi(i)}.$$

Then all we need to do is

- for every base rule $\frac{}{b} \in \mathcal{R}$ prove that $\phi(b)$ holds; and
- for every inductive rule $\frac{h_1 \ldots h_k}{c} \in \mathcal{R}$ prove that whenever $h_i \in I$,

$$(\phi(h_1) \text{ and } \phi(h_2) \text{ and } \ldots \text{ and } \phi(h_k)) \quad \text{implies} \quad \phi(c)$$

We call the propositions $\phi(h_j)$ **inductive hypotheses**. We refer to carrying out the bulleted ($\bullet$) tasks as "verifying **property closure**".

Table 3.1: Rule Induction

Suppose that $\Sigma \stackrel{\text{def}}{=} \{a, b, c\}$. *Then a derivation tree for abac is*

$$\dfrac{\dfrac{\dfrac{}{a}(1) \qquad \dfrac{}{b}(1)}{ab}(2) \qquad \dfrac{\dfrac{}{a}(1) \qquad \dfrac{}{c}(1)}{ac}(2)}{abac}(2)$$

(5) Let V be a set of ***propositional variables****. The set of (first order) propositions Prop is inductively defined by the rules below. There are two distinguished (atomic) propositions* true *and* false*. Each proposition denotes a finite tree. In fact* true *and* false *are constructors with zero arguments, as is each p. The remaining logical connectives are constructors with two arguments, and are written in a sugared (infix) notation.*

$$\frac{}{v}[v \in V] \qquad \frac{}{\mathsf{false}} \qquad \frac{}{\mathsf{true}} \qquad \frac{\phi \quad \psi}{\phi \wedge \psi} \qquad \frac{\phi \quad \psi}{\phi \vee \psi} \qquad \frac{\phi \quad \psi}{\phi \rightarrow \psi}$$

3.4 Rule Induction

Discussion 3.4.1 *In this section we see how inductive techniques of proof which the reader has met before fit into the framework of inductively defined sets. We shall write $\phi(x)$ to denote a proposition about x. For example, if $\phi(x) \stackrel{\text{def}}{=} x \geq 2$, then $\phi(3)$ is true and $\phi(0)$ is false. If $\phi(a)$ is true then we often say that $\phi(a)$* ***holds****.*

Discussion 3.4.2 *We present in Table 3.1 a useful principle called* ***Rule Induction****. It will be used throughout the remainder of these notes.*

Discussion 3.4.3 *The Principle of Mathematical Induction arises as a special case of Rule Induction. We can regard the set* $\mathbb{N}$ *as inductively defined by the rules*

$$\frac{}{0}\ (zero) \qquad \frac{n}{n+1}\ (add1)$$

Suppose we wish to show that $\phi(n)$ *holds for all* $n \in \mathbb{N}$*, that is* $\forall n \in \mathbb{N}.\phi(n)$*. According to Rule Induction, we need to verify*

- *property closure for zero, that is* $\phi(0)$*; and*
- *property closure for add1, that is for every natural number n,* $\phi(n)$ *implies* $\phi(n+1)$*, that is* $\forall n \in \mathbb{N}.\ (\phi(n)\ \text{implies}\ \phi(n+1))$

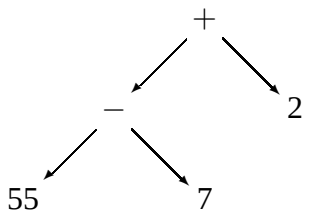
and this amounts to precisely what one needs to verify for Mathematical Induction.

Examples 3.4.4

(1) Here is another example of abstract syntax trees defined inductively. Let a set of constructors be $\mathbb{Z} \cup \{+,-\}$*. The integers will label leaf nodes, and* $+$*,* $-$ *will take two arguments written with an infix notation. The set of abstract syntax trees* $\mathcal{T}$ *inductively defined by these constructors is given by*

$$\frac{}{n} \qquad \frac{T_1 \quad T_2}{T_1 + T_2} \qquad \frac{T_1 \quad T_2}{T_1 - T_2}$$

Note that the base rules correspond to leaf nodes. In the example tree



$55-7$ *is a subtree of* $(55-7)+2$*, as are the leaves* 55*,* 7 *and* 2*.*

The principle of ***structural induction*** *is defined to be an instance of rule induction when the inductive definition is of abstract syntax trees. Make sure you understand that if* $\mathcal{T}$ *is an inductively defined set of syntax trees, to prove* $\forall T \in \mathcal{T}.\phi(T)$ *we have to prove:*

- $\phi(L)$ *for each leaf node L; and*
- *assuming* $\phi(T_1)$ *and* $\ldots$ *and* $\phi(T_n)$ *prove* $\phi(C(T_1,\ldots,T_n))$ *for each constructor C and all trees* $T_i \in \mathcal{T}$*.*

These two points are precisely property closure for base and inductive rules.

Consider the proposition $\phi(T)$ *given by* $L(T) = N(T)+1$ *where* $L(T)$ *is the number of leaves in T, and* $N(T)$ *is the number of* $+,-$*-nodes of T. We can prove by structural induction*

$$\forall T \in \mathcal{T}. \quad \boxed{L(T) = N(T)+1}$$

where the functions $L, N : \mathcal{T} \to \mathbb{N}$ are defined recursively by

- $L(n) = 1$ *and* $L(+(T_1, T_2)) = L(T_1) + L(T_2)$ *and* $L(-(T_1, T_2)) = L(T_1) + L(T_2)$
- $N(n) = 0$ *and* $N(+(T_1, T_2)) = N(T_1) + N(T_2) + 1$ *and* $N(-(T_1, T_2)) = N(T_1) + N(T_2) + 1$

This is left as an exercise.

(2) Let $\Sigma = \{a, b, c\}$ and let a set[2] S of words be defined inductively by the rules

$$\frac{}{b}\,(1) \qquad (2) \qquad \frac{w}{aaw}\,(3) \qquad \frac{w \quad w'}{ww'}\,(4)$$

Suppose that we wish to prove that every word in S has an even number of occurrences of a. Write $\#(w)$ for the number of occurrences of a in w, and

$$\phi(w) \quad \stackrel{\text{def}}{=} \quad \#(w) \text{ is even.}$$

We prove that $\forall w \in S.\phi(w)$ holds, using Rule Induction; thus we need to verify property closure for each of the rules (1) to (4):

(Rule (1)): $\#(b) = 0$, even. So $\phi(b)$ holds.

(Rule (2)): $\#(c) = 0$, even. So $\phi(c)$ holds.

(Rule (3)): Suppose that $w \in S$ is any element and $\phi(w)$ holds, that is $\#(w)$ is even (this is the inductive hypothesis). Then $\#(aaw) = 2 + \#(w)$ which is even, so $\phi(aaw)$ holds.

(Rule (4)): Suppose $w, w' \in S$ are any elements and $\#(w)$ and $\#(w')$ are even (these are the inductive hypotheses). Then so too is $\#(ww') = \#(w) + \#(w')$.

Thus by Rule Induction we are done: we have $\forall w \in S.\phi(w)$.

Discussion 3.4.5 *Sometimes it is convenient to add a rule R to a set $\mathcal{R}$, which does not alter the resulting set I. We say that a rule*

$$\frac{h_1 \quad \ldots \quad h_k}{c}\,R$$

*is a **derived** rule of $\mathcal{R}$ if there is a derivation tree whose leaves are either the conclusions of base rules or are instances of the h_i, and the conclusion is c. We shall make use of derived rules in Lemma 4.4.2.*

*The rule R is called **admissible** if one can prove*

$$(h_1 \in I) \text{ and } \ldots \text{ and } (h_k \in I) \text{ implies } (c \in I)$$

[2]Note that $S \subseteq \Sigma^*$. So any element of S is a word, but there are some words based on the alphabet Σ which are not in S.

Proposition 3.4.6 *Let I be inductively defined by $\mathcal{R}$, and suppose that R is a derived rule. Then the set I' inductively defined by $\mathcal{R} \cup \{R\}$ is also I. Any derived rule is admissible.*

Proof It is clear that $I \subset I'$. It is an exercise in rule induction to prove that $I' \subset I$. Verify property closure for each of the rules in $\mathcal{R} \cup \{R\}$, the property $\phi(i) \stackrel{\text{def}}{=} i \in I$. It is clear that derived rules are admissible. □

3.5 Recursively Defined Functions

Discussion 3.5.1 *Let I be inductively defined by a set of rules $\mathcal{R}$, and A any set. A function $f : I \to A$ can be defined by*

- *specifying an element $f(b) \in A$ for every base rule $\frac{}{b} \in \mathcal{R}$; and*
- *specifying $f(c) \in A$ in terms of $f(h_1) \in A$ and $f(h_2) \in A$ .... and $f(h_k) \in A$ for every inductive rule $\frac{h_1,\ldots,h_k}{c} \in \mathcal{R}$,*

*provided that each instance of a rule in $\mathcal{R}$ introduces a different element of I—why do we need this condition? When a function is defined in this way, it is said to be **recursively** defined.*

Examples 3.5.2

(1) The factorial function $F : \mathbb{N} \to \mathbb{N}$ is usually defined recursively. We set

- $F(0) \stackrel{\text{def}}{=} 1$ *and*
- $\forall n \in \mathbb{N}.F(n+1) \stackrel{\text{def}}{=} (n+1) * F(n)$.

Thus $F(3) = (2+1) * F(2) = 3 * 2 * F(1) = 3 * 2 * 1 * F(0) = 3 * 2 * 1 * 1 = 6$. Are there are brackets missing from the previous calculation? If so, insert them.

(2) Consider the propositions defined on page 48. Suppose that ψ_i and x_i are propositions and propositional variables for $1 \leq i \leq n$. Then there is a recursively defined function Prop $\to$ Prop whose action is written $\phi \mapsto \phi[\psi_1,\ldots,\psi_n/x_1,\ldots,x_n]$ which computes the simultaneous substitution of the ϕ_i for the x_i where the x_i are distinct. We set

- $x[\psi_1,\ldots,\psi_n/x_1,\ldots,x_n] \stackrel{\text{def}}{=} \psi_j$ *if x is x_j;*
- $x[\psi_1,\ldots,\psi_n/x_1,\ldots,x_n] \stackrel{\text{def}}{=} x$ *if x is none of the x_i;*
- $$(\phi \wedge \phi')[\psi_1,\ldots,\psi_n/x_1,\ldots,x_n] \stackrel{\text{def}}{=} (\phi[\psi_1,\ldots,\psi_n/x_1,\ldots,x_n]) \wedge (\phi'[\psi_1,\ldots,\psi_n/x_1,\ldots,x_n])$$
- *The other clauses are similar.*

Discussion 3.5.3 *More generally we shall use the notion of simultaneous substitution on syntax trees with variable binding operations. We shall assume that the reader has some knowledge of the concepts of free and bound variables. We illustrate the definition of* ***simultaneous substitution*** *via a simple example. Let* F *be a constructor with one argument, and* B *of two arguments. Consider the syntax trees defined by*

$$\frac{}{v} \qquad \frac{T}{\mathsf{F}(T)} \qquad \frac{T}{\mathsf{B}(v,T)}$$

where v *ranges over a set of variables, and free occurrences of* v *in* T *are bound in* $\mathsf{B}(v,T)$*. Write* $[\vec{T}/\vec{v}]$ *for* $[T_1,\ldots,T_n/v_1,\ldots,v_n]$ *with the* v_i *all distinct. Then we define by recursion*

- $v[\vec{T}/\vec{v}] \stackrel{\text{def}}{=} T_i$ *if* $v = v_i$
- $v[\vec{T}/\vec{v}] \stackrel{\text{def}}{=} v$ *if* v *is none of the* v_i
- $\mathsf{F}(T)[\vec{T}/\vec{v}] \stackrel{\text{def}}{=} \mathsf{F}(T[\vec{T}/\vec{v}])$
- $\mathsf{B}(v,T)[\vec{T}/\vec{v}] \stackrel{\text{def}}{=} \mathsf{B}(v',T'[\vec{T}/\vec{v}])$ *where* v' *is a fresh variable distinct from all variables in* T*,* v*,* $\vec{T}$ *and* $\vec{v}$*, and* T' *is* T *with all free occurrences of* v *changed to* v'*.*

Note that this definition, while quite adequate for these notes, is rather inefficient from the point of an implementation. Also, we are cheating a little, in the sense that T' *has not been defined formally. All of these matters can be made rigorous—but the definitions are surprisingly tricky to get right, and quite lengthy, so we omit them.*

Part II

Categorical Logic and Type Theory

4

Categorical Propositional Logic

4.1 Intuitionistic Propositional Logic

Discussion 4.1.1 *In this chapter we shall study (intuitionistic) propositional logic. Let Gnd be a set of* ***ground propositions****. Each such proposition is thought of as a "basic" or "atomic" proposition. The set of (first order) propositions Prop is inductively defined by the rules in Table 4.1. There are two distinguished (atomic) propositions* true *and* false*. Each proposition denotes an abstract syntax tree—see Chapter 3.*

We shall require the notion of a *finite list*. For the purposes of these notes, a (finite) **list** over a set S is an element of the set

$$[S] \stackrel{\text{def}}{=} \bigcup_{n<\omega} S^n$$

where S^n is the set of n-tuples of S and $S^0 \stackrel{\text{def}}{=} \{\,\mathsf{nil}\,\}$ where nil denotes the empty list. We shall denote a typical element of $[S]$ by $s_1,\ldots,s_n$ and we shall write $s \in L$ to indicate that s occurs in the list (tuple) L. We shall also write $Set(L)$ for the set $\{\ l \mid l \in L\ \}$. For example (taking S to be $\mathbb{N}$) we have $Set(2,4,3,4,4,3) = \{2,3,4\}$.

Discussion 4.1.2 *An IpL* ***sequent****, $\Delta \vdash \phi$, is a pair consisting of a finite list Δ of propositions, together with a proposition ϕ. We shall adopt some (standard) notation. We use , to denote list concatenation. Thus if $\Delta \in Prop^n$, then Δ,ϕ stands for the (expected) concatenated list in $Prop^{n+1}$. If $\Delta \in Prop^n$ and $\Delta' \in Prop^m$ then $\Delta,\Delta' \in Prop^{n+m}$.*

*An IpL-****signature*** *Sg is specified by giving a set of ground propositions Gnd. An IpL-****theory*** *Th is a pair (Sg,Ax) where Sg is an IpL-signature and Ax is a set of sequents. Each such sequent is called an* ***axiom*** *of Th. Given a theory Th, the theorems of Th consist of the sequents that are inductively generated by the rules in Table 4.2. If $\Delta \vdash \phi$ is a theorem (that is, it has a derivation using the rules) then we shall sometimes write $Th \ \rhd\ \Delta \vdash \phi$.*

$$\frac{}{p}\ [p \in Gnd] \qquad \frac{}{\mathsf{true}} \qquad \frac{}{\mathsf{false}} \qquad \frac{\phi \quad \psi}{\phi \wedge \psi} \qquad \frac{\phi \quad \psi}{\phi \vee \psi} \qquad \frac{\phi \quad \psi}{\phi \rightarrow \psi}$$

Table 4.1: Inductive Definition of Propositions

$$\frac{}{\Delta \vdash \phi}\,[\Delta \vdash \phi \in Ax] \qquad \frac{\Delta,\phi,\phi',\Delta' \vdash \psi}{\Delta,\phi',\phi,\Delta' \vdash \psi}\,\text{EXCH} \qquad \frac{\Delta,\phi,\phi,\Delta' \vdash \psi}{\Delta,\phi,\Delta' \vdash \psi}\,\text{CTRN}$$

$$\frac{}{\Delta',\phi,\Delta \vdash \phi}\,\text{ID} \qquad \frac{\Delta \vdash \phi \quad \phi,\Delta' \vdash \psi}{\Delta',\Delta \vdash \psi}\,\text{CUT}$$

$$\frac{}{\Delta \vdash \mathsf{true}}\,\text{TRUE-I} \qquad \frac{\Delta \vdash \mathsf{false}}{\Delta \vdash \psi}\,\text{FALSE-E}$$

$$\frac{\Delta \vdash \phi \quad \Delta \vdash \psi}{\Delta \vdash \phi \wedge \psi}\,\text{AND-I} \qquad \frac{\Delta \vdash \phi \wedge \psi}{\Delta \vdash \phi}\,\text{AND-E}_l \qquad \frac{\Delta \vdash \phi \wedge \psi}{\Delta \vdash \psi}\,\text{AND-E}_r$$

$$\frac{\Delta \vdash \phi}{\Delta \vdash \phi \vee \psi}\,\text{OR-E}_l \qquad \frac{\Delta \vdash \psi}{\Delta \vdash \phi \vee \psi}\,\text{OR-E}_l \qquad \frac{\Delta,\phi \vdash \theta \quad \Delta,\psi \vdash \theta \quad \Delta \vdash \phi \vee \psi}{\Delta \vdash \theta}\,\text{OR-E}_r$$

$$\frac{\Delta,\phi \vdash \psi}{\Delta \vdash \phi \to \psi}\,\text{IMP-I} \qquad \frac{\Delta \vdash \phi \to \psi \quad \Delta \vdash \phi}{\Delta \vdash \psi}\,\text{IMP-E}$$

Table 4.2: Inductive Definition of *IpL* Theorems

Exercise 4.1.3 *Refer to Section 3.3.1. For the inductive definition in Table 4.2, work out precisely what I and U are.*

4.2 Deriving a Categorical Semantics

Discussion 4.2.1 *We shall consider how to give a semantics to the logic IpL. In particular we shall work out how to give a semantics to a theory $Th = (Sg, Ax)$. We have an informal understanding of a theorem $\Delta \vdash \phi$, namely that if all of the propositions in Δ are valid, then it follows that ϕ is valid. Of course, there are many ways of making the meaning of "valid" precise, and these fall under the topics of "pure" logic, and philosophy. Here we shall assume that the reader has a very basic knowledge of (first order) logic, and we will not pursue a general discussion of validity.*

We shall look for a mathematical "space" (which we shall call H) in which we can interpret, or model, the IpL propositions. For example, H might be a topological space, group, category, or a poset. We shall assume that H has some notion of "element" (which is very clear in the case of, for example, a poset) and will write statements like $h \in H$, but we will not give a precise definition of this notation for the time being.

We shall model a proposition ϕ as an element $[\![\phi]\!] \in H$. We call $[\![\phi]\!]$ the ***denotation*** *of*

ϕ. *We sometimes refer to* $[\![\phi]\!]$ *as the* ***meaning*** *of* ϕ. *The reader should recall the classical denotations of propositions,* $[\![\phi]\!] \in \mathbb{B}$, *where* $\mathbb{B}$ *is the set of Booleans. Our semantics will generalize these ideas. For the time being, we shall assume that* $[\![\phi]\!]$ *is defined for every* ϕ, *and return to the actual definition later on.*

A sequent of the form $\phi \vdash \psi$ *can be thought of as a "relationship" between* ϕ *and* ψ. *So we can capture this idea in our model by considering binary relations over* H. *We shall denote these relations by* $\leq$ *but for the time being we shall make no assumptions about the properties satisfied by* $\leq$. *The minimal requirement of our semantics should be that that* $[\![\phi]\!] \leq [\![\psi]\!]$ *whenever* $Th \rhd \phi \vdash \psi$.

What about theorems of the form $\Delta \vdash \phi$? *The context is a list of propositions, and each proposition has a denotation in* H. *It seems reasonable that the denotation of* Δ *should depend on the denotations of each of the propositions in* Δ. *Let us write* $\overline{\Delta}$ *for the finite list of denotations of the propositions in* Δ, *an element of* $[H]$. *We can then look for a function* $[H] \to H$ *whose effect will be written* $L \mapsto \Box(L)$, *so that we can define the denotation of* Δ *to be* $[\![\Delta]\!] \stackrel{\text{def}}{=} \Box(\overline{\Delta})$. *(To be more precise, note that a function* $[H] \to H$ *is specified by a family of functions* $(\Box(-)_n : H^n \to H \mid n < \omega)$. *However, we shall overload notation, writing* $L \mapsto \Box(L)$ *for any* $L \mapsto \Box(L)_n$.*) In the case that* Δ *is simply* ϕ, *we would expect that* $[\![\Delta]\!] = [\![\phi]\!]$. *Thus we will assert that* $\Box(h) \stackrel{\text{def}}{=} h$ *for any* $h \in H$. *The minimal requirement of our semantics should be that* $[\![\Delta]\!] \leq [\![\phi]\!]$ *whenever* $Th \rhd \Delta \vdash \phi$. *In such a case we say that the theorem is* ***satisfied*** *by the semantics.*

We return to the denotation of propositions. It is reasonable to require the denotational operation $- \mapsto [\![-]\!]$ *to be defined by recursion using the inductive definition of propositions. Thus we shall write* $\boxed{\#} : H \times H \to H$ *for a function which gives the semantics of a logical operator #, so that we may define*

- $[\![\mathsf{true}]\!] \stackrel{\text{def}}{=} \top \in H$, *an element to be determined;*
- $[\![\mathsf{false}]\!] \stackrel{\text{def}}{=} \bot \in H$, *an element to be determined;*
- $[\![\phi \wedge \psi]\!] \stackrel{\text{def}}{=} [\![\psi]\!] \boxed{\wedge} [\![\psi]\!]$;
- $[\![\phi \vee \psi]\!] \stackrel{\text{def}}{=} [\![\psi]\!] \boxed{\vee} [\![\psi]\!]$; *and*
- $[\![\phi \to \psi]\!] \stackrel{\text{def}}{=} [\![\psi]\!] \boxed{\to} [\![\psi]\!]$,

where the denotation $[\![p]\!]$ *of any ground proposition may be chosen to be any element of* H.

The axioms of a theory assert that certain sequents are to be regarded as "fundamental" facts. The rules for inductively defining theorems allow us to derive new facts based on the fundamental facts. This leads to the definition of a structure for a signature Sg and a model of a theory. A ***structure*** $\mathbf{M}$ *in* H *for a signature is given by specifying an element* $[\![p]\!]_{\mathbf{M}} \in H$ *for each* $p \in Gnd$. *One can then define* $[\![\phi]\!]_{\mathbf{M}}$ *by recursion as above, and a* ***model*** *of Th is a structure* $\mathbf{M}$ *which satisfies each of the axioms of Th. We shall now look for conditions*

on $(H, \leq, \Box(-), \boxed{\#}, \bot, \top)$ *which ensure that for any theory Th and for any model* $\mathbf{M}$*, the theorems are all satisfied. We shall attempt to discover necessary and sufficient conditions. Consider any sequent* $\Delta \vdash \phi$*, not just a theorem, which is also said to be satisfied by* $\mathbf{M}$ *if* $[\![\Delta]\!] = \Box(\overline{\Delta}) \leq [\![\phi]\!]$*. A typical rule for deducing theorems looks like*

$$\frac{\Delta_1 \vdash \phi_1 \qquad \ldots \qquad \Delta_n \vdash \phi_n}{\Delta \vdash \phi} R$$

In order to ensure that all theorems are satisfied, we want to find necessary and sufficient conditions on all such rules R to ensure that for all Δ_i*,* ϕ_i*,* Δ *and* ϕ

$$(\Box(\overline{\Delta_1}) \leq [\![\phi_1]\!]) \text{ and } \ldots \text{ and } (\Box(\overline{\Delta_n}) \leq [\![\phi_n]\!]) \text{ implies } \Box(\overline{\Delta}) \leq [\![\phi]\!]$$

If this holds, we shall say that the semantics is sound for the rule. It is clearly sufficient to require that for all $L_i, L \in [H]$ *and* $h_i, h \in H$

$$(\Box(L_1) \leq h_1) \text{ and } \ldots \text{ and } (\Box(L_n) \leq h_n) \text{ implies } \Box(L) \leq h \qquad (*)$$

However, $(*)$ *is also necessary. Each sequent* $\Delta_j \vdash \phi_j$ *can take the form*

$$Ax \ \rhd \ p_1, \ldots, p_m \vdash p$$

As the satisfaction of such axioms must range over all possible models, the images $([\![p_1]\!], \ldots, [\![p_m]\!], [\![p]\!])$ *must be onto* $[H] \times H$*. Thus in fact it is necessary that* $(*)$ *holds.*

We shall consider each of the rules for deriving theorems in turn, and gradually establish necessary and sufficient conditions (NSC) for soundness. Before that we shall deduce a couple of facts by looking at restricted instances of rules. First note that there are instances of the rule ID *of the form* $\phi \vdash \phi$*. As* ϕ *could be any ground proposition, it is necessary that for any* $h \in H$ *we have* $h \leq h$*, that is* $\leq$ *must be reflexive. There are instances of the rule* CUT *of the form*

$$\frac{\delta \vdash \phi \quad \phi \vdash \psi}{\delta \vdash \psi}$$

Here, each sequent could take the form $p \vdash q$*, because the two hypotheses could be axioms of this form. Thus it is necessary that* $\leq$ *be transitive. So* $\leq$ *must satisfy the axioms of a preorder, or, put another way, H and* $\leq$ *together form a category in which there is at most one morphism* $h \leq h'$ *between any two objects* h, h' *in H. Although we do not have conditions which force H to be a set, we shall keep this discussion simple by requiring this. Thus* $(H, \leq)$ *is a preordered set, and we can talk about the notions of bounds, meets, joins and so on within such a standard framework.*

$\boxed{ID}$ *The NSC for soundness are that* $\Box(L, h, L') \leq h$ *for all* $L, L' \in [H]$ *and* $h \in H$*. Equivalently, the NSC are that* $\Box(L)$ *is a lower bound of the set* $Set(L)$ *for any* $L \in [H]$*.*

$\boxed{EXCH}$ *The NSC are that for any* L, L', h, h', k *we have*

$$\Box(L, h, h', L') \leq k \text{ implies } \Box(L, h', h, L') \leq k$$

and thus simpler but equivalent NSC are

$$\Box(L,h,h',L') \cong \Box(L,h',h,L')$$

$\boxed{\textit{CTRN}}$ *In a similar fashion to the previous rule, we can deduce that NSC are for any* L,L',h,h' *we have*

$$\Box(L,h,h,L') \cong \Box(L,h,L')$$

$\boxed{\textit{CUT}}$ *The NSC are that for any* $L,L' \in [H]$ *and* $k,k' \in H$*, if* $\Box(L) \leq k$ *and* $\Box(k,L') \leq k'$ *then* $\Box(L,L') \leq k'$*. Taking* $L \stackrel{\text{def}}{=} h$ *for any* $h \in H$ *and* $k' \stackrel{\text{def}}{=} \Box(k,L')$*, we can deduce that if* $h \leq k$ *then* $\Box(h,L') \leq \Box(k,L')$*. Given the NSC from* EXCH*, it is easy to see that* $\Box(-)$ *is in fact monotone in each argument—short exercise. So suppose now that* $L \stackrel{\text{def}}{=} k_1,\ldots,k_n$ *and for each i,* $h \leq k_i$*. Then we can calculate, using the NCS from* CTRN *for the first step,*

$$\begin{array}{rcl} h = \Box(h) & \cong & \Box(h,\ldots,h,h) \\ & \leq & \Box(k_1,\ldots,k_{n-1},k_n) \end{array}$$

Hence we see that $\Box(L)$ *is a greatest lower bound (meet) of the finite set* $Set(L)$*, which is determined uniquely up to isomorphism because* H *is a preordered set. In fact in the next section, we shall define* $\Box(L)$ *to be* $\bigwedge Set(L)$ *for any* $L \in [H]$*. For the time being we shall stick to the formal box notation on lists, and make use of the properties we have deduced.*

$\boxed{\textit{OR-I}_l}$ *and* $\boxed{\textit{OR-I}_r}$ *and* $\boxed{\textit{OR-E}}$ *NSC for the soundness of these rules are*

$$\begin{array}{rcl} \Box(L) \leq h & \text{implies} & \Box(L) \leq h \boxed{\vee} k \\ \Box(L) \leq k & \text{implies} & \Box(L) \leq h \boxed{\vee} k \\ (\Box(L,h) \leq l \text{ and } \Box(L,k) \leq l \text{ and } \Box(L) \leq h \boxed{\vee} k) & \text{implies} & \Box(L) \leq l \end{array}$$

By taking L to be h and k in the first two implications respectively, we can see that we must have $h \vee k \leq h \boxed{\vee} k$*. Now take L to be* $h \boxed{\vee} k$ *and l to be* $h \vee k$ *in the third implication. Note that* $\Box(h \boxed{\vee} k,h) \leq h \leq h \vee k$*, similarly* $\Box(h \boxed{\vee} k,k) \leq k \leq h \vee k$*, and hence that* $h \boxed{\vee} k \leq h \vee k$*. Thus up to isomorphism,* $h \boxed{\vee} k$ *is given by join. However, this is not sufficient to ensure that the third implication always holds. In fact the joins in H must be distributive for meets, that is*

$$\Box(L',h' \boxed{\vee} k') \leq \Box(L',h') \boxed{\vee} \Box(L',k') \qquad (*)$$

must always hold. To see that this is sufficient for soundness of the third implication, note that $\Box(L) \leq \Box(L,h \boxed{\vee} k)$ *if* $\Box(L) \leq h \boxed{\vee} k$*, and thus using* $(*)$ *we have* $\Box(L) \leq l \boxed{\vee} l \leq l$*. To see necessity, just take L to be* $\Box(L',h' \boxed{\vee} k')$ *and l to be* $\Box(L',h') \boxed{\vee} \Box(L',k')$*, along with* h' *for h and* k' *for k. It is an exercise to check that the hypotheses of the third implication hold, and thus we can deduce*

$$\Box(L',h' \boxed{\vee} k') \leq \Box(L',h') \boxed{\vee} \Box(L',k')$$

and hence that $*$ *holds.*

$$\frac{}{[\![p]\!] \text{ is specified}} \qquad \frac{}{[\![\mathsf{true}]\!] \stackrel{\text{def}}{=} \top} \text{ (where } \top \in H \text{ is the top element)}$$

$$\frac{}{[\![\mathsf{false}]\!] \stackrel{\text{def}}{=} \bot} \text{ (where } \bot \in H \text{ is the bottom element)}$$

$$\frac{[\![\phi]\!] = h \quad [\![\psi]\!] = k}{[\![\phi \wedge \psi]\!] = h \wedge k} \qquad \frac{[\![\phi]\!] = h \quad [\![\psi]\!] = k}{[\![\phi \vee \psi]\!] = h \vee k} \qquad \frac{[\![\phi]\!] = h \quad [\![\psi]\!] = k}{[\![\phi \to \psi]\!] = h \Rightarrow k}$$

Table 4.3: Recursive Definition of *IpL* Semantics

IMP-I and IMP-E *The NSC for IMP-I are that if $\Box(L,h) \leq k$ then $L \leq h \boxed{\to} k$, and that if $L \leq h \boxed{\to} k$ and $L \leq h$ then $L \leq k$.*

From the second condition, it follows by taking L to be $\Box(h, h \boxed{\to} k)$ that

$$\Box(h, h \boxed{\to} k) \leq k \qquad (\dagger)$$

Now suppose that $L \leq h \boxed{\to} k$. Then it follows from the monotonicity of $\Box(-)$, and $(\dagger)$, that $\Box(L,h) \leq k$. Hence (recall the first condition) $\Box(L,h) \leq k$ just in case $L \leq h \boxed{\to} k$. Thus $+ \boxed{\to} -$ is determined uniquely up to isomorphism as Heyting implication—see page 12.

We conclude that $(H, \leq)$ is a Heyting prelattice.

Exercises 4.2.2

(1) Work out NSC for the soundness of the three rules AND-I and AND-E$_l$ and AND-E$_r$.

(2) Work out NSC for the soundness of the rules TRUE-I and FALSE-E.

4.3 Categorical Semantics and the Soundness Theorem

Discussion 4.3.1 *Let Gnd be a set of ground propositions, and* **M** *a structure for Gnd in a Heyting prelattice H. Then the semantics of propositions is given by recursion using the inductive rules for deriving propositions, and the definition is given in Table 4.3. The structure* **M** ***satisfies*** *a sequent $\Delta \vdash \phi$ if*

$$[\![\Delta]\!] \stackrel{\text{def}}{=} \bigwedge Set(\overline{\Delta}) = \bigwedge \{\, [\![\delta]\!] \mid \delta \in \Delta \,\} \leq [\![\phi]\!]$$

Recall that **M** *is a* ***model*** *of $Th = (Sg, Ax)$ if it satisfies each of the axioms in Ax. We now have a soundness theorem.*

Theorem 4.3.2 *Let $Th = (Sg, Ax)$ be an IpL-theory and* **M** *a model of Th in a Heyting prelattice. Then* **M** *satisfies each of the theorems of Th.*

Proof The details of the proof appear implicitly in Section 4.2. It is an *exercise* to verify the details explicitly. □

Examples 4.3.3 *For examples of Heyting prelattices (and lattices) see Section 1.3.*

4.4 Classifying Preorders and the Completeness Theorem

Discussion 4.4.1 *A classifying preorder for any IpL-theory is a Heyting prelattice which is, in some sense, the "least" such preorder in which there is a model of Th. We shall give a definition of classifying preorder which determines it uniquely up to isomorphism of preordered sets, and characterizes it by a universal property. Moreover, we shall show how to give an explicit construction of such a preorder. First, we need some preliminary definitions.*

Suppose that we are given a homomorphism of Heyting prelattices, say $f : H \to K$*, and a model* $\mathbf{M}$ *of a theory* $Th = (Sg, Ax)$ *in* H*. We shall show how to define a new model* $f_*\mathbf{M}$*, namely one of Th in K. We can define a structure* $f_*\mathbf{M}$ *for Sg in K by specifying* $[\![p]\!]_{f_*\mathbf{M}} \stackrel{\text{def}}{=} f([\![p]\!]_{\mathbf{M}}) \in K$*. In fact this structure is indeed a model. Firstly one can prove by rule induction over the definitions of the propositions that for any* ϕ *we have* $[\![\phi]\!]_{f_*\mathbf{M}} \cong f([\![\phi]\!]_{\mathbf{M}})$*; more precisely we prove*

$$\forall \phi \in Prop. \quad \boxed{[\![\phi]\!]_{f_*\mathbf{M}} \cong f([\![\phi]\!]_{\mathbf{M}})}$$

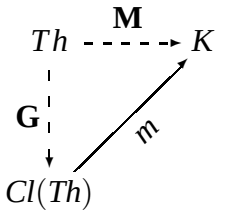
by rule induction. Property closure for the base rules is trivial in the case of ground propositions, and holds (immediately) for the rules for true *and* false *because* f *is a homomorphism. Property closure for the inductive rules relies on* f *being a homomorphism, and the easy fact (Lemma 1.2.5) that computing meets preserves isomorphisms. Now suppose that* $Ax \rhd \Delta \vdash \phi$*. Then we have*

$$[\![\Delta]\!]_{f_*\mathbf{M}} \stackrel{\text{def}}{=} \bigwedge\{\, [\![\delta]\!]_{f_*\mathbf{M}} \mid \delta \in \Delta \,\} \cong \bigwedge\{\, f([\![\delta]\!]_{\mathbf{M}}) \mid \delta \in \Delta \,\} \cong'$$

$$f(\bigwedge\{\, [\![\delta]\!]_{\mathbf{M}} \mid \delta \in \Delta \,\}) \leq f([\![\phi]\!]_{\mathbf{M}}) \cong [\![\phi]\!]_{f_*\mathbf{M}}$$

where the isomorphisms $\cong$ *hold by the fact proved above, and* $\cong'$ *holds because* f *is a homomorphism. Thus* $f_*\mathbf{M}$ *satisfies the axioms of Th too, and is thus a model of Th in K.*

Let Th be a IpL-theory. A Heyting prelattice $Cl(Th)$ *is called the* ***classifying*** *prelattice of Th if there is a model* $\mathbf{G}$ *of Th in* $Cl(Th)$ *for which given any Heyting prelattice K, and a model* $\mathbf{M}$ *of Th in K, then there is a homomorphism of Heyting prelattices* $m : Cl(Th) \to K$ *such that* $m_*\mathbf{G} = \mathbf{M}$*, and moreover any two such m are naturally isomorphic.*



where $m_*\mathbf{G} = \mathbf{M}$.

Thus $Cl(Th)$ is defined up to isomorphism by a universal property.

Before stating a theorem concerning the existence of such classifiers, we have a preliminary lemma. The lemma states that, given an IpL-theory Th, the collection of theorems inductively defined by the set of rules in Table 4.2, can in fact be inductively defined by an alternative set of rules in Table 4.4. These latter rules form what is known as an ***adjoint calculus****. This is because the definitions of meets, joins and Heyting implications are all examples of the categorical notion of adjunction—these notes do not make use of adjunctions, and we will not pursue such ideas here. However, the reader should notice that the (adjoint) rules for conjunction, disjunction and implication are virtually mirror images of the definitions of meet, join and Heyting implication.*

Lemma 4.4.2 *The class of theorems inductively defined by the set of rules in Table 4.2 is exactly the same as that defined by the rules in Table 4.4. Note that*

$$\frac{s_1 \quad \ldots \quad s_n}{\quad s \quad}$$

abbreviates the $n+1$ rules

$$\frac{s_1 \quad \ldots \quad s_n}{\quad s \quad} \qquad \frac{s}{s_i}$$

Proof Each rule found in one inductive definition is a *derived* rule of the other system. The proof of this is routine manipulation of the rules, but it may require a little thought. □

Theorem 4.4.3 *Each IpL-theory Th has a classifying Heyting prelattice $Cl(Th)$. In fact we can construct a* ***canonical*** *classifier using the syntax of Th, where $m_* \mathbf{G} = \mathbf{M}$.*

Proof Given $Th = (Sg, Ax)$, define a relation $\leq$ on the set *Prop* of propositions by $\phi \leq \psi$ if and only if $Th \rhd \phi \vdash \psi$. Then using Lemma 4.4.2 we can easily see that $(Prop, \leq)$ is a Heyting prelattice; the rules yield the existence of binary meets and joins, which is equivalent to (non-empty) finite meets and joins, and it is an *exercise* to check the distributivity requirement (hint: use $\vdash$ OR-E backwards). Existence of Heyting implication, top, and bottom elements is also immediate.

We can define a structure $\mathbf{G}$ for Sg in $Cl(Th)$ by setting $[\![p]\!]_{\mathbf{G}} \stackrel{\text{def}}{=} p$. It follows trivially that $[\![\phi]\!]_{\mathbf{G}} = \phi$, and hence that $\mathbf{G}$ satisfies the axioms and so is a model.

Now let K be a Heyting prelattice and $\mathbf{M}$ a model of Th in K. We shall show that there is a unique (up to isomorphism) homomorphism of Heyting prelattices $m : Cl(Th) \to K$ such that $\mathbf{M} = m_* \mathbf{G}$.

$$\frac{}{\Delta \vdash \phi}\,[\Delta \vdash \phi \in Ax] \qquad \frac{\Delta,\phi,\phi',\Delta' \vdash \psi}{\Delta,\phi',\phi,\Delta' \vdash \psi}\,\text{EXCH} \qquad \frac{\Delta,\phi,\phi,\Delta' \vdash \psi}{\Delta,\phi,\Delta' \vdash \psi}\,\text{CTRN}$$

$$\frac{}{\Delta',\phi,\Delta \vdash \phi}\,\text{ID} \qquad \frac{\Delta \vdash \phi \quad \phi,\Delta' \vdash \psi}{\Delta',\Delta \vdash \psi}\,\text{CUT}$$

$$\frac{}{\Delta,\mathsf{false} \vdash \phi} \dashv \text{FALSE-E} \qquad \frac{}{\Delta \vdash \mathsf{true}} \dashv \text{TRUE-I}$$

$$\frac{\Delta \vdash \phi \quad \Delta \vdash \psi}{\Delta \vdash \phi \wedge \psi} \dashv \text{AND-I}$$

$$\frac{\Delta,\phi \vdash \theta \quad \Delta,\psi \vdash \theta}{\Delta,\phi \vee \psi \vdash \theta} \dashv \text{OR-E}$$

$$\frac{\Delta,\phi \vdash \psi}{\Delta \vdash \phi \to \psi} \dashv \text{IMP-I}$$

Table 4.4: Inductive Definition of *IpL* Theorems by Adjoint Rules

We define $m : Cl(Th) \to K$ by $\phi \mapsto [\![\phi]\!]_{\mathbf{M}}$. Suppose that $\phi \leq \psi$ in $Cl(Th)$. Then by Theorem 4.3.2, and the definition of $\leq$, we have $m(\phi) = [\![\phi]\!]_{\mathbf{M}} \leq [\![\psi]\!]_{\mathbf{M}} = m(\psi)$ in K. Thus m is monotone. Also, given the definition of $[\![\phi]\!]_{\mathbf{M}}$ for arbitrary ϕ, m is trivially a homomorphism of Heyting prelattices. Finally for each ground proposition p, we have

$$[\![p]\!]_{\mathbf{M}} = m(p) = m([\![p]\!]_{\mathbf{G}}) = [\![p]\!]_{m_* \mathbf{G}}$$

and thus $\mathbf{M} = m_* \mathbf{G}$.

Suppose that there is another homomorphism of Heyting prelattices $m' : Cl(Th) \to K$ for which $m'_* \mathbf{G} = \mathbf{M}$. If ϕ is an element of $Cl(Th)$ then

$$m(\phi) \stackrel{\text{def}}{=} [\![\phi]\!]_{\mathbf{M}} = [\![\phi]\!]_{m'_* \mathbf{G}} \cong m'([\![\phi]\!]_{\mathbf{G}}) = m'(\phi)$$

for any ϕ, and hence $m \cong m'$. □

Discussion 4.4.4 *We have seen that the semantics is sound, meaning that any theorem of a theory is satisfied by a model. The converse notion to soundness is called completeness. The semantics for a theory Th is said to be* ***complete*** *if a sequent S of Th is a theorem whenever for all Heyting prelattices H, S is satisfied by all models* $\mathbf{M}$ *in H.*

Theorem 4.4.5 *The semantics of IpL-theories in Heyting prelattices is both sound and complete.*

Proof *Soundness was proved in Theorem 4.3.2. Conversely, let $\Delta \vdash \phi$ be a sequent for a theory Th. If it is satisfied by all models, in particular it is satisfied by the generic model of Th in $Cl(Th)$. Thus we have $\bigwedge Set(\Delta) \leq \phi$ in $Cl(Th)$. Hence*

$$Th \ \rhd \ (\ldots((\delta_1 \wedge \delta_2) \wedge \delta_3)\ldots) \wedge \delta_n \vdash \phi$$

where $\delta_i \in Set(\Delta)$, from which we may deduce (care—exercise to check the details!) that $Th \ \rhd \ \Delta \vdash \phi$ as required. □

5

Categorical Type Theory

5.1 Type Theory with Products, Sums and Functions

Discussion 5.1.1 *We shall define the notion of a signature for a type theory, which consists of basic data from which to build types and terms.*

A $\lambda\times+$***-signature**, Sg, is given by the following data:*

• *A collection of* ***ground types**. The collection of types is inductively defined by the rules in Table 5.1, where* γ *is any ground type. We call* $\sigma\times\tau$ *a binary product type,* $\sigma+\tau$ *a binary sum type, and* $\sigma\Rightarrow\tau$ *a function type.*

• *A collection of* ***function symbols** each of which has an* ***arity** which is a natural number (possibly* 0*).*

• *A* ***sorting** for each function symbol* f*, which is a non-empty list* $[\sigma_1,\ldots,\sigma_a,\sigma]$ *of* $a+1$ *types, where* a *is the arity of* f*. We shall write* $f:\sigma_1\ldots\sigma_a\to\sigma$ *to denote the sorting of* f*. In the case that* k *is a function symbol of arity* 0 *we shall denote the sorting by* $k:\sigma$ *and the function symbol* k *will be referred to as a* ***constant** of type* σ*.*

We can now inductively define the ***raw terms** generated by a* $\lambda\times+$*-signature Sg, using the rules given in Table 5.2, where we assume that we are given a countably infinite stock of* ***variables**, say* $Var=\{x,y,\ldots\}$*. In the figure,* x,y *are any variables,* k *is any constant and* f *is any function symbol of non-zero arity* a*, and* σ *and* τ *are any types.*

Informally, think of the raw terms in the following ways:

• $\langle\rangle$ *can be thought of as "a unique element of a one point set".*

• $\sigma\times\tau$ *is a "cartesian product" of the types* σ *and* τ*.* $\langle-,-\rangle$ *takes a pair of arguments* M *and* N *and returns the pair* $\langle M,N\rangle$*.*

• Fst *takes a pair* P *and returns the first argument* $\mathsf{Fst}(P)$ *and similarly* Snd *takes a pair* P *and returns the second argument* $\mathsf{Snd}(P)$*.*

• $\sigma+\tau$ *is a "disjoint union" of the types* σ *and* τ*. Further,* $\mathsf{Inl}_\tau(M)$ *is an insertion of* M *(of*

$$\frac{}{\gamma}\qquad\frac{}{\mathsf{unit}}\qquad\frac{}{\mathsf{null}}\qquad\frac{\sigma\quad\tau}{\sigma\times\tau}\qquad\frac{\sigma\quad\tau}{\sigma+\tau}\qquad\frac{\sigma\quad\tau}{\sigma\Rightarrow\tau}$$

Table 5.1: Inductive Definition of Types

$$\frac{}{x} \qquad \frac{}{k} \qquad \frac{M_1 \quad \ldots \quad M_a}{f(M_1,\ldots,M_a)}$$

$$\frac{}{\langle\rangle} \qquad \frac{M \quad N}{\langle M,N\rangle} \qquad \frac{P}{\mathsf{Fst}(P)} \qquad \frac{P}{\mathsf{Snd}(P)}$$

$$\frac{M}{\mathsf{Inl}_\tau(M)} \qquad \frac{M}{\mathsf{Inr}_\sigma(M)} \qquad \frac{S \quad E \quad F}{\mathsf{Case}(S,\, x.E \mid y.F)}$$

$$\frac{M}{\lambda x:\sigma.M} \qquad \frac{F \quad A}{F\,A}$$

Table 5.2: Inductive Definition of Raw Terms

type σ) into the disjoint union $\sigma+\tau$.

• *The raw term $\mathsf{Case}(S, x.E \mid y.F)$ represents $E[M/x]$ if it is the case that the term S (of type $\sigma+\tau$) "is" a term M of type σ, and represents $F[N/y]$ if it is the case that S "is" a term N of type τ. Note that any free occurrences of the variable x in E, and y in F, are bound in $\mathsf{Case}(S, x.E \mid y.F)$.*

• *$\sigma \Rightarrow \tau$ is a "function space" of the types σ and τ. $\lambda x:\sigma.M$ is a function whose value at an argument A is M with free occurrences of the variable x in M replaced by A. Note that any free occurrences of x in M are bound in $\lambda x:\sigma.M$.*

• *$F\,A$ is the result of the application of a function F to an argument A.*

Remark 5.1.2 *Each raw term denotes an abstract syntax tree. For example, $\langle M,N\rangle$ is sugar for $\mathsf{Pair}(M,N)$ and $\lambda x:\sigma.M$ is sugar for $\mathsf{lam}(x,\sigma,M)$. We shall soon make use of simultaneous substitution of raw terms for free variables, $T[\vec{U}/\vec{v}]$. Please see page 51. Do not forget that there are variable binding operations, and that the terms N_i replace free occurrences of the v_i in T. All variable clashes must be renamed.*

Discussion 5.1.3 *A **context** is a finite list of (variable, type) pairs, usually written as $\Gamma = [x_1:\sigma_1,\ldots,x_n:\sigma_n]$, where the variables are required to be distinct. A **term-in-context** is a judgement of the form $\Gamma \vdash M:\sigma$ where Γ is a context, M is a raw term and σ a type. Given a signature Sg, the **proved terms** are those terms-in-context which are inductively generated by the rules in Table 5.3. If $\Gamma \vdash M:\sigma$ is a proved term, we shall sometimes write $Sg \rhd \Gamma \vdash M:\sigma$ to indicate this. This definition deserves comment. How exactly does it map onto our definition of inductively defined sets? Recall Section 3.3.1. The set U is the set of all terms-in-context, formally the set whose elements are all the triples of the form*

$$\frac{}{\Gamma, x:\sigma, \Gamma' \vdash x:\sigma} \qquad \frac{}{\Gamma \vdash k:\sigma}\ (k:\sigma) \qquad \frac{\Gamma \vdash M_1:\sigma_1 \quad \ldots \quad \Gamma \vdash M_a:\sigma_a}{\Gamma \vdash f(M_1,\ldots,M_a):\tau}\ (f:\sigma_1,\ldots,\sigma_a \to \tau)$$

$$\frac{}{\Gamma \vdash \langle\rangle : \mathsf{unit}} \qquad \frac{\Gamma \vdash M:\sigma \quad \Gamma \vdash N:\tau}{\Gamma \vdash \langle M,N\rangle : \sigma \times \tau} \qquad \frac{\Gamma \vdash P:\sigma \times \tau}{\Gamma \vdash \mathsf{Fst}(P):\sigma} \qquad \frac{\Gamma \vdash P:\sigma \times \tau}{\Gamma \vdash \mathsf{Snd}(P):\tau}$$

$$\frac{\Gamma \vdash S:\mathsf{null}}{\Gamma \vdash \mathsf{Emp}_\sigma(S):\sigma} \qquad \frac{\Gamma \vdash M:\sigma}{\Gamma \vdash \mathsf{Inl}_\tau(M):\sigma+\tau} \qquad \frac{\Gamma \vdash N:\tau}{\Gamma \vdash \mathsf{Inr}_\sigma(N):\sigma+\tau}$$

$$\frac{\Gamma \vdash S:\sigma+\tau \quad \Gamma, x:\sigma \vdash E:\delta \quad \Gamma, y:\tau \vdash F:\delta}{\Gamma \vdash \mathsf{Case}(S,\, x.E \mid y.F):\delta}$$

$$\frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x:\sigma.M:\sigma \Rightarrow \tau} \qquad \frac{\Gamma \vdash F:\sigma \Rightarrow \tau \quad \Gamma \vdash A:\sigma}{\Gamma \vdash FA:\tau}$$

Table 5.3: Inductive Definition of $\lambda \times +$ Proved Terms

(Γ, M, σ). *Instead of writing* $(\Gamma, M, \sigma) \in U$ *we write* $\Gamma \vdash M:\sigma$. *The set of proved terms, that is the inductively defined subset* I *of* U*, consists of those terms-in-context* $\Gamma \vdash M:\sigma$ *which have a derivation tree, that is*

$$(\Gamma, M, \sigma) \in I \iff Sg \ \rhd\ \Gamma \vdash M:\sigma$$

using the notation introduced above.

Remark 5.1.4 *It is assumed that both the hypothesis and conclusion of each of these rules are well formed. For example, in the rule which introduces a function type, it is implicit that* x *does not appear in* Γ*, because* Γ *is a well formed context in both the hypothesis and conclusion of the rule.*

Discussion 5.1.5 *Now let us begin to think about how to manipulate proved terms. Informally, it should not matter what the order of the (variable, type) pairs is in a context* Γ. *Also, it is reasonable to be able to add more variables to a context of a proved term to produce another proved term. We might say that "declaring more identifiers for the program* M *will not affect the well formedness of* M*; and the order of declaration is also unimportant." Finally, we can give rules for the substitution of raw terms. Let us make these ideas precise.*

We can derive rules for the permutation of contexts (altering the order of (variable, type) pairs), and weakening of contexts (adding (variable, type) pairs to contexts). Let π *be a permutation of the first* n *positive integers. If* $\Gamma = [x_1:\sigma_1,\ldots,x_n:\sigma_n]$ *then write* $\pi\Gamma$ *for the*

context $[x_{\pi(1)} : \sigma_{\pi(1)}, \ldots, x_{\pi(n)} : \sigma_{\pi(n)}]$. *Also write* $\Gamma \subseteq \Gamma'$ *if* Γ *is a sublist of* Γ'. *The next few results formalise some of our intuitions. The proofs are easy, but we shall spell out the proof of the first result in detail to illustrate the techniques involved. Each gives rise to an admissible rule—why?*

Lemma 5.1.6 *Whenever* $Sg \ \triangleright \ \Gamma \vdash M : \sigma$, *we have* $Sg \ \triangleright \ \pi\Gamma \vdash M : \sigma$.

Proof We use rule induction for the rules in Table 5.3. More precisely we prove

$$\forall \ Sg \ \triangleright \ \Gamma \vdash M : \sigma. \qquad \boxed{Sg \ \triangleright \ \pi\Gamma \vdash M : \sigma}$$

We give some examples of property closure.

(*Property Closure for the base rule for variables*): We have to show that $Sg \ \triangleright \ \overline{\Gamma}, x : \sigma, \overline{\Gamma'} \vdash x : \sigma$, where $\overline{\Gamma}, x : \sigma, \overline{\Gamma'}$ is any permutation of the list $\Gamma, x : \sigma, \Gamma'$. But this is the case, being an instance of the rule for introducing variables!

(*Property Closure for the inductive rule for function symbols*): The inductive hypotheses are $Sg \ \triangleright \ \pi\Gamma \vdash M_i : \sigma_i$ for each i, that is, there is a derivation for each term-in-context. But now we can just apply an instance of the rule to these derivations to deduce that $Sg \ \triangleright \ \pi\Gamma \vdash f(M_1, \ldots, M_a) : \sigma$, as required. □

Lemma 5.1.7 *If* $\Gamma \subseteq \Gamma'$ *and* $Sg \ \triangleright \ \Gamma \vdash M : \sigma$, *then* $Sg \ \triangleright \ \Gamma' \vdash M : \sigma$.

Proof Rule induction for the rules in Table 5.3. □

With this, we can prove a simple lemma which tells us how we may substitute raw terms in other raw terms.

Lemma 5.1.8 *Suppose that* $Sg \ \triangleright \ \Gamma, x : \sigma, \Gamma' \vdash N : \tau$ *and* $Sg \ \triangleright \ \Gamma \vdash M : \sigma$. *Then* $Sg \ \triangleright \ \Gamma, \Gamma' \vdash N[M/x] : \tau$.

Proof We use induction Rule induction for the rules in Table 5.3. The precise statement to be proved is

$\forall Sg \ \triangleright \ \Delta \vdash N : \tau.$

$$\boxed{\forall \Gamma, \Gamma', x, \sigma, M. \ (\Delta \equiv \Gamma, x : \sigma, \Gamma' \text{ and } Sg \ \triangleright \ \Gamma \vdash M : \sigma) \text{ implies } Sg \ \triangleright \ \Gamma, \Gamma' \vdash N[M/x] : \tau}$$

appealing to Lemma 5.1.7 in the case when N is x. Recall also Remark 5.1.4. □

Proposition 5.1.9 *Suppose that* Γ *is a context and that* M *is a raw term. If we are able to derive* $Sg \ \triangleright \ \Gamma \vdash M : \sigma$ *and* $Sg \ \triangleright \ \Gamma \vdash M : \sigma'$ *then the types* σ *and* σ' *are identical.*

Proof Use induction on the derivation of $Sg \ \triangleright \ \Gamma \vdash M : \sigma$. □

Discussion 5.1.10 *In view of Proposition 5.1.9, we might say that the rules in Table 5.3 define a monomorphic type assignment system, in the sense that for any given context Γ, if $Sg \ \triangleright \ \Gamma \vdash M : \sigma$ for some raw term M, then σ is the unique type which may appear "after the colon." We refer informally to σ as the type of M.*

Exercises 5.1.11

(1) Work through the details of the proofs of Lemmas 5.1.7 and 5.1.8, and Proposition 5.1.9. Prove that if $Sg \ \triangleright \ \Gamma \vdash M : \sigma$, then the free variables of M appear in Γ.

(2) Fix a signature. Let $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash N : \tau$ be a proved term, and $\Gamma \vdash M_i : \sigma_i$ proved terms for each i. Prove that $\Gamma \vdash N[\vec{M}/\vec{x}] : \tau$ is a proved term.

Discussion 5.1.12 *A $\lambda \times +$-**theory**, Th, is a pair (Sg, Ax) where Sg is a $\lambda \times +$-signature and Ax is a collection of equations-in-context for Sg. An **equation-in-context** is a judgement of the form $\Gamma \vdash M = M' : \sigma$ where $\Gamma \vdash M : \sigma$ and $\Gamma \vdash M' : \sigma$ are proved terms. The equations-in-context in Ax are called the **axioms** of the theory. We indicate this by writing $Ax \ \triangleright \ \Gamma \vdash M = M' : \sigma$. The **theorems** of Th consist of the judgements of the form $\Gamma \vdash M = M' : \sigma$ inductively generated[1] by the rules in Tables 5.4 and 5.5—it is a consequence of the rules that $Sg \ \triangleright \ \Gamma \vdash M : \sigma$ and $Sg \ \triangleright \ \Gamma \vdash M' : \sigma$, and hence that each theorem is indeed an equation-in-context.*

5.2 Deriving a Categorical Semantics

Discussion 5.2.1 *Suppose we model σ and τ by "objects" A and B about which we make no assumptions. Let us model $x : \sigma \vdash M : \tau$ as a "relationship" between A and B about which we make no assumptions; we write $A \xrightarrow{m} B$ for this. Now let us think about how our syntactic term language is built up. Recall that each term is a finite tree. Each such tree has a top level constructor (R say), and a finite number of immediate subtrees. For the purposes of this discussion, we shall restrict to the case when there is one subtree, and the outermost constructor does not bind variables.*[2]

We shall analyze the semantics of terms by noting that all terms are built up by substitution, in the sense that a raw term $\mathsf{R}(M)$ is precisely $\mathsf{R}(x)[M/x]$. We now think about the process of substitution in general. Suppose that we have proved terms $x : \sigma \vdash M : \tau$ and $y : \tau \vdash N : \gamma$.

[1] Note that in the first rule, $Ax \ \triangleright \ \Gamma \vdash M = M' : \sigma$ is not an hypothesis, but a "side condition". The rule is a base rule. It would be better to write all of the side conditions to the right of rules (such as "where π is a permutation") but this would consume a lot of space. We will often write side conditions above the rule line; with a modicum of common sense this should not cause problems.

[2] A proper treatment of variable binding is beyond the scope of these notes. We shall identify syntax trees up to a re-naming of bound variables.

$$\frac{Ax \ \triangleright \ \Gamma \vdash M = M' : \sigma}{\Gamma \vdash M = M' : \sigma}$$

$$\frac{Sg \ \triangleright \ \Gamma \vdash M : \sigma}{\Gamma \vdash M = M : \sigma} \qquad \frac{\Gamma \vdash M = M' : \sigma}{\Gamma \vdash M' = M : \sigma} \qquad \frac{\Gamma \vdash M = M' : \sigma \quad \Gamma \vdash M' = M'' : \sigma}{\Gamma \vdash M = M'' : \sigma}$$

$$\frac{\Gamma \vdash M = M' : \sigma}{\pi\Gamma \vdash M = M' : \sigma} \quad \text{(where } \pi \text{ is a permutation)}$$

$$\frac{\Gamma \vdash M = M' : \sigma}{\Gamma' \vdash M = M' : \sigma} \quad \text{(where } \Gamma \subseteq \Gamma' \text{)}$$

$$\frac{\Gamma, x : \sigma \vdash N = N' : \tau \quad \Gamma \vdash M = M' : \sigma}{\Gamma \vdash N[M/x] = N'[M'/x] : \tau}$$

Table 5.4: Inductive Definition of $\lambda \times +$ Theorems

We have seen that there is a proved term $x : \sigma \vdash N[M/y] : \gamma$—so how should we model this? Let us just say for the moment that whatever models this term depends on how we model $x : \sigma \vdash M : \tau$ and $y : \tau \vdash N : \gamma$. We can write this as

$$\frac{[\![x : \sigma \vdash M : \tau]\!] = A \xrightarrow{m} B \qquad [\![y : \tau \vdash N : \gamma]\!] = B \xrightarrow{n} C}{[\![x : \sigma \vdash N[M/y] : \gamma]\!] = A \xrightarrow{\Box(n,m)} C}$$

where $\Box(n,m)$ is some relationship (between A and C) depending on n and m. What about the order of substitution of terms for term variables? Let $z : \gamma \vdash L : \delta$ be a further proved term (where we tacitly assume that x, y and z are distinct variables). Note that we shall identify the semantics of the proved terms

$$x : \sigma \vdash (L[N/z])[M/y] : \delta \qquad \textit{and} \qquad x : \sigma \vdash L[N[M/y]/z] : \delta$$

because $(L[N/z])[M/y]$ and $L[N[M/y]/z]$ are equal syntax trees up to re-naming bound variables. Thus

$$\Box(\Box(l,n),m) = \Box(l,\Box(n,m))$$

Now we shall think about how proved terms with exactly one free variable are formed. We will have to model $x : \sigma \vdash x : \sigma$ as a relationship $A \xrightarrow{\star_A} A$. If we think about how the substitution of terms for variables is modelled, then we deduce that if $E \xrightarrow{e} A$ and $A \xrightarrow{m} B$ then $\Box(\star_A, e) = e$ and $\Box(m, \star_A) = m$. A proved term $x : \sigma \vdash \mathsf{R}(M) : \tau'$ is the proved term $x : \sigma \vdash \mathsf{R}(y')[M/y'] : \tau'$, and so will be modelled by the relationship $A \xrightarrow{\Box(r,m)} B'$, where B'

$$\frac{Sg \ \triangleright \ \Gamma \vdash M : \mathsf{unit}}{\Gamma \vdash M = \langle\rangle : \mathsf{unit}} \qquad \frac{Sg \ \triangleright \ \Gamma \vdash M : \sigma \quad Sg \ \triangleright \ \Gamma \vdash N : \tau}{\Gamma \vdash \mathsf{Fst}(\langle M, N\rangle) = M : \sigma} \qquad \frac{Sg \ \triangleright \ \Gamma \vdash M : \sigma \quad Sg \ \triangleright \ \Gamma \vdash N : \tau}{\Gamma \vdash \mathsf{Snd}(\langle M, N\rangle) = N : \tau}$$

$$\frac{Sg \ \triangleright \ \Gamma \vdash P : \sigma \times \tau}{\Gamma \vdash \langle \mathsf{Fst}(P), \mathsf{Snd}(P)\rangle = P : \sigma \times \tau}$$

$$\frac{Sg \ \triangleright \ \Gamma \vdash S : \mathsf{null} \quad Sg \ \triangleright \ \Gamma, x : \mathsf{null} \vdash M : \sigma}{\Gamma \vdash \mathsf{Emp}_\sigma(S) = M[S/x] : \sigma}$$

$$\frac{Sg \ \triangleright \ \Gamma \vdash M : \sigma \quad Sg \ \triangleright \ \Gamma, x : \sigma \vdash E : \delta \quad Sg \ \triangleright \ \Gamma, y : \tau \vdash F : \delta}{\Gamma \vdash \mathsf{Case}(\mathsf{Inl}_\tau(M), x.E \mid y.F) = E[M/x] : \delta}$$

$$\frac{Sg \ \triangleright \ \Gamma \vdash N : \tau \quad Sg \ \triangleright \ \Gamma, x : \sigma \vdash E : \delta \quad Sg \ \triangleright \ \Gamma, y : \tau \vdash F : \delta}{\Gamma \vdash \mathsf{Case}(\mathsf{Inr}_\sigma(N), x.E \mid y.F) = F[N/x] : \delta}$$

$$\frac{Sg \ \triangleright \ \Gamma \vdash S : \sigma + \tau \quad Sg \ \triangleright \ \Gamma, z : \sigma + \tau \vdash L : \delta}{\Gamma \vdash \mathsf{Case}(S, x.L[\mathsf{Inl}_\tau(x)/z] \mid y.L[\mathsf{Inr}_\sigma(y)/z]) = L[S/z] : \delta} \quad \text{(provided } x, y \notin fv(L)\text{)}$$

$$\frac{\Gamma \vdash S = S' : \sigma + \tau \quad \Gamma, x : \sigma \vdash E = E' : \delta \quad \Gamma, y : \tau \vdash F = F' : \delta}{\Gamma \vdash \mathsf{Case}(S, x.E \mid y.F) = \Gamma \vdash \mathsf{Case}(S', x.E' \mid y.F') : \delta}$$

$$\frac{Sg \ \triangleright \ \Gamma, x : \sigma \vdash M : \tau \quad Sg \ \triangleright \ \Gamma \vdash A : \sigma}{\Gamma \vdash (\lambda x : \sigma.M)A = M[A/x] : \tau} \qquad \frac{Sg \ \triangleright \ \Gamma \vdash F : \sigma \Rightarrow \tau}{\Gamma \vdash \lambda x : \sigma.(Fx) = F : \sigma \Rightarrow \tau} \quad \text{(provided } x \notin fv(F)\text{)}$$

$$\frac{\Gamma, x : \sigma \vdash M = M' : \tau}{\Gamma \vdash \lambda x : \sigma.M = \lambda x : \sigma.M' : \sigma \Rightarrow \tau}$$

Table 5.5: Inductive Definition of $\lambda \times +$ Theorems, Continued

models τ' and we specify that the proved term $x : \tau \vdash \mathsf{R}(x) : \tau'$ is modelled by $B \xrightarrow{r} B'$. Now we summarise our deductions, writing $n \circ m$ for $\Box(n,m)$ and id_A for $\star_A$:

- *Types are interpreted by "objects," say A, B...*
- *Proved terms are interpreted by "relationships," say $A \xrightarrow{m} B$...*
- *For each object A there is a relationship id_A.*
- *Given relationships $A \xrightarrow{m} B$ and $B \xrightarrow{n} C$, there is a relationship $A \xrightarrow{n \circ m} C$.*
- *Given relationships $E \xrightarrow{e} A$ and $A \xrightarrow{m} B$, then we have $id_A \circ e = e$ and $m \circ id_A = m$.*
- *For any $A \xrightarrow{m} B$, $B \xrightarrow{n} C$ and $C \xrightarrow{l} D$, we have $l \circ (n \circ m) = (l \circ n) \circ m$.*

Note that the above summary amounts to the specification of a category. Thus we have deduced, subject to certain primitive assumptions about how to model constructors and substitution, that we can model type theory in which exactly one free variable appears in a term, in an arbitrary category. In such a category, the substitution of raw terms for variables will be interpreted by composition of morphisms.

Following our intuitions about more than one variable appearing in a term M, we will model a proved term $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash M : \tau$ in a category with finite products as a morphism of the form $[\![\Gamma \vdash M : \tau]\!] : [\![\Gamma]\!] \to [\![\tau]\!]$ where $\Gamma \stackrel{\text{def}}{=} x_1 : \sigma_1, \ldots, x_n : \sigma_n$ and $[\![\Gamma]\!]$ stands for $[\![\sigma_1]\!] \times \ldots \times [\![\sigma_n]\!]$.

Discussion 5.2.2 *We shall give a semantics to $\lambda \times +$-theories. Some readers may know that this syntax can be modelled in bicartesian closed categories. However, we shall present a uniform analysis of the syntax and rules of $\lambda \times +$-theories to discover what, in categorical terms, is the most general interpretation. Types will be modelled by objects in a category. In a $\lambda \times +$-theory, the types are specified by giving a collection of ground types, and then constructing further types from the ground types using the type constructors $\times$, $+$ and $\Rightarrow$. The interpretation of a type $\sigma \times \tau$ will depend on the interpretations of σ and τ, and similarly for the sum and function type constructors. The proved terms will be interpreted by morphisms in a category, and the assumption that the theorems are soundly interpreted will then determine equations which hold between morphisms. In the cases of binary product and sum types, and function types, we shall see that the equations between morphisms will determine the objects which model the types up to isomorphism. Finally, recall the basic assumption that all of our syntax is interpreted in a category with (at least) finite products: products are used to model the list of types which appear in contexts.*

Let us suppose that we are given a $\lambda \times +$-theory $Th = (Sg, Ax)$ and that $\mathcal{C}$ is a (locally small) category with finite products. First we consider the types of Sg. We have to give an object $[\![\gamma]\!]$ of $\mathcal{C}$ to interpret each of the ground types γ, an object $[\![\mathsf{unit}]\!]$ to interpret unit*, and an object $[\![\mathsf{null}]\!]$ to interpret* null*. We cannot say anything more specific at the moment. There should be operations in $\mathcal{C}$ which give objects $A \Box B$, $A \oplus B$ and $A \Diamond B$ for all objects A and*

B so that we can define $[\![\sigma \times \tau]\!] \stackrel{\text{def}}{=} [\![\sigma]\!] \Box [\![\tau]\!]$, $[\![\sigma + \tau]\!] \stackrel{\text{def}}{=} [\![\sigma]\!] \oplus [\![\tau]\!]$, *and* $[\![\sigma \Rightarrow \tau]\!] \stackrel{\text{def}}{=} [\![\sigma]\!] \Diamond [\![\tau]\!]$. *Having done this, we can now choose a morphism* $[\![f]\!] : [\![\sigma_1]\!] \times \ldots \times [\![\sigma_n]\!] \to [\![\sigma]\!]$ *in* $\mathcal{C}$ *for each function symbol* $f : \sigma_1 \ldots \sigma_n \to \sigma$ *of Sg. Now recall that the interpretation of a proved term* $\Gamma \vdash M : \sigma$ *is given by a morphism* $[\![\Gamma \vdash M : \sigma]\!] : [\![\Gamma]\!] \to [\![\sigma]\!]$ *in* $\mathcal{C}$. *At the moment we do not know how to define such an interpretation, but by looking at how to soundly interpret the theorems of Th we will deduce how to do this.*

Let us think about the rules of formation of proved terms in general, assuming just one hypothesis. A typical rule looks like

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \mathsf{R}(M) : \tau} \quad (\mathsf{R})$$

where $\mathsf{R}(M)$ *is a new raw term depending on M. Now suppose that* $m \stackrel{\text{def}}{=} [\![\Gamma \vdash M : \sigma]\!]$ *which is an element of* $\mathcal{C}([\![\Gamma]\!], [\![\sigma]\!])$. *How do we model* $[\![\Gamma \vdash \mathsf{R}(M) : \tau]\!] \in \mathcal{C}([\![\Gamma]\!], [\![\tau]\!])$*? All we can say at the moment is that this latter morphism will depend on m, and we can model this idea by having a function*

$$\Phi_{[\![\sigma]\!],[\![\tau]\!],[\![\Gamma]\!]} : \mathcal{C}([\![\Gamma]\!], [\![\sigma]\!]) \longrightarrow \mathcal{C}([\![\Gamma]\!], [\![\tau]\!])$$

and setting $[\![\Gamma \vdash \mathsf{R}(M) : \tau]\!] \stackrel{\text{def}}{=} \Phi_{[\![\sigma]\!],[\![\tau]\!],[\![\Gamma]\!]}(m)$. *Suppose that* $x : \gamma \vdash M : \sigma$ *and* $y : \gamma' \vdash N : \gamma$ *are any two given proved terms. If* $m \stackrel{\text{def}}{=} [\![x : \gamma \vdash M : \sigma]\!]$ *and* $n \stackrel{\text{def}}{=} [\![y : \gamma' \vdash N : \gamma]\!]$ *then* $[\![y : \gamma' \vdash M[N/x] : \sigma]\!] = m \circ n$. *Note that there are proved terms*

$$y : \gamma' \vdash \mathsf{R}(M)[N/x] : \tau \qquad \textit{and} \qquad y : \gamma' \vdash \mathsf{R}(M[N/x]) : \tau.$$

However, both of the above raw terms should be syntactically identical up to re-naming, and therefore the categorical interpretations should be the same, that is

$$\Phi_{[\![\sigma]\!],[\![\tau]\!],[\![\gamma]\!]}(m) \circ n = \Phi_{[\![\sigma]\!],[\![\tau]\!],[\![\gamma']\!]}(m \circ n). \qquad (*)$$

In fact we can be certain that $(*)$ *will hold if we demand that for every object A and B of* $\mathcal{C}$ *there is a natural transformation*

$$\Phi_{A,B} : \mathcal{C}(-, A) \longrightarrow \mathcal{C}(-, B) : \mathcal{C}^{op} \longrightarrow \mathcal{S}et.$$

The existence of this natural transformation is not only sufficient for the soundness of R*, but if we are considering the rule as part of any theory Th it is also necessary. This is because M could be any term of the form* $f(\vec{x})$ *or k where f and k are function symbols, and hence the denotation of M may be any morphism. Let us now think about specific types and terms.*

First we deal with the type unit*, interpreted by an object* $U = [\![\mathsf{unit}]\!]$. *There must always be a morphism* $u_{[\![\Gamma]\!]} \stackrel{\text{def}}{=} [\![\Gamma \vdash \langle\rangle : \mathsf{unit}]\!] : [\![\Gamma]\!] \to U$. *Looking at the equations for the unit type, if this is to be soundly interpreted, then whenever there is a morphism* $m \stackrel{\text{def}}{=} [\![\Gamma \vdash M : \mathsf{unit}]\!]$ *in*

$\mathcal{C}$, we must have $m = u_{[\![\Gamma]\!]}$. Noting that ground types could denote any objects in $\mathcal{C}$, NSC for soundness are that for any object A of $\mathcal{C}$, there must exist a unique morphism $u_A : A \to U$, that is up to isomorphism $[\![\mathsf{unit}]\!]$ is a terminal object 1 of $\mathcal{C}$.

Recall that the rule for introducing product terms is

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \sigma}{\Gamma \vdash \langle M, N \rangle : \sigma \times \sigma}$$

In order to soundly interpret this rule we shall need a natural transformation

$$\Phi_{A,B} : \mathcal{C}(-,A) \times \mathcal{C}(-,B) \longrightarrow \mathcal{C}(-,A \square B)$$

for all objects A and B of $\mathcal{C}$. Now let $m : C \to A$ and $n : C \to B$ be morphisms of $\mathcal{C}$. Applying naturality in C at the morphism $\langle m, n \rangle : C \to A \times B$ we deduce

$$(\Phi_{A,B})_C(\pi_A \langle m, n \rangle, \pi_B \langle m, n \rangle) = (\Phi_{A,B})_{A \times B}(\pi_A, \pi_B) \circ \langle m, n \rangle,$$

that is $(\Phi_{A,B})_C(m,n) = (\Phi_{A,B})_{A \times B}(\pi_A, \pi_B) \circ \langle m, n \rangle$. Now let us define the morphism $q_{A,B} : A \times B \to A \square B$ to be $(\Phi_{A,B})_{A \times B}(\pi_A, \pi_B)$. Then we can make the definition

$$[\![\Gamma \vdash \langle M, N \rangle : A \times B]\!] \stackrel{\text{def}}{=} [\![\Gamma]\!] \xrightarrow{\langle [\![\Gamma \vdash M : \sigma]\!], [\![\Gamma \vdash N : \sigma]\!] \rangle} [\![\sigma]\!] \times [\![\tau]\!] \xrightarrow{q_{[\![\sigma]\!],[\![\tau]\!]}} [\![\sigma]\!] \square [\![\tau]\!].$$

Recall one of the rules for eliminating product types

$$\frac{\Gamma \vdash P : \sigma \times \sigma}{\Gamma \vdash \mathsf{Fst}(P) : \sigma}$$

Arguing as above, to model this rule we shall need (for each A and B) a natural transformation $\Phi_{A,B} : \mathcal{C}(-, A \square B) \longrightarrow \mathcal{C}(-,A)$. Recall the Yoneda lemma. With this, we may deduce that

$$[\mathcal{C}^{op}, \mathcal{S}et](H_{A \square B}, H_A) \cong \mathcal{C}(A \square B, A)$$

which is to say that each natural transformation $\Phi_{A,B}$ corresponds to a unique morphism $p_{A,B} : A \square B \to A$. Moreover, the Yoneda Lemma says that the components of $\Phi_{A,B}$ are given by $(\Phi_{A,B})_C = \mathcal{C}(C, p_{A,B})$. So now we can define

$$[\![\Gamma \vdash \mathsf{Fst}(P) : \sigma]\!] \stackrel{\text{def}}{=} [\![\Gamma]\!] \xrightarrow{[\![\Gamma \vdash P : \sigma \times \sigma]\!]} [\![\sigma]\!] \square [\![\tau]\!] \xrightarrow{p_{[\![\sigma]\!],[\![\tau]\!]}} [\![\sigma]\!].$$

Of course we can deduce a semantics for proved terms of the form $\Gamma \vdash \mathsf{Snd}(P) : \sigma$ in much the same way, involving a morphism $p'_{A,B} : A \square B \to B$. Our last task is to see what information

we obtain by soundly interpreting the equations-in-context for product types. These are

$$\frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \mathsf{Fst}(\langle M, N\rangle) = M : \sigma} \quad (1) \qquad \frac{\Gamma \vdash M : \sigma \quad \Gamma \vdash N : \tau}{\Gamma \vdash \mathsf{Snd}(\langle M, N\rangle) = N : \tau} \quad (2)$$

$$\frac{\Gamma \vdash P : \sigma \times \tau}{\Gamma \vdash \langle \mathsf{Fst}(P), \mathsf{Snd}(P)\rangle = P : \sigma \times \tau} \quad (3)$$

If we put $h \stackrel{\text{def}}{=} [\![\Gamma \vdash P : \sigma \times \tau]\!] : C \to A \Box B$, $m \stackrel{\text{def}}{=} [\![\Gamma \vdash M : \sigma]\!] : C \to A$ *and* $n \stackrel{\text{def}}{=} [\![\Gamma \vdash N : \tau]\!] : C \to B$, *and demand that our categorical interpretation satisfies the equations-in-context, this forces*

$$\begin{aligned} p_{A,B} \circ q_{A,B} \circ \langle m, n\rangle &= m && (1)\\ p'_{A,B} \circ q_{A,B} \circ \langle m, n\rangle &= n && (2)\\ q_{A,B} \circ \langle p_{A,B} \circ h, p'_{A,B} \circ h\rangle &= h && (3) \end{aligned}$$

At last we are done, because these equations imply that, up to isomorphism, $A \Box B$ *and* $A \times B$ *are the same. Thus we may soundly interpret binary product types by binary categorical product.*

To soundly interpret the rule

$$\frac{\Gamma \vdash S : \mathsf{null}}{\Gamma \vdash \mathsf{Emp}_\sigma(S) : \sigma}$$

we shall need a natural transformation $\Phi_A : \mathcal{C}(-, N) \longrightarrow \mathcal{C}(-, A)$, *where* $N = [\![\mathsf{null}]\!]$. *The Yoneda Lemma says that the components of* Φ_A *are given by* $(\Phi_A)_C = \mathcal{C}(C, n_A)$ *where* $n_A : N \to A$ *is a morphism, one for each A. So now we can define*

$$[\![\Gamma \vdash \mathsf{Emp}_\sigma(S) : \sigma]\!] \stackrel{\text{def}}{=} [\![\Gamma]\!] \xrightarrow{[\![\Gamma \vdash S : \mathsf{null}]\!]} N \xrightarrow{n_{[\![\sigma]\!]}} [\![\sigma]\!].$$

If we write $s \stackrel{\text{def}}{=} [\![\Gamma \vdash S : \mathsf{null}]\!] : C \to N$, *and* $m \stackrel{\text{def}}{=} [\![\Gamma, x : \mathsf{null} \vdash M : \sigma]\!] : C \times N \to A$ *then the equations for the* null *type will be soundly modelled providing that*

$$n_A \circ s = m \circ \langle id, s\rangle \qquad (\dagger)$$

holds for any such morphisms. Suppose that $t : N \to A$. *Taking s to be* id_N *and m to be* $t \circ \pi_N$, *then*

$$n_A = t \circ \pi_N \circ \langle id_N, id_N\rangle = t$$

Thus N is an initial object in the category $\mathcal{C}$. *In fact* $(\dagger)$ *forces N to be distributive, that is* $\pi_N : C \times N \to N$ *is an isomorphism for every C. It is clear that* $\pi_N \circ n_{C\times N} = n_N = id_N : N \to N$. *For the converse, take s to be* π_N *and let* $\pi_C : C \times N \to C$. *Then*

$$n_{C\times N} \circ \pi_N = \pi_{C\times N} \circ \langle id_{C\times N}, \pi_C\rangle = id_{C\times N}$$

Thus up to isomorphism $[\![\mathsf{null}]\!]$ is a distributive initial object. The details for binary sums are left as a longish exercise—see page 77.

Now we shall resume the investigation of the semantics of function types. To soundly interpret the introduction rule

$$\frac{\Gamma, x:\sigma \vdash M:\tau}{\Gamma \vdash \lambda x:\sigma.M:\sigma\Rightarrow\tau}$$

we shall need (for every object A and B) a natural transformation

$$\Phi_{A,B} : \mathcal{C}(-\times A, B) \longrightarrow \mathcal{C}(-, A\diamond B),$$

and we can then define

$$[\![\Gamma \vdash \lambda x:\sigma.M:\sigma\Rightarrow\tau]\!] \stackrel{\text{def}}{=} (\Phi_{[\![\sigma]\!],[\![\tau]\!]})_{[\![\Gamma]\!]}([\![\Gamma, x:\sigma\vdash M:\tau]\!]) : [\![\Gamma]\!] \to ([\![\sigma]\!]\diamond[\![\tau]\!]).$$

To soundly interpret the elimination rule

$$\frac{\Gamma\vdash M:\sigma\Rightarrow\tau \quad \Gamma\vdash N:\sigma}{\Gamma\vdash MN:\tau}$$

we shall need a natural transformation

$$\Psi_{A,B} : \mathcal{C}(-, A\diamond B)\times\mathcal{C}(-,A) \longrightarrow \mathcal{C}(-,B)$$

for all objects A and B of $\mathcal{C}$. Given any two morphisms $m : C\to A\diamond B$ and $n : C\to A$ and applying naturality, we have

$$(\Psi_{A,B})_C(m,n) = (\Psi_{A,B})_{(A\diamond B)\times A}(\pi,\pi')\circ\langle m,n\rangle$$

where $\pi : (A\diamond B)\times A\to A\diamond B$ and $\pi' : (A\diamond B)\times A\to A$. So if we define the morphism $ev_{A,B}\stackrel{\text{def}}{=}(\Psi_{A,B})_{(A\diamond B)\times A}(\pi,\pi')$, we can make the definition

$$[\![\Gamma\vdash MN:\tau]\!] \stackrel{\text{def}}{=}$$
$$[\![\Gamma]\!] \xrightarrow{\langle [\![\Gamma\vdash M:\sigma\Rightarrow\tau]\!], [\![\Gamma\vdash N:\sigma]\!]\rangle} ([\![\sigma]\!]\diamond[\![\tau]\!])\times[\![\sigma]\!] \xrightarrow{ev_{[\![\sigma]\!],[\![\tau]\!]}} [\![\tau]\!].$$

The equations-in-context for the function type are

$$\frac{\Gamma, x:\sigma\vdash F:\tau \quad \Gamma\vdash M:\sigma}{\Gamma\vdash(\lambda x:\sigma.F)M = F[M/x]:\tau} \quad (4) \qquad \frac{\Gamma\vdash M:\sigma\Rightarrow\tau}{\Gamma\vdash\lambda x:\sigma.(Mx) = M:\sigma\Rightarrow\tau} \quad (5)$$

If our categorical interpretation is to satisfy the equation-in-context (4), then we must have $ev_{A,B}\langle(\Phi_{A,B})_C(f),m\rangle = f\langle id,m\rangle$ for all morphisms $f : C\times A\to B$ and $m : C\to A$. Using the naturality of $\Phi_{A,B}$ we can show that this equation holds just in case

$$ev_{A,B}((\Phi_{A,B})_C(f)\times id) = f. \qquad (*)$$

Satisfaction of (5) requires that

$$(\Phi_{A,B})_C(ev_{A,B}(m \times id_A)) = m \qquad (\dagger)$$

for every morphism $m : C \to A \diamond B$. If we define a natural transformation

$$\Theta : \mathcal{C}(-, A \diamond B) \longrightarrow \mathcal{C}(- \times A, B)$$

by setting

$$\Theta_C(m) \stackrel{\text{def}}{=} ev_{A,B} \circ (m \times id)$$

the equations $(*)$ and $(\dagger)$ imply that Θ is a natural bijection. Thus, up to isomorphism in the category $\mathcal{C}$, the object $A \diamond B$ is exactly the exponential $A \Rightarrow B$ and of course the morphism $ev_{A,B} : (A \diamond B) \times A \to B$ is the evaluation morphism, and such categorical structure will soundly interpret function types.

Exercises 5.2.3 *With reference to Section 5.2:*

(1) Work through the details of the derivation of the semantics of binary product types. Be careful to understand the crucial fact that because the procedures of deriving proved terms and performing substitutions commute, the procedures of deriving a proved term can be modelled in an appropriate categorical structure by operations which are natural in their arguments.

(2) Work through the details of the derivation of the semantics of function types. In particular, prove the equations $(*)$ and $(\dagger)$ hold using naturality of $\Phi_{A,B}$.

(3) (a) Use the Yoneda lemma to deduce that to soundly interpret rules

$$\frac{\Gamma \vdash M : \sigma}{\Gamma \vdash \mathsf{Inl}_\tau(M) : \sigma + \tau} \qquad \frac{\Gamma \vdash N : \tau}{\Gamma \vdash \mathsf{Inr}_\sigma(N) : \sigma + \tau}$$

it is necessary and sufficient to give morphisms $i : A \to A + B$ and $j : B \to A + B$ of $\mathcal{C}$ for all objects A and B, where we may define

$$[\![\Gamma \vdash \mathsf{Inl}_\tau(M) : \sigma + \tau]\!] \stackrel{\text{def}}{=} i \circ [\![\Gamma \vdash M : \sigma]\!]$$
$$[\![\Gamma \vdash \mathsf{Inr}_\sigma(N) : \sigma + \tau]\!] \stackrel{\text{def}}{=} j \circ [\![\Gamma \vdash N : \tau]\!].$$

(b) By writing down an appropriate family of functions on morphism sets which will give a sound interpretation to

$$\frac{\Gamma \vdash S : \sigma + \tau \quad \Gamma, x : \sigma \vdash E : \delta \quad \Gamma, y : \tau \vdash F : \delta}{\Gamma \vdash \mathsf{Case}(S, x.E \mid y.F) : \delta}$$

and considering naturality conditions, prove that your functions may be specified in terms of a family of functions

$$\Phi_C : \mathcal{C}(C \times A, D) \times \mathcal{C}(C \times B, D) \longrightarrow \mathcal{C}(C \times (A + B), D)$$

which are natural in C. We can then define

$$[\![\Gamma \vdash \mathsf{Case}(S, x.E \mid y.F) : \delta]\!] \stackrel{\text{def}}{=} \Phi_{[\![\Gamma]\!]}([\![\Gamma, x:\sigma \vdash E:\delta]\!], [\![\Gamma, y:\tau \vdash F:\delta]\!]) \circ \langle id_{[\![\Gamma]\!]}, [\![\Gamma \vdash S : \sigma+\tau]\!]\rangle.$$

(c) Using the semantics assigned to proved terms, write down the equations which must hold between morphisms of $\mathcal{C}$ in order that the equations-in-context which appear on pages 70 and 71 are always satisfied. Deduce that the function

$$\mathcal{C}(C \times (A+B), D) \longrightarrow \mathcal{C}(C \times A, D) \times \mathcal{C}(C \times B, D)$$

given by $f \mapsto (f \circ (id_C \times i), f \circ (id_C \times j))$ is a bijection. Hence show that the object $A+B$ is indeed the binary coproduct of A and B.

(d) By considering the bijection

$$\mathcal{C}(C \times A, D) \times \mathcal{C}(C \times B, D) \;\cong\; \mathcal{C}((C \times A) + (C \times B), D),$$

use the Yoneda lemma to prove that the binary products of $\mathcal{C}$ must ***distribute*** *over binary coproducts, that is for all objects A, B and C of $\mathcal{C}$ we have*

$$C \times (A+B) \cong (C \times A) + (C \times B).$$

Thus to interpret the case syntax soundly we require a category with finite products and binary coproducts, for which binary products distribute over binary coproducts.

5.3 Categorical Semantics

Discussion 5.3.1 *We formalise the discussion of Section 5.2. Let $\mathcal{C}$ be a bicartesian closed category and let Sg be a $\lambda \times +$-signature. Then a* ***structure,*** $\mathbf{M}$*, for Sg in $\mathcal{C}$ is specified by giving:*

- *For every ground type γ of Sg an object $[\![\gamma]\!]$ of $\mathcal{C}$,*
- *for every constant function symbol $k : \sigma$, a global element $[\![k]\!]$ of $[\![\sigma]\!]$ (where $[\![\sigma]\!]$ is defined below), and*
- *for every function symbol $f : \sigma_1 \ldots \sigma_n \to \tau$ of Sg with non-zero arity, a morphism*

$$[\![f]\!] : [\![\sigma_1]\!] \times \ldots \times [\![\sigma_n]\!] \to [\![\tau]\!],$$

where we define $[\![\sigma]\!]$ for an arbitrary type σ via structural induction, setting $[\![\mathsf{unit}]\!] \stackrel{\text{def}}{=} 1$, $[\![\sigma \times \tau]\!] \stackrel{\text{def}}{=} [\![\sigma]\!] \times [\![\tau]\!]$, $[\![\mathsf{null}]\!] \stackrel{\text{def}}{=} 0$, $[\![\sigma + \tau]\!] \stackrel{\text{def}}{=} [\![\sigma]\!] + [\![\tau]\!]$, and $[\![\sigma \Rightarrow \tau]\!] \stackrel{\text{def}}{=} [\![\sigma]\!] \Rightarrow [\![\tau]\!]$ (and of course $[\![\gamma]\!]$ is given).

$$\frac{}{[\![\Gamma, x:\sigma, \Gamma' \vdash x:\sigma]\!] \stackrel{\text{def}}{=} \pi : [\![\Gamma]\!] \times [\![\sigma]\!] \times [\![\Gamma']\!] \to [\![\sigma]\!]}$$

$$\frac{}{[\![\Gamma \vdash k:\sigma]\!] \stackrel{\text{def}}{=} [\![k]\!] \circ ! : [\![\Gamma]\!] \to 1 \to [\![\sigma]\!]} \quad (k:\sigma)$$

$$\frac{[\![\Gamma \vdash M_1:\sigma_1]\!] = m_1 : [\![\Gamma]\!] \to [\![\sigma_1]\!] \quad \ldots \quad [\![\Gamma \vdash M_n:\sigma_n]\!] = m_n : [\![\Gamma]\!] \to [\![\sigma_n]\!]}{[\![\Gamma \vdash f(\vec{M}):\tau]\!] = [\![f]\!] \circ \langle m_1, \ldots, m_n \rangle : [\![\Gamma]\!] \to ([\![\sigma_1]\!] \times \ldots \times [\![\sigma_n]\!]) \to [\![\tau]\!]} \quad (f:\sigma_1, \ldots, \sigma_n \to \tau)$$

$$\frac{}{[\![\Gamma \vdash \langle\rangle : \mathsf{unit}]\!] \stackrel{\text{def}}{=} ! : [\![\Gamma]\!] \to 1} \quad \text{(where 1 is the terminal object of } \mathcal{C})$$

$$\frac{[\![\Gamma \vdash M:\sigma]\!] = m : [\![\Gamma]\!] \to [\![\sigma]\!] \quad [\![\Gamma \vdash N:\tau]\!] = n : [\![\Gamma]\!] \to [\![\tau]\!]}{[\![\Gamma \vdash \langle M, N\rangle : \sigma \times \tau]\!] = \langle m, n\rangle : [\![\Gamma]\!] \to ([\![\sigma]\!] \times [\![\tau]\!])}$$

$$\frac{[\![\Gamma \vdash P:\sigma \times \tau]\!] = p : [\![\Gamma]\!] \to ([\![\sigma]\!] \times [\![\tau]\!])}{[\![\Gamma \vdash \mathsf{Fst}(P):\sigma]\!] = \pi_1 \circ p : [\![\Gamma]\!] \to ([\![\sigma]\!] \times [\![\tau]\!]) \to [\![\sigma]\!]}$$

$$\frac{[\![\Gamma \vdash P:\sigma \times \tau]\!] = p : [\![\Gamma]\!] \to ([\![\sigma]\!] \times [\![\tau]\!])}{[\![\Gamma \vdash \mathsf{Snd}(P):\tau]\!] = \pi_2 \circ p : [\![\Gamma]\!] \to ([\![\sigma]\!] \times [\![\tau]\!]) \to [\![\tau]\!]}$$

Table 5.6: Recursive Definition of $\lambda \times +$ Categorical Semantics

Given a context $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ *we set* $[\![\Gamma]\!] \stackrel{\text{def}}{=} [\![\sigma_1]\!] \times \ldots \times [\![\sigma_n]\!]$. *Then for every proved term* $\Gamma \vdash M : \sigma$ *we shall use the structure* $\mathbf{M}$ *to specify a morphism*

$$[\![\Gamma \vdash M:\sigma]\!] : [\![\Gamma]\!] \to [\![\sigma]\!]$$

in $\mathcal{C}$. *The semantics of proved terms is specified recursively using the rules for introducing proved terms and the definition is given in Tables 5.6 and 5.7. It is easy to see that substitution of terms is modelled by categorical composition of morphisms. We have*

Lemma 5.3.2 *Let* $\Gamma' \vdash N : \tau$ *be a proved term where* $\Gamma' = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ *and let* $\Gamma \vdash M_i : \sigma_i$ *be proved terms for* $i = 1$ *to n. Then one can show that* $\Gamma \vdash N[\vec{M}/\vec{x}] : \tau$ *is a proved term and further that*

$$[\![\Gamma \vdash N[\vec{M}/\vec{x}]:\tau]\!] = [\![\Gamma' \vdash N:\tau]\!] \circ \langle [\![\Gamma \vdash M_1:\sigma_1]\!], \ldots, [\![\Gamma \vdash M_n:\sigma_n]\!] \rangle$$

where $N[\vec{M}/\vec{x}]$ *denotes simultaneous substitution.*

Proof By rule induction on the derivation of the judgement $\Gamma' \vdash N : \tau$. □

$$\frac{[\![\Gamma \vdash S : \mathsf{null}]\!] = s : [\![\Gamma]\!] \to 0}{[\![\Gamma \vdash \mathsf{Emp}_\sigma(S) : \sigma]\!] = !\circ \cong \circ\langle id_{[\![\Gamma]\!]}, s\rangle : [\![\Gamma]\!] \to [\![\Gamma]\!] \times 0 \cong 0 \to [\![\sigma]\!]}$$

$$\frac{[\![\Gamma \vdash M : \sigma]\!] = m : [\![\Gamma]\!] \to [\![\sigma]\!]}{[\![\Gamma \vdash \mathsf{Inl}_\tau(M) : \sigma + \tau]\!] = i \circ m : [\![\Gamma]\!] \to [\![\sigma]\!] \to [\![\sigma]\!] + [\![\tau]\!]}$$

$$\frac{[\![\Gamma \vdash N : \tau]\!] = n : [\![\Gamma]\!] \to [\![\tau]\!]}{[\![\Gamma \vdash \mathsf{Inr}_\sigma(N) : \sigma + \tau]\!] = j \circ n : [\![\Gamma]\!] \to [\![\tau]\!] \to [\![\sigma]\!] + [\![\tau]\!]}$$

$$\frac{\left\{\begin{array}{l} [\![\Gamma \vdash S : \sigma + \tau]\!] = s : [\![\Gamma]\!] \to [\![\sigma]\!] + [\![\tau]\!] \\ [\![\Gamma, x : \sigma \vdash E : \delta]\!] = e : [\![\Gamma]\!] \times [\![\sigma]\!] \to [\![\delta]\!] \\ [\![\Gamma, y : \sigma \vdash F : \delta]\!] = f : [\![\Gamma]\!] \times [\![\tau]\!] \to [\![\delta]\!] \end{array}\right.}{\begin{array}{l} [\![\Gamma \vdash \mathsf{Case}(S, x.E \mid y.F) : \delta]\!] = \\ \qquad [e, f]\circ \cong \circ\langle id_{[\![\Gamma]\!]}, s\rangle : [\![\Gamma]\!] \to [\![\Gamma]\!] \times ([\![\sigma]\!] + [\![\tau]\!]) \cong ([\![\Gamma]\!] \times [\![\sigma]\!]) + ([\![\Gamma]\!] \times [\![\tau]\!]) \to [\![\delta]\!] \end{array}}$$

$$\frac{[\![\Gamma, x : \sigma \vdash M : \tau]\!] = m : [\![\Gamma]\!] \times [\![\sigma]\!] \to [\![\tau]\!]}{[\![\Gamma \vdash \lambda x : \sigma.M : \sigma \Rightarrow \tau]\!] = \lambda(m) : [\![\Gamma]\!] \to [\![\sigma]\!] \Rightarrow [\![\tau]\!]}$$

$$\frac{[\![\Gamma \vdash F : \sigma \Rightarrow \tau]\!] = f : [\![\Gamma]\!] \to ([\![\sigma]\!] \Rightarrow [\![\tau]\!]) \quad [\![\Gamma \vdash A : \sigma]\!] = a : [\![\Gamma]\!] \to [\![\sigma]\!]}{[\![\Gamma \vdash F\,A : \tau]\!] \stackrel{\text{def}}{=} ev \circ \langle f, a\rangle : [\![\Gamma]\!] \to ([\![\sigma]\!] \Rightarrow [\![\tau]\!]) \times [\![\sigma]\!] \to [\![\tau]\!]}$$

Table 5.7: Recursive Definition of $\lambda \times +$ Categorical Semantics, Continued

Exercises 5.3.3

(1) Look at the details of the categorical semantics of $\lambda\times+$-theories and understand the ideas of such a model.

(2) Work through the details of the proof of Lemma 5.3.2.

5.4 Categorical Models and the Soundness Theorem

Discussion 5.4.1 *Let* $\mathbf{M}$ *be a structure for a* $\lambda\times+$*-signature in a bicartesian closed category* $\mathcal{C}$*. Given an equation-in-context* $\Gamma\vdash M=M':\sigma$ *we say that* $\mathbf{M}$ ***satisfies*** *the equation-in-context if* $[\![\Gamma\vdash M:\sigma]\!]$ *and* $[\![\Gamma\vdash M':\sigma]\!]$ *are equal morphisms in* $\mathcal{C}$*. We say that* $\mathbf{M}$ *is a* ***model*** *of a* $\lambda\times+$*-theory* $Th=(Sg,Ax)$ *if* $\mathbf{M}$ *satisfies all of the equations-in-context in Ax. We shall also speak of* $\mathbf{M}$ *satisfying axioms and theorems. We will prove a soundness theorem for* $\lambda\times+$*-theories; of course the details really all appear in the previous section.*

Theorem 5.4.2 *Let* $\mathcal{C}$ *be a bicartesian closed category, Th a* $\lambda\times+$*-theory and* $\mathbf{M}$ *a model of Th in* $\mathcal{C}$*. Then* $\mathbf{M}$ *satisfies any equation-in-context which is a theorem of Th.*

Proof We need to see that if $\Gamma\vdash M=M':\sigma$ is a theorem of *Th*, then $[\![\Gamma\vdash M:\sigma]\!]$ and $[\![\Gamma\vdash M':\sigma]\!]$ are equal morphisms in $\mathcal{C}$. This can be shown by rule induction using the rules for deriving theorems. We give one example of this. Let

$$m\stackrel{\text{def}}{=}[\![\Gamma,x:\sigma\vdash M:\tau]\!]:[\![\Gamma]\!]\times[\![\sigma]\!]\to[\![\tau]\!]$$

and $a\stackrel{\text{def}}{=}[\![\Gamma\vdash A:\sigma]\!]:[\![\Gamma]\!]\to[\![\sigma]\!]$. Then we have

(*Property Closure for the (base) rule*):

$$\frac{Sg\ \rhd\ \Gamma,x:\sigma\vdash M:\tau\quad Sg\ \rhd\ \Gamma\vdash A:\sigma}{\Gamma\vdash(\lambda x:\sigma.M)A=M[A/x]:\tau}$$

$$\begin{aligned}[\![\Gamma\vdash(\lambda x:\sigma.M)A:\tau]\!] &= ev\langle[\![\Gamma\vdash\lambda x:\sigma.M:\tau]\!],[\![\Gamma\vdash A:\sigma]\!]\rangle\\ &= ev\langle\lambda(m),a\rangle\\ &= ev(\lambda(m)\times id)\langle id,a\rangle\\ &= m\langle id,a\rangle\\ &= [\![\Gamma\vdash M[A/x]:\tau]\!]\end{aligned}$$

where the last step follows by an application of Lemma 5.3.2. □

Exercise 5.4.3 *Work through the remaining details of the proof of Theorem 5.4.2.*

5.5 Classifying Categories and the Completeness Theorem

Discussion 5.5.1 *A classifying category for a $\lambda\times+$-theory can be thought of as a bicartesian closed category which is in some sense the smallest such category in which Th can be modelled soundly. We shall see later that the classifying category arises through a formal construction using the syntax of the theory Th.*

Suppose that we are given a morphism of bicartesian closed categories $F:\mathcal{C}\to\mathcal{D}$. Let $\mathbf{M}$ be a model of Th in $\mathcal{C}$. We shall show how to define a new model, of Th in $\mathcal{D}$, denoted by $F_*\mathbf{M}$. First a lemma.

Lemma 5.5.2 *If we set $[\![\gamma]\!]_{F_*\mathbf{M}} \stackrel{\text{def}}{=} F[\![\gamma]\!]_{\mathbf{M}}$ where γ is a ground type of Th, then it follows from this that there is a canonical isomorphism $[\![\sigma]\!]_{F_*\mathbf{M}} \cong F[\![\sigma]\!]_{\mathbf{M}}$ where σ is any type of Th.*

Proof *This can be proved by rule induction over types, making use of the fact that F is a morphism of bicartesian closed categories.* □

Then a structure $F_*\mathbf{M}$ is given by $[\![\gamma]\!]_{F_*\mathbf{M}} \stackrel{\text{def}}{=} F[\![\gamma]\!]_{\mathbf{M}}$ on ground types and $[\![f]\!]_{F_*\mathbf{M}}$ is given by the composition

$$[\![\sigma_1]\!]_{F_*\mathbf{M}}\times\ldots\times[\![\sigma_n]\!]_{F_*\mathbf{M}} \cong F[\![\sigma_1]\!]_{\mathbf{M}}\times\ldots\times F[\![\sigma_n]\!]_{\mathbf{M}} \cong'$$
$$F([\![\sigma_1]\!]_{\mathbf{M}}\times\ldots\times[\![\sigma_n]\!]_{\mathbf{M}}) \stackrel{F[\![f]\!]_{\mathbf{M}}}{\longrightarrow} F[\![\tau]\!]_{\mathbf{M}} \cong [\![\tau]\!]_{F_*\mathbf{M}}$$

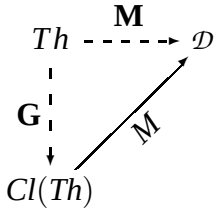
where $f:\sigma_1,\ldots,\sigma_n\to\tau$ is a function symbol of Th, the isomorphims $\cong$ exist because of Lemma 5.5.2, and $\cong'$ arises from F preserving finite products.

In fact $F_*\mathbf{M}$ is a model of Th. Given a proved term $\Gamma\vdash M:\sigma$ one can show by induction that the morphism $[\![\Gamma\vdash M:\sigma]\!]_{F_*\mathbf{M}}$ is given by the composition

$$[\![\sigma_1]\!]_{F_*\mathbf{M}}\times\ldots\times[\![\sigma_n]\!]_{F_*\mathbf{M}} \cong F([\![\sigma_1]\!]_{\mathbf{M}}\times\ldots\times[\![\sigma_n]\!]_{\mathbf{M}}) \stackrel{F[\![\Gamma\vdash M:\sigma]\!]_{\mathbf{M}}}{\longrightarrow} F[\![\sigma]\!]_{\mathbf{M}}.$$

If we are given proved terms $\Gamma\vdash M:\sigma$ and $\Gamma\vdash N:\sigma$ for which $[\![\Gamma\vdash M:\sigma]\!]_{\mathbf{M}} = [\![\Gamma\vdash N:\sigma]\!]_{\mathbf{M}}$ then certainly $[\![\Gamma\vdash M:\sigma]\!]_{F_*\mathbf{M}} = [\![\Gamma\vdash N:\sigma]\!]_{F_*\mathbf{M}}$. Thus if $\mathbf{M}$ is a model of Th in $\mathcal{C}$ then $F_*\mathbf{M}$ is a model of Th in $\mathcal{D}$.

Let Th be a $\lambda\times+$-theory. A bicartesian closed category $Cl(Th)$ is called the ***classifying*** *category of Th if there is a model $\mathbf{G}$ of Th in $Cl(Th)$ for which given any category $\mathcal{D}$ with finite products, and a model $\mathbf{M}$ of Th in $\mathcal{D}$, then there is a functor $M:Cl(Th)\to\mathcal{D}$ such that $M_*\mathbf{G}=\mathbf{M}$ and moreover, any two such functors M are naturally isomorphic.*



where $M_*\mathbf{G}=\mathbf{M}$.

Theorem 5.5.3 *Every $\lambda\times+$-theory Th has a classifying category $Cl(Th)$. We can construct a* ***canonical*** *classifying category using the syntax of Th.*

Proof The objects of $Cl(Th)$ are exactly the types of the $\lambda\times+$-signature of *Th*. The morphisms of $Cl(Th)$ are, roughly speaking, equivalence classes of raw terms with at most one free variable where the equivalence relation is given by provable equality in *Th*. More precisely, a morphism $\sigma\to\tau$ is an equivalence class $(x:\sigma\mid M)$ of pairs of the form $(x:\sigma,M)$ where x is a variable and M a raw term for which $Sg\ \rhd\ x:\sigma\vdash M:\tau$, with equivalence relation

$$(x:\sigma,M)\sim(x':\sigma,M')\qquad\text{iff}\qquad Th\ \rhd\ x:\sigma\vdash M=M'[x/x']:\tau.$$

Note that each morphism has a source object, and that the target is determined uniquely by the monomorphic type system. Composition of morphisms is given by raw term substitution and the identity on σ is given by $(x:\sigma\mid x)$.

Given two objects σ and τ, the binary product consists of the object $\sigma\times\tau$ together with suitable projections. The projection $\pi_\sigma:\sigma\times\tau\to\sigma$ is given by $(z:\sigma\times\tau\mid \mathsf{Fst}(z))$ and the projection π_τ is defined likewise from Snd. If we are given a pair of morphisms $(x:\gamma\mid M):\gamma\to\sigma$ and $(y:\gamma\mid N):\gamma\to\tau$, then the mediating morphism is given by

$$(z:\gamma\mid\langle M[z/x],N[z/y]\rangle):\gamma\to\sigma\times\tau.$$

We leave the reader to check that we **have** defined a binary product, and that $(x:\sigma\mid\langle\rangle)$ is the unique morphism $\sigma\to\mathsf{unit}$ so that unit is a terminal object for $Cl(Th)$.

We omit the details which concern binary coproducts, leaving these as an *exercise*. Note that $(z:\mathsf{null}\mid\mathsf{Emp}_\sigma(x))$ is the unique morphism $\sigma\to\mathsf{null}$. Recall that the classifier must be *distributive*. This may be verified in two ways. One either appeals to Theorem 2.8.8, after the function space structure of $Cl(Th)$ has been identified, or one can verify distributivity directly from the type theory equations. However, the latter may require some ingenuity on the part of the reader! Tedious *exercise*!

Now we move to the cartesian closed structure. The exponential of objects τ and γ is given by $\tau\Rightarrow\gamma$. Given a morphism $(z:\sigma\times\tau\mid M):\sigma\times\tau\to\gamma$ the exponential mate $\lambda(z:\sigma\times\tau\mid M):\sigma\to(\tau\Rightarrow\gamma)$ is given by

$$(x:\sigma\mid\lambda y:\tau.M[\langle x,y\rangle/z])$$

(for which we can show that $Sg\ \rhd\ x:\sigma\vdash\lambda y:\tau.M[\langle x,y\rangle/z]:\tau\Rightarrow\gamma$ is a proved term).

We define a structure $\mathbf{G}$ for *Sg* in $Cl(Th)$. First define $[\![\gamma]\!]_{\mathbf{G}}\stackrel{\text{def}}{=}\gamma$ where γ is any ground type of *Sg* (and hence it follows that $[\![\sigma]\!]_{\mathbf{G}}=\sigma$ for any type σ). If $f:\sigma_1,\ldots,\sigma_n\to\tau$ is a function symbol of *Sg* with non-zero arity n then also define

$$[\![f]\!]_{\mathbf{G}}\stackrel{\text{def}}{=}(z:\Pi_1^n\sigma_i\mid f(\mathsf{Proj}_1(z),\ldots,\mathsf{Proj}_n(z)))$$

where $\Pi_1^n\sigma_i \stackrel{\text{def}}{=} (\ldots((\sigma_1 \times \sigma_2) \times \sigma_3) \times \ldots) \times \sigma_n$ and where $\mathsf{Proj}_n(z) \stackrel{\text{def}}{=} \mathsf{Snd}(z)$, $\mathsf{Proj}_{n-1}(z) \stackrel{\text{def}}{=} \mathsf{Snd}(\mathsf{Fst}(z))$ and so on. Certainly we have

$$Sg \ \rhd \ z : \Pi_1^n\sigma_i \vdash f(\mathsf{Proj}_1(z), \ldots, \mathsf{Proj}_n(z)) : \tau.$$

Finally, if $k : \sigma$ then $[\![k]\!]_{\mathbf{G}} \stackrel{\text{def}}{=} (x : \mathsf{unit} \mid k)$.

We check that $\mathbf{G}$ is indeed a model of $Th = (Sg, Ax)$. Suppose that $Sg \ \rhd \ \Gamma \vdash M : \sigma$. Then we can prove by induction that

$$[\![\Gamma \vdash M : \sigma]\!]_{\mathbf{G}} = (z : \Pi_1^n\sigma_i \mid M[\mathsf{Proj}_1(z), \ldots, \mathsf{Proj}_n(z)/x_1, \ldots, x_n])$$

where the x_i are the variables in Γ. Now, if we have $Th \ \rhd \ \Gamma \vdash M = M' : \sigma$, one can see from the properties of substitution that

$$Th \ \rhd \ z : \Pi_1^n\sigma_i \vdash M[\mathsf{Proj}_1(z), \ldots, \mathsf{Proj}_n(z)/x_1, \ldots, x_n] = M'[\mathsf{Proj}_1(z), \ldots, \mathsf{Proj}_n(z)/x_1, \ldots, x_n] : \sigma$$

and hence that $[\![\Gamma \vdash M : \sigma]\!]_{\mathbf{G}} = [\![\Gamma \vdash M' : \sigma]\!]_{\mathbf{G}}$. So $\mathbf{G}$ is indeed a model.

Now let $\mathbf{M}$ be a model of a $\lambda\times+$-theory Th in $\mathcal{D}$. We define $M : Cl(Th) \to \mathcal{D}$ by

$$(x : \sigma \mid M) : \sigma \longrightarrow \tau \quad \longmapsto \quad [\![x : \sigma \vdash M : \tau]\!]_{\mathbf{M}} : [\![\sigma]\!]_{\mathbf{M}} \longrightarrow [\![\tau]\!]_{\mathbf{M}}$$

The soundness theorem says that the definition makes sense. It is easy to see that M is a bicartesian closed functor.

It is routine to verify that $M_*\mathbf{G} = \mathbf{M}$. The action of $M_*\mathbf{G}$ is essentially to apply the structure $[\![-]\!]_{\mathbf{M}}$. For example, consider a function symbol $f : \sigma_1, \sigma_2 \to \tau$. Then

$$\begin{aligned}
[\![f]\!]_{M_*\mathbf{G}} &= M(z : \sigma_1 \times \sigma_2 \mid f(\mathsf{Proj}_1(z), \mathsf{Proj}_2(z))) \\
&= [\![z : \sigma_1 \times \sigma_2 \vdash f(\mathsf{Proj}_1(z), \mathsf{Proj}_2(z)) : \tau]\!]_{\mathbf{M}} \\
&= [\![f]\!]_{\mathbf{M}} \circ \langle [\![z : \sigma_1 \times \sigma_2 \vdash \mathsf{Fst}(z) : \sigma_1]\!]_{\mathbf{M}}, [\![z : \sigma_1 \times \sigma_2 \vdash \mathsf{Snd}(z) : \sigma_2]\!]_{\mathbf{M}} \rangle \\
&= [\![f]\!]_{\mathbf{M}} \circ \langle \pi, \pi' \rangle \\
&= [\![f]\!]_{\mathbf{M}}.
\end{aligned}$$

Suppose that there is another bicartesian closed functor $M' : Cl(Th) \to \mathcal{D}$ for which $M'_*\mathbf{G} = \mathbf{M}$. If σ is an object of $Cl(Th)$ then

$$M\sigma \stackrel{\text{def}}{=} [\![\sigma]\!]_{\mathbf{M}} = [\![\sigma]\!]_{M'_*\mathbf{G}} \cong M'[\![\sigma]\!]_{\mathbf{G}} = M'\sigma$$

using Lemma 5.5.2, and this gives rise to a natural isomorphism $M \cong M'$. □

Exercise 5.5.4 *Prove that $M \cong M'$ is indeed a natural isomorphism.*

Discussion 5.5.5 *We have seen that the categorical semantics is sound, meaning that any theorem of a theory is satisfied by a categorical model. The converse notion to soundness is called completeness. The categorical semantics for a theory Th is said to be* ***complete*** *if an equation-in-context E of Th is a theorem whenever for all categories $\mathcal{C}$ with finite products, E is satisfied by all models $\mathbf{M}$ in $\mathcal{C}$.*

Theorem 5.5.6 *The categorical semantics of $\lambda\times+$-theories in categories with finite products is both sound and complete.*

Proof *Soundness was proved in Theorem 5.4.2. Conversely, let $\Gamma \vdash M = M' : \sigma$ be an equation-in-context of a theory Th. If it is satisfied by all models, in particular it is satisfied by the generic model of Th in $Cl(Th)$ and hence $Th \rhd \Gamma \vdash M = M' : \sigma$ as required.* □

6

Applications and Further Study

6.1 The Disjunction Property

Discussion 6.1.1 *Suppose that $Th = (Sg, \varnothing)$ is an IpL theory for which there are no (non-logical) axioms. Then if $Th \ \rhd \vdash \phi \vee \psi$ is a theorem, so too are $Th \ \rhd \vdash \phi$ and $Th \ \rhd \vdash \psi$. This is known as the **disjunction property**, DP. We shall show how this can be proved using categorical methods.*

*The **categorical disjunction property**, CDP, for a Heyting prelattice, states that if we have $\top \cong h \vee k$ then either $\top \cong h$ or $\top \cong k$. We shall prove DP by showing that $Cl(Th)$ satisfies CDP—DP for IpL is then immediate, thanks to the definition of $Cl(Th)$ (and the trivial fact that $Th \ \rhd \ \mathsf{true} \vdash \theta$ just in case $Th \ \rhd \vdash \theta$).*

Lemma 6.1.2 *Let $\Gamma : H \to K$ be a function between Heyting prelattices which preserves finite meets. Define*

$$\mathcal{GL}(\Gamma) \stackrel{\text{def}}{=} \{ \ (k,h) \in K \times H \ \mid \ k \leq \Gamma(h) \ \}$$

with the pointwise order. Then $\mathcal{GL}(\Gamma)$ is a Heyting prelattice where

- *the top element is $(\top_K, \top_H)$*
- *the bottom element is $(\bot_K, \bot_H)$*
- $(k,h) \wedge (k',h') \stackrel{\text{def}}{=} (k \wedge k', h \wedge h')$
- $(k,h) \vee (k',h') \stackrel{\text{def}}{=} (k \vee k', h \vee h')$
- $(k,h) \Rightarrow (k',h') \stackrel{\text{def}}{=} ((k \Rightarrow k') \wedge \Gamma(h \Rightarrow h'), h \Rightarrow h')$

and $\pi_2 : \mathcal{GL}(\Gamma) \to H$ is a homomorphism of Heyting prelattices.

Proof Routine calculations, left as an *exercise*. □

Proposition 6.1.3 *Set $\mathbb{B} \stackrel{\text{def}}{=} \{ \bot, \top \}$ where $\bot \leq \top$, the set of Booleans. Let $\Gamma : Cl(Th) \to \mathbb{B}$ be defined by*

$$\phi \mapsto \begin{cases} \top & \text{if } \phi \cong \mathsf{true} \\ \bot & \text{otherwise} \end{cases}$$

Then Γ is a finite meet preserving function, and hence $\mathcal{GL}(\Gamma)$ is a Heyting prelattice. Further, $\mathcal{GL}(\Gamma)$ satisfies CDP.

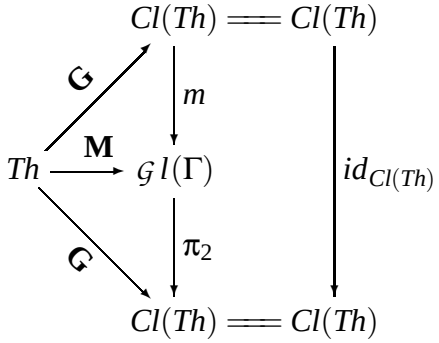
Proof Note that $Th \rhd \mathsf{true} \vdash \phi \wedge \psi$ if and only if $Th \rhd \mathsf{true} \vdash \phi$ and $Th \rhd \mathsf{true} \vdash \psi$. Hence Γ preserves binary meets, and further $\Gamma(\mathsf{true}) = \top$. Then apply Lemma 6.1.2. Now suppose that

$$(b, \phi) \vee (b', \phi') = (b \vee b', \phi \vee \phi') \cong (\top, \mathsf{true}) \in \mathcal{GL}(\Gamma).$$

Hence $b \vee b' \cong \top \in \mathbb{B}$ and hence either $b = \top$ or $b' = \top$. In the former case we must have $\top \leq \Gamma(\phi)$ hence $\top = \Gamma(\phi)$ implying $\phi \cong \mathsf{true}$. In the latter case $\phi' \cong \mathsf{true}$ follows similarly. Thus CDP holds for $\mathcal{GL}(\Gamma)$. □

Theorem 6.1.4 *The logic IpL satisfies DP.*

Proof We show CDP for $Cl(Th)$. Let us define a structure $\mathbf{M}$ for Th in $\mathcal{GL}(\Gamma)$ where Γ was defined in Proposition 6.1.3. We define $[\![p]\!]_{\mathbf{M}} \stackrel{\text{def}}{=} (\Gamma(p), [\![p]\!]_{\mathbf{G}})$ which is an element of $\mathcal{GL}(\Gamma)$ because $[\![p]\!]_{\mathbf{G}} = p$. This is trivially a model of Th because $Ax = \varnothing$. Hence there is a homomorphism m such that the upper left triangle below commutes up to equality (of structures)



as does the lower one given the definition of $\mathbf{M}$. It follows that $(\pi_2 \circ m)_* \mathbf{G} = \mathbf{G}$ and hence from the universal property of $Cl(Th)$ we have $\pi_2 \circ m \cong id_{Cl(Th)}$.

Now let $\phi \vee \psi \cong \mathsf{true}$ in $Cl(Th)$. We must have

$$m(\phi \vee \psi) \cong m(\phi) \vee m(\psi) \cong (\top, \mathsf{true})$$

in $\mathcal{GL}(\Gamma)$. By CDP for $\mathcal{GL}(\Gamma)$ (Proposition 6.1.3) we have either

(i) $m(\phi) \cong (\top, \mathsf{true})$ in which case $\pi_2(m(\phi)) = \mathsf{true} \cong id(\phi) = \phi$, or

(ii) $m(\psi) \cong (\top, \mathsf{true})$ so that $\psi \cong \mathsf{true}$ similarly.

□

6.2 A Conservative Extension of Type Theories

Discussion 6.2.1 *Before we begin this section, we need some preliminary definitions. An* ***algebraic theory*** *is defined to be a $\lambda \times +$-theory in which there are no product, sum,*

and function types, and (of course) none of the corresponding terms. More precisely, an algebraic theory $Th = (Sg, Ax)$ consists of

• *a collection of* ***types****;*

• *a collection of* ***function symbols*** *each with an arity and sorting, but note that the types in the sorting are restricted to the specified types;*

• *raw terms generated from these data, using only the rules*

$$\frac{}{x} \qquad \frac{}{k} \qquad \frac{M_1 \quad \ldots \quad M_a}{f(M_1, \ldots, M_a)}$$

• *proved terms, generated by the expected subset of rules from Table 5.3;*

• *theorems, generated by the rules in Table 5.4.*

It should be clear that many of the meta-theoretic results about $\lambda \times +$-theories hold also for algebraic theories. Further, we can give a categorical semantics to an algebraic theory in a category with finite products, in the expected way. Most of the results from Chapter 5 concerning categorical models apply to algebraic theories. However, the construction of a canonical classifying category requires modification. This is because the type theory no longer has explicit binary product types!

Theorem 6.2.2 *Every algebraic theory Th has a classifying theory $Cl(Th)$. A classifying category can be constructed from the syntax of Th and we shall refer to it as the* ***canonical*** *classifying category of the theory Th.*

Proof *We begin with some notation. Let $\sigma_1, \ldots, \sigma_n$ and τ be fixed types. Then we define $(\Gamma \mid M)$ to be an equivalence class of pairs (Γ, M) where*

• *$\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ is a context where the types appearing in Γ are the given $\sigma_1, \ldots, \sigma_n$,*

• *M is a raw term for which $Sg \rhd \Gamma \vdash M : \tau$, and*

• *the equivalence relation is defined by $(\Gamma, M) \sim (\Gamma', M')$ just in case $Th \rhd \Gamma \vdash M = M'[\vec{x}/\vec{x'}] : \tau$ where $\vec{x}$ are the variables in Γ and $\vec{x'}$ the variables in Γ'.*

The objects of $Cl(Th)$ are finite lists of types from the algebraic signature Sg of Th, for example $\vec{\sigma} \stackrel{\text{def}}{=} [\sigma_1, \ldots, \sigma_n]$. The morphisms with source $\vec{\sigma}$ and target $\vec{\tau}$, where $\tau \stackrel{\text{def}}{=} [\tau_1, \ldots, \tau_m]$ and both $\vec{\sigma}$ and $\vec{\tau}$ are non-empty lists, are given by finite lists of the form

$$[(\Gamma \mid M_1), \ldots, (\Gamma \mid M_m)] : \vec{\sigma} \to \vec{\tau}$$

where the types $\vec{\sigma}$ appear in Γ and we have $Sg \rhd \Gamma \vdash M_j : \tau_j$ for $1 \leq j \leq m$. Such a list will be written $(\Gamma \mid \vec{M})$ (take care with this abbreviation). We leave the reader to consider what happens if $\vec{\sigma}$ or $\vec{\tau}$ is empty. It is clear that $([x_1 : \sigma_1, \ldots, x_n : \sigma_n] \mid \vec{x})$ is the identity

morphism on $\vec{\sigma}$. Now consider the morphisms $(\Gamma \mid \vec{M}) : \vec{\sigma} \to \vec{\tau}$ and $(\Gamma' \mid \vec{N}) : \vec{\tau} \to \vec{\gamma}$. The composition $(\Gamma' \mid \vec{N}) \circ (\Gamma \mid \vec{M}) : \vec{\sigma} \to \vec{\gamma}$ is given by $[(\Gamma \mid N_1[\vec{M}/\vec{y}]), \ldots, (\Gamma \mid N_l[\vec{M}/\vec{y}])]$ where $\vec{y}$ are the variables in Γ'. Of course we must verify that the composition is well defined: these details are left to the reader. The terminal object of $Cl(Th)$ is the empty list $[]$ and binary products are given by list concatenation.

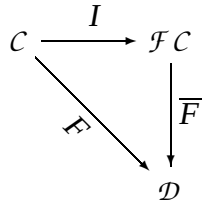
The generic model $\mathbf{G}$ of Th in $Cl(Th)$ is given by putting $[\![\sigma]\!]_{\mathbf{G}} \stackrel{\text{def}}{=} [\sigma]$ where σ is a type of Sg and $[\![f]\!]_{\mathbf{G}} \stackrel{\text{def}}{=} (\Gamma \mid f(x_1, \ldots, x_n))$ where $f : \sigma_1 \ldots \sigma_n \to \tau$ is a function symbol of Sg and $\Gamma \stackrel{\text{def}}{=} [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$.

The verification of the universal property is left as an exercise. □

Discussion 6.2.3 *We shall apply techniques of category theory and categorical semantics to prove a result about $\lambda \times +$-theories which have been generated from algebraic theories. By the $\lambda \times +$-theory generated from a given algebraic theory, we mean the $\lambda \times +$-theory which takes the types and function symbols of the algebraic signature as the ground types and function symbols of its $\lambda \times +$-signature, and the axioms of the algebraic theory as its axioms. The result we are going to prove (stated formally in Theorem 6.2.8) says that any raw term of the $\lambda \times +$-theory which has ground type is provably equal in the $\lambda \times +$-theory to a raw term of the algebraic theory.*

In order to prove this result, we shall set up a little more category-theoretic machinery. Let $\mathcal{C}$ be a category with finite products. Then a category $\mathcal{FC}$ is the ***relatively free*** *bicartesian closed category generated by $\mathcal{C}$ if there is a finite product preserving functor $I : \mathcal{C} \to \mathcal{FC}$ which satisfies the following two properties:*

(i) Suppose that $F : \mathcal{C} \to \mathcal{D}$ is a finite product preserving functor and $\mathcal{D}$ is a bicartesian closed category. Then there is a bicartesian closed functor $\overline{F} : \mathcal{FC} \to \mathcal{D}$ for which the following diagram commutes up to natural isomorphism:



We shall write $\phi : \overline{F}I \cong F$ for this.

(ii) If also $\overline{\overline{F}}$ and $\overline{\phi}$ have the same properties as $\overline{F}$ and ϕ, then there is a natural isomorphism $\psi : \overline{F} \to \overline{\overline{F}}$.

Let $Th = (Sg, Ax)$ be an algebraic theory. Let $Th' = (Sg', Ax')$ be the $\lambda \times +$-theory for which the ground types of Sg' are the types of Sg, the function symbols of Sg' are the function symbols of Sg, and $Ax' \stackrel{\text{def}}{=} Ax$. We shall now define a functor $I : Cl(Th) \to Cl(Th')$. On an object $\vec{\gamma}$ of $Cl(Th)$ set

$$I(\vec{\gamma}) \stackrel{\text{def}}{=} (\ldots(\gamma_1 \times \gamma_2) \times \ldots) \times \gamma_n$$

and given a morphism $(\Gamma \mid \vec{M})_{Th} : \vec{\gamma} \to \vec{\gamma'}$ *(where the subscript Th denotes equivalence up to provable equality in Th), then we set*

$$I(\Gamma \mid \vec{M})_{Th} \stackrel{\text{def}}{=} (z : \Pi\gamma_i \mid \langle \ldots \langle \widehat{M_1}, \widehat{M_2} \rangle, \ldots, \widehat{M_m} \rangle)_{Th'}$$

in which we have written $\Pi\gamma_i$ *for* $(\ldots(\gamma_1 \times \gamma_2) \times \ldots) \times \gamma_n$ *and also*

$$\widehat{M_j} \stackrel{\text{def}}{=} M_j[\mathsf{Proj}_1(z)/x_1, \ldots, \mathsf{Proj}_j(z)/x_j, \ldots, \mathsf{Proj}_n(z)/x_n]$$

where $\mathsf{Proj}_j(z)$ *is defined on page 84.*

Our programme for the rest of this section is as follows. First we prove Proposition 6.2.5 which shows that a certain classifying category plays the role of a relatively free bicartesian closed category. Second, we prove a purely categorical result called the "logical relations" gluing lemma, which is Lemma 6.2.6. Third, we apply the gluing lemma in the proof of Corollary 6.2.7 and then prove Theorem 6.2.8.

Remark 6.2.4 *Note that there is a well known "standard" gluing lemma (whose preorder analogue is Lemma 6.1.2) for which the glued category is defined as a comma category. The proof that exponentials exist requires a little work—cf Lemma 6.1.2. The "logical relations" construction given here simplifies the proof, also making it more uniform.*

Proposition 6.2.5 *The functor* $I : Cl(Th) \to Cl(Th')$ *presents* $Cl(Th')$ *as the relatively free bicartesian closed category generated by* $Cl(Th)$.

Proof Let $F : Cl(Th) \to \mathcal{C}$ be a functor which preserves finite products where $\mathcal{C}$ is a bicartesian closed category. We shall define a functor $\overline{F} : Cl(Th') \to \mathcal{C}$ (using the syntactic structure of the category $Cl(Th')$) through the following clauses. On objects we set

- $\overline{F}\gamma \stackrel{\text{def}}{=} F[\gamma]$ where γ is a ground type of Sg',
- $\overline{F}(\mathsf{unit}) \stackrel{\text{def}}{=} 1_{\mathcal{C}}$, the terminal object of $\mathcal{C}$,
- $\overline{F}(\mathsf{null}) \stackrel{\text{def}}{=} 0_{\mathcal{C}}$, the initial object of $\mathcal{C}$,
- $\overline{F}(\sigma \times \tau) \stackrel{\text{def}}{=} \overline{F}\sigma \times \overline{F}\tau$,
- $\overline{F}(\sigma + \tau) \stackrel{\text{def}}{=} \overline{F}\sigma + \overline{F}\tau$, and
- $\overline{F}(\sigma \Rightarrow \tau) \stackrel{\text{def}}{=} \overline{F}\sigma \Rightarrow \overline{F}\tau$.

On morphisms $(z : \delta \mid M)$ of $Cl(Th')$ we put

- $\overline{F}(z : \delta \mid \langle\rangle) \stackrel{\text{def}}{=} ! : \overline{F}\delta \to 1_{\mathcal{C}}$,
- $\overline{F}(z : \delta \mid z : \delta) \stackrel{\text{def}}{=} id_{\overline{F}\delta}$,

• $\overline{F}(z:\delta \mid f(\vec{M})) \stackrel{\text{def}}{=} F([x_1:\gamma_1,\ldots,x_n:\gamma_n] \mid f(x_1,\ldots,x_n))\circ \cong \circ$
$\langle \overline{F}(z:\delta \mid M_1),\ldots,\overline{F}(z:\delta \mid M_n)\rangle$, where the isomorphism exists because F preserves finite products,

• $\overline{F}(z:\delta \mid \mathsf{Fst}(P)) \stackrel{\text{def}}{=} \pi_1 \overline{F}(z:\delta \mid P)$ where $\pi_1 : \overline{F}\sigma \times \overline{F}\tau \to \overline{F}\sigma$,

• $\overline{F}(z:\delta \mid \mathsf{Snd}(P)) \stackrel{\text{def}}{=} \pi_2 \overline{F}(z:\delta \mid P)$ where $\pi_2 : \overline{F}\sigma \times \overline{F}\tau \to \overline{F}\tau$,

• $\overline{F}(z:\delta \mid \langle M,M'\rangle) \stackrel{\text{def}}{=} \langle \overline{F}(z:\delta \mid M),\overline{F}(z:\delta \mid M')\rangle$,

• the definition of $\overline{F}$ on morphisms involving finite coproducts is left as an *exercise*,

• $\overline{F}(z:\delta \mid MN) \stackrel{\text{def}}{=} ev\langle \overline{F}(z:\delta \mid M),\overline{F}(z:\delta \mid N)\rangle$,

• $\overline{F}(z:\delta \mid \lambda x:\sigma.M) \stackrel{\text{def}}{=} \lambda(\overline{F}(y:\delta\times\sigma \mid M[\mathsf{Fst}(y)/z]))$.

Note that $\overline{F}$ essentially preserves all of the structure of $Cl(Th')$ on the nose. It follows from this that $\overline{F}$ does indeed preserve finite products, coproducts and exponentials, and that $\overline{F}$ is a bicartesian closed functor.

Now we shall define a natural isomorphism $\phi : \overline{F}I \cong F$. It follows from the definitions that given an object $\vec{\gamma}$ of $Cl(Th)$, we have $\overline{F}I(\vec{\gamma}) = \Pi_1^n F[\gamma_i]$, and as F is finite product preserving we can set

$$\phi_{\vec{\gamma}} \stackrel{\text{def}}{=} \langle F\pi_1,\ldots,F\pi_n\rangle^{-1} : \Pi_1^n F[\gamma_i] \longrightarrow F(\vec{\gamma})$$

where $\pi_i : \vec{\gamma} \to [\gamma_i]$ in $Cl(Th)$. Of course we have to check that ϕ is a natural transformation. The details are omitted, but the reader should verify that $\phi : \overline{F}I \cong F$ is a natural isomorphism as an exercise, using structural induction.

Suppose now that $\overline{\overline{F}}$ and $\overline{\phi}$ also satisfy the roles of $\overline{F}$ and ϕ. We shall define a natural isomorphism $\psi : \overline{F} \to \overline{\overline{F}}$ through the following clauses:

• $\psi_\gamma \stackrel{\text{def}}{=} \overline{\phi}_{[\gamma]}^{-1} : \overline{F}\gamma \to \overline{\overline{F}}\gamma$, where we note that $\overline{F}\gamma = F[\gamma]$ and $\overline{\overline{F}}\gamma = \overline{\overline{F}}I[\gamma]$,

• $\psi_{\sigma\times\tau} \stackrel{\text{def}}{=} \cong \circ \psi_\sigma \times \psi_\tau : (\overline{F}\sigma \times \overline{F}\tau) \to (\overline{\overline{F}}\sigma \times \overline{\overline{F}}\tau) \to \overline{\overline{F}}(\sigma\times\tau)$,

• $\psi_{\sigma+\tau} \stackrel{\text{def}}{=} \cong \circ \psi_\sigma + \psi_\tau : (\overline{F}\sigma + \overline{F}\tau) \to (\overline{\overline{F}}\sigma + \overline{\overline{F}}\tau) \to \overline{\overline{F}}(\sigma+\tau)$, and

• $\psi_{\sigma\Rightarrow\tau} \stackrel{\text{def}}{=} \cong \circ \psi_\sigma^{-1} \Rightarrow \psi_\tau : (\overline{F}\sigma \Rightarrow \overline{F}\tau) \to (\overline{\overline{F}}\sigma \Rightarrow \overline{\overline{F}}\tau) \to \overline{\overline{F}}(\sigma\Rightarrow\tau)$,

in which the isomorphisms are the canonical ones. It is clear from the definition that the morphisms ψ_σ are always isomorphisms. It remains to see that ψ is a natural transformation. Suppose that $m : \sigma \to \tau$ is a morphism in $Cl(Th')$. Then we need to check that the diagram

$$\begin{array}{ccc} \overline{F}\sigma & \xrightarrow{\psi_\sigma} & \overline{\overline{F}}\sigma \\ {\scriptstyle \overline{F}m}\downarrow & (*) & \downarrow{\scriptstyle \overline{\overline{F}}m} \\ \overline{F}\sigma & \xrightarrow[\psi_\tau]{} & \overline{\overline{F}}\tau \end{array}$$

commutes. Any such morphism m is of the form $(x : \sigma \mid M)$; we prove that the diagram $(*)$ commutes by structural induction on the raw term M. We shall give an example of just one case, namely $m \stackrel{\text{def}}{=} (z : \delta \mid \lambda x : \sigma.M) : \delta \to \sigma \Rightarrow \tau$. So it remains to check the commutativity of the diagram

$$\begin{array}{ccc}
\overline{F}\delta & \xrightarrow{\quad\psi_\delta\quad} & \overline{\overline{F}}\delta \\
{\scriptstyle \overline{F}(z:\delta \mid \lambda x:\sigma.M)}\downarrow & & \downarrow{\scriptstyle \overline{\overline{F}}(z:\delta \mid \lambda x:\sigma.M)} \\
\overline{F}\sigma \Rightarrow \overline{F}\tau & \xrightarrow[\psi_\sigma^{-1} \Rightarrow \psi_\tau]{} & \overline{\overline{F}}\sigma \Rightarrow \overline{\overline{F}}\tau \;\cong\; \overline{\overline{F}}(\sigma \Rightarrow \tau)
\end{array}$$

Now, of course $\overline{\overline{F}}$ preserves binary products, and so our task is equivalently to see that the diagram

$$\begin{array}{ccc}
\overline{F}\delta & \xrightarrow{\quad\psi_\delta\quad} & \overline{\overline{F}}\delta \\
{\scriptstyle \overline{F}(z:\delta \mid \lambda x:\sigma.M)}\downarrow & (**) & \downarrow{\scriptstyle \lambda(\overline{\overline{F}}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z])\circ\cong)} \\
\overline{F}\sigma \Rightarrow \overline{F}\tau & \xrightarrow[\psi_\sigma^{-1} \Rightarrow \psi_\tau]{} & \overline{\overline{F}}\sigma \Rightarrow \overline{\overline{F}}\tau
\end{array}$$

commutes, where $\overline{\overline{F}}\delta \times \overline{\overline{F}}\sigma \cong \overline{\overline{F}}(\delta \times \sigma)$. Now, by induction, the following diagram commutes:

$$\begin{array}{ccc}
\overline{F}(\delta\times\sigma) & \xrightarrow{\quad\psi_{\delta\times\sigma}\quad} & \overline{\overline{F}}(\delta\times\sigma) \\
{\scriptstyle \overline{F}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z])}\downarrow & & \downarrow{\scriptstyle \overline{\overline{F}}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z])} \\
\overline{F}\tau & \xrightarrow[\psi_\tau]{} & \overline{\overline{F}}\tau
\end{array}$$

and so we have

$$\begin{aligned}
f &\stackrel{\text{def}}{=} \overline{\overline{F}}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z]) \circ \psi_{\delta\times\sigma} \circ (id_{\overline{F}\delta} \times \psi_\sigma^{-1}) \\
&= \psi_\tau \circ \overline{F}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z]) \circ (id_{\overline{F}\delta} \times \psi_\sigma^{-1}) : \overline{F}\delta \times \overline{\overline{F}}\sigma \to \overline{\overline{F}}\tau.
\end{aligned}$$

We show that of each of the paths of $(**)$ is the exponential mate of f, implying that $(**)$ commutes via the universal property of exponentials. So,

$$\begin{aligned}
& ev \circ ([(\psi_\sigma^{-1} \Rightarrow \psi_\tau) \circ \overline{F}(z:\delta \mid \lambda x:\sigma.M)] \times id_{\overline{\overline{F}}\sigma}) \\
&\quad = \psi_\tau \circ ev \circ (id_{\overline{F}\sigma \Rightarrow \overline{F}\tau} \times \psi_\sigma^{-1}) \circ (\overline{F}(z:\delta \mid \lambda x:\sigma.M) \times id_{\overline{\overline{F}}\sigma}) \\
&\quad = \psi_\tau \circ ev \circ (\lambda(\overline{F}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z])) \times id_{\overline{F}\sigma}) \circ (id_{\overline{F}\delta} \times \psi_\sigma^{-1}) \\
&\quad = \psi_\tau \circ \overline{F}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z]) \circ (id_{\overline{F}\delta} \times \psi_\sigma^{-1}) \\
&\quad = f,
\end{aligned}$$

and also

$$\begin{aligned} ev \circ ([\lambda(\overline{\overline{F}}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z]) \circ \cong) \circ \psi_\delta] \times id_{\overline{\overline{F}}\sigma}) & \\ = \ & \overline{\overline{F}}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z]) \circ \cong \circ (\psi_\delta \times id_{\overline{\overline{F}}\sigma}) \\ = \ & \overline{\overline{F}}(u:\delta\times\sigma \mid M[\mathsf{Fst}(u)/z]) \circ \cong \circ (\psi_\delta \times \psi_\sigma)(id_{\overline{F}\delta} \times \psi_\sigma^{-1}) \\ = \ & f \end{aligned}$$

as required. The other inductive cases are similar; we deduce that ψ is indeed a natural transformation. □

Lemma 6.2.6 *Let $\mathcal{D}$ be a bicartesian closed category and let $I : \mathcal{C} \to \mathcal{D}$ preserve finite products. Write*

$$\Gamma \stackrel{\text{def}}{=} (I^{op})^* \circ H : \mathcal{D} \longrightarrow [\mathcal{D}^{op}, \mathcal{S}et] \longrightarrow [\mathcal{C}^{op}, \mathcal{S}et].$$

We define a category $\mathcal{G}l(\Gamma)$ as follows:

- *The objects of $\mathcal{G}l(\Gamma)$ are triples $(F, \lhd, D)$ where $F : \mathcal{C}^{op} \to \mathcal{S}et$ is a functor, D is an object of $\mathcal{D}$, and $\lhd$ is an operation which assigns to each object C of $\mathcal{C}$ a subset, denoted by $\lhd_C$, of $FC \times \Gamma DC$.*
- *A morphism $(\alpha, d) : (F, \lhd, D) \to (F', \lhd', D')$ is given by a natural transformation $\alpha : F \to F'$ and a morphism $d : D \to D'$ in $\mathcal{D}$ for which if $x \lhd_C u$ then $\alpha_C(x) \lhd'_C d \circ u$, where of course $x \in FC$ and $u \in \Gamma DC \stackrel{\text{def}}{=} \mathcal{D}(IC, D)$.*

Then $\mathcal{G}l(\Gamma)$ is a bicartesian closed category and the obvious functor $\pi_2 : \mathcal{G}l(\Gamma) \to \mathcal{D}$ is a morphism of bicartesian closed categories.

Proof That $\mathcal{G}l(\Gamma)$ is indeed a category is routine. As a general principle, the structure of $\mathcal{G}l(\Gamma)$ is specified by a "logical relations" procedure on the subset $\lhd_C$. We sketch most of the constructions.

(*Terminal Object*): The terminal object is $(1_{[\mathcal{C}^{op}, \mathcal{S}et]}, \lhd^1, 1_{\mathcal{D}})$ where for an object C of $\mathcal{C}$ we set $* \lhd^1_C !_{IC}$. Here, $* \in 1_{[\mathcal{C}^{op}, \mathcal{S}et]}(C) = \{*\}$, and $!_{IC} : IC \to 1_{\mathcal{D}}$ is the unique map to the terminal object of $\mathcal{D}$.

(*Binary Products*): We set

$$(F, \lhd, D) \times (F', \lhd', D') \stackrel{\text{def}}{=} (F \times F', \lhd \times \lhd', D \times D')$$

where $(x, x')(\lhd \times \lhd')_C u$ just in case $x \lhd_C \pi u$ and $x' \lhd'_C \pi' u$ where of course $\pi : D \times D' \to D$ and $\pi' : D \times D' \to D'$ in $\mathcal{D}$. The projections in $\mathcal{G}l(\Gamma)$ are of course given by pairing of projections in $[\mathcal{C}^{op}, \mathcal{S}et]$ and $\mathcal{D}$, such as:

$$\pi_{(F, \lhd, D)} \stackrel{\text{def}}{=} (\pi_F, \pi_D) : (F \times F', \lhd \times \lhd', D \times D') \longrightarrow (F, \lhd, D).$$

(*Initial Object*): The initial object is $(0_{[\mathcal{C}^{op},\mathcal{S}et]}, \lhd^0, 0_{\mathcal{D}})$ where for an object C of $\mathcal{C}$ we set $\lhd^0_C \stackrel{\text{def}}{=} \varnothing$ noting that $0_{[\mathcal{C}^{op},\mathcal{S}et]}(C) = \varnothing$ and of course $\varnothing \times \mathcal{D}(IC, 0) = \varnothing$.

(*Binary Coproducts*): Write $i : A \to A + A'$ and $j : A' \to A + A'$ for coproduct insertions.

We set

$$(F, \lhd, D) + (F', \lhd', D') \stackrel{\text{def}}{=} (F + F', \lhd + \lhd', D + D')$$

where if $x \in FC$ then $i(x)(\lhd + \lhd')_C u$ just in case $u : IC \to D + D'$ factors through $i : D \to D + D'$ in $\mathcal{D}$ and if $x' \in F'C$ then $j(x')(\lhd + \lhd')_C u$ just in case u factors through $j : D' \to D + D'$ in $\mathcal{D}$. (Note that every $\xi \in FC + F'C$ is of the form $i(x)$ or $j(x')$).

(*Exponentials*): Set

$$(F, \lhd, D) \Rightarrow (F', \lhd', D') \stackrel{\text{def}}{=} (F \Rightarrow F', \lhd \Rightarrow \lhd', D \Rightarrow D')$$

where $\Phi(\lhd \Rightarrow \lhd')_C w$ just in case for all $x \lhd_C u$ we have

$$ev_C(\Phi, x) \lhd'_C ev \circ \langle w, u \rangle$$

Here, ev_C is a component of the natural transformation $ev : (F \Rightarrow F') \times F \to F'$ in the bicartesian closed category $[\mathcal{C}^{op}, \mathcal{S}et]$. The evaluation morphisms in $\mathcal{G}l(\Gamma)$ are given pointwise, as is the operation of currying. So for example if (α, d) is a morphism in $\mathcal{G}l(\Gamma)$ then we set $\lambda((\alpha, d)) \stackrel{\text{def}}{=} (\lambda(\alpha), \lambda(d))$: we check that this is a good definition. Suppose that $x \lhd_C u$ and $x' \lhd'_C u'$. Then

$$ev_C(\lambda(\alpha)_C(x), x') = \alpha_C(x, x') \lhd'_C d \circ \langle u, u' \rangle = ev \circ \langle \lambda(d) \circ u, u' \rangle$$

and so $\lambda(\alpha)_C(x)(\lhd \Rightarrow \lhd')_C \lambda(d) \circ u$ whenever $x \lhd_C u$.

□

Corollary 6.2.7 *Let $\mathcal{C}$ be a locally small category, and $\mathcal{F}\mathcal{C}$ the freely generated bicartesian closed category. Then the canonical functor $I : \mathcal{C} \to \mathcal{F}\mathcal{C}$ is full and faithful.*

Proof We appeal to Lemma 6.2.6 where $\mathcal{D}$ is taken to be $\mathcal{F}\mathcal{C}$. We define a functor $J : \mathcal{C} \to \mathcal{G}l(\Gamma)$: on objects C of $\mathcal{C}$ define JC by $(H_C, \lhd^C, IC)$ where the subset

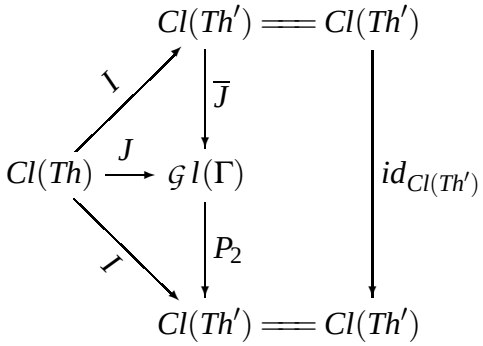
$$\lhd^C_{C'} \subseteq \mathcal{C}(C', C) \times \mathcal{D}(IC', IC)$$

is defined by just requiring $c \lhd^C_{C'} Ic$ for each morphism $c : C' \to C$ in $\mathcal{C}$. On morphisms c of $\mathcal{C}$ we set $Jc \stackrel{\text{def}}{=} (H_c, Ic)$. Note that J is faithful for the Yoneda embedding H is faithful. For fullness, let $(\alpha, d) : JC \to JC'$. Hence $\alpha : H_C \to H_{C'}$ and so $\alpha = H_c$ for some $c : C \to C'$ in $\mathcal{C}$. Now certainly $id_C \lhd^C_C id_{IC}$ and so

$$\alpha_C(id_C) = \mathcal{C}(C, c)(id_C) = c \lhd^{C'}_C d \circ id_{IC} = d$$

implying $d = Ic$; therefore $Jc = (\alpha, d)$, that is J is full.

Consider the following diagram



Using Proposition 6.2.5, the functor $\overline{J}$ exists and $\overline{J} \circ I \cong J$ naturally. By definition, $P_2 \circ J = I$. It follows that $P_2 \circ \overline{J} \circ I \cong I$ naturally, that is $(P_2 \circ \overline{J}) \circ I \cong I$, and as $id_{Cl(Th')} \circ I \cong I$ (trivially!) it follows from the universal property of relatively free bicartesian closed categories and Proposition 6.2.5 that $id_{Cl(Th')} \cong P_2 \circ \overline{J}$ naturally. This latter isomorphism implies that $\overline{J}$ is faithful. This fact, together with J full and faithful proved above, and $\overline{J} \circ I \cong J$ implies that I is full and faithful. □

Theorem 6.2.8 *Let $Th = (Sg, Ax)$ be an algebraic theory. Let $Th' = (Sg', Ax')$ be the $\lambda \times +$-theory for which the ground types of Sg' are the types of Sg, the function symbols of Sg' are the function symbols of Sg, and $Ax' \stackrel{\text{def}}{=} Ax$. Suppose that*

$$Sg' \ \rhd \ [x_1 : \gamma_1, \ldots, x_n] : \gamma_n \vdash E : \gamma$$

is a proved term generated from the $\lambda \times +$-signature Sg', where the types γ_i appearing in the context and the type γ are ground types of Sg', that is, types of Sg. Let us write $\Gamma \stackrel{\text{def}}{=} [x_1 : \gamma_1, \ldots, x_n : \gamma_n]$. Then there exists a raw term M for which

$$Sg \ \rhd \ \Gamma \vdash M : \gamma \quad \text{and} \quad Th' \ \rhd \ \Gamma \vdash E = M : \gamma.$$

Moreover, if there is another raw term M' for which $Sg \ \rhd \ \Gamma \vdash M' : \gamma$ and also $Th' \ \rhd \ \Gamma \vdash E = M' : \gamma$ then we have $Th \ \rhd \ \Gamma \vdash M = M' : \gamma$, that is M is unique up to provable equality.

Proof The fact that $I : Cl(Th) \to Cl(Th')$ is full and faithful proves the theorem: we give details for the existence part of the theorem. Suppose that $Sg' \ \rhd \ \Gamma \vdash E : \gamma$. We shall write

$$\widehat{E} \stackrel{\text{def}}{=} E[\mathsf{Proj}_1(z)/x_1, \ldots, \mathsf{Proj}_j(z)/x_j, \ldots, \mathsf{Proj}_n(z)/x_n]$$

and

$$\Pi\vec{\gamma} \stackrel{\text{def}}{=} (\ldots(\gamma_1 \times \gamma_2) \times \ldots) \times \gamma_n.$$

Then we certainly have

$$e \stackrel{\text{def}}{=} (z : \Pi\vec{\gamma} \mid \widehat{E})_{Th'} : I\vec{\gamma} \to I[\gamma]$$

in $Cl(Th')$. Using the fullness of I, there is a morphism $(\Gamma \mid M)_{Th} : \vec{\gamma} \to [\gamma]$ which is taken to e by I. But this implies that $Th' \ \rhd \ z : \Pi\vec{\gamma} \vdash \widehat{M} = \widehat{E} : \gamma$, that is

$$Th' \ \rhd \ \Gamma \vdash M = E : \gamma$$

The uniqueness part of the theorem follows from I's faithfulness in a similar fashion. □

6.3 The Curry Howard Correspondence

In this penultimate section of the notes, we explain very briefly how the material in the previous two chapters is connected. First, look at Table 4.1 and Table 5.1. You will notice that if we were to write $\times$ for $\wedge$, $+$ for $\vee$, and $\Rightarrow$ for $\to$, and then *regard any ground proposition as a ground type*, then the inductive definitions are identical. This is an example of the so-called **propositions as types** correspondence, also known as the **Curry Howard** correspondence. The logic *IpL* is a **constructive** logic—the rule of *proof by contradiction* is not present. This is the reason why the Disjunction Property holds. In fact proved terms of $\lambda\times+$-signatures can be shown to witness proofs of *IpL*-theorems. A precise version of the Curry Howard correspondence appears in the theorem below.

Theorem 6.3.1 *Let $Th = (Sg, \varnothing)$ be an IpL theory, and $Th' = (Sg', \varnothing)$ be a $\lambda\times+$-theory. Let the collection of ground propositions of Sg be the collection of ground types of Sg', where Sg' has no function symbols. Then for all propositions ϕ_i, ψ, and variables x_i, we have*

$$Th \ \rhd \ \phi_1, \ldots, \phi_n \vdash \psi \quad \text{implies} \quad \exists P.\ (Sg' \ \rhd \ x_1 : \phi_1, \ldots, x_n : \phi_n \vdash P : \psi)$$

and for all propositions ϕ_i, ψ, variables x_i, and raw terms P, we have

$$Sg' \ \rhd \ x_1 : \phi_1, \ldots, x_n : \phi_n \vdash P : \psi \quad \text{implies} \quad Th \ \rhd \ \phi_1, \ldots, \phi_n \vdash \psi$$

Exercises 6.3.2

(1) Search the literature for information about constructive logics. In particular, note that IpL can be turned into classical logic by adding the rule of proof by contradiction (reductio ad absurdum)

$$\frac{\Delta, \phi \to \mathsf{false} \vdash \mathsf{false}}{\Delta \vdash \phi}$$

where $\psi \to \mathsf{false}$ *is the definition of negation. Try to discover, or read about, alternative systems of rules for classical (and constructive) logic, and show that the same class of theorems is generated by the rules of each system.*

(2) Restricting to ground propositions, true*, and binary conjunctions, along with the type theory involving only unit and binary product expressions, try to prove each direction of the Curry Howard correspondence by using rule induction over IpL theorems, and* $\lambda \times +$ *proved terms. You will need to define mappings from theorems to proved terms, and back again. Proved terms to theorems is easy:*

$$x_1 : \phi_1, \ldots, x_n : \phi_n \vdash P : \psi \quad \mapsto \quad \phi_1, \ldots, \phi_n \vdash \psi$$

The other direction is by recursion. For example, for the rule ID *we set*

$$\delta_1, \ldots, \delta_n, \phi, \delta_{n+2}, \ldots, \delta_m \vdash \phi \quad \mapsto \quad x_1 : \delta_1, \ldots, x_n : \delta_n, x : \phi, x_{n+2} : \delta_{n+2}, \ldots, x_m : \delta_m \vdash x : \phi$$

where all variables are distinct. The reader is left to consider all other rules in Table 4.2 as exercises. The rest of the proof requires great care! For example, the logic has a rule of contraction CTRN*, and exchange* EXCH*, but in the case of* EXCH *the corresponding feature of the type theory was only proved as an admissible rule (see Lemma 5.1.6). Readers should consider whether it makes sense to adjust the definitions of rules for defining theorems and proved terms.*

(3) Search the literature for information about the Curry Howard correspondence. Do some reading, and think about the full proof of Theorem 6.3.1.

6.4 Where Now?

Rather than give a comprehensive survey of the literature, we will suggest a few references, mainly books and handbook chapters, which could form the basis of further study. Thus our references are in no way encyclopedic, and in fact exclude journal and conference research articles—for these please see the excellent bibliographies which can be found in many of the references cited below.

Category Theory

There are a number of books which cover basic category theory. For a short and gentle introduction, see [Pie91]. For a longer first text see [BW90]. Both of these books are

intended for computer scientists. The original and recommended general reference for category theory is [Mac71], which was written for mathematicians. A very concise and fast paced introduction can be found in [FS90] which also covers the theory of allegories (which, roughly, are to relations, what categories are to functions). Again for the more advanced reader, try [Tay99] which is an essential read for anyone interested in categorical logic, and which has a lot of useful background information. The Handbook of Logic in Computer Science has a wealth of material which is related to categorical logic; there is a chapter [Poi92] on category theory. Finally we mention [Wal91] which, apart from being a very interesting introduction to category theory due to the many and varied computing examples, has a short chapter devoted to distributive categories.

Predicate Logic

In these notes we have only studied first order propositional logic. Both first order and higher order predicate logic is covered explicitly, and in depth, in [Jac99]. The author presents a semantics based upon the notion of a *fibration*. There is also a good deal of material about logical (and other) quantifiers in [Tay99]. Both of these books have superb bibliographies, and any reader wishing to trace some of the history of categorical logic and related categorical machinery will find [Tay99] a fascinating read.

Polymorphic Type Theories

A very direct way of generalizing the type theories presented in these notes is to consider a notion of *type variable*. Having done this, one can then quantify over such variables to yield systems of polymorphism related to those of real programming languages. An introduction can be found in [Cro93], where the approach is quite similar to the one taken in these notes. The semantics is given in terms of *indexed categories*. For an approach using *fibrations*, and more advanced material, see [Jac99].

Topos Theory

Topos is a category which has properties similar to those of the category $\mathcal{S}et$ of sets and functions. Any topos arises as a (particular kind of) classifying category for a constructive set theory. Toposes have been the subject of intensive study over the last few decades. For an introduction which covers some of the material presented in these notes, as well as toposes, see [LS86]. One of the original texts is [Joh77]. This is intended for an advanced mathematical audience, and is perhaps not for the feint hearted. Although there are more up-to-date texts on the market, it is a mine of information for those wanting to study this subject in depth. For a more gentle general introduction to toposes, see [McL91]. For a text more biased towards mainstream mathematics, see

[MM92] or [BW85]. Also, the handbook chapter [MM96] provides a concise introductory summary.

Bibliography

[BW85] M. Barr and C. Wells. *Toposes, Triples and Theories*. Springer-Verlag, 1985.

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science*. International Series in Computer Science. Prentice Hall, 1990.

[Cro93] R. L. Crole. *Categories for Types*. Cambridge Mathematical Textbooks. Cambridge University Press, 1993. xvii+335 pages, ISBN 0521450926HB, 0521457017PB.

[FS90] P.J. Freyd and A. Scedrov. *Categories, Allegories*. Elsevier Science Publishers, 1990. Appears as Volume 39 of the North-Holland Mathematical Library.

[Jac99] Bart Jacobs. *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Elsevier, 1999.

[Joh77] P.T. Johnstone. *Topos Theory*. Academic Press, 1977.

[LS86] J. Lambek and P.J. Scott. *Introduction to Higher Order Categorical Logic*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 1986.

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, 1971.

[McL91] C. McLarty. *Elementary Categories, Elementary Toposes*, volume 21 of *Oxford Logic Guides*. Oxford University Press, 1991.

[MM92] Saunders Mac Lane and Ieke Moerdijk. *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. Universitext. Springer-Verlag, New York, 1992.

[MM96] Saunders Mac Lane and Ieke Moerdijk. Topos theory. In M. Hazewinkel, editor, *Handbook of Algebra, Vol. 1*, pages 501–528. North-Holland, Amsterdam, 1996.

[Pie91] B.C. Pierce. *Basic Category Theory for Computer Scientists*. Foundations of Computing Series. The MIT Press, 1991.

[Poi92] A. Poigné. Basic category theory. In *Handbook of Logic in Computer Science, Volume 1*. Oxford University Press, 1992.

[Tay99] Paul Taylor. *Practical Foundations of Mathematics*. Number 59 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, Cambridge, 1999.

[Wal91] R. F. C. Walters. *Categories and Computer Science*. Number 28 in Cambridge Computer Science Texts. Cambridge University Press, 1991.

Index