Categorical Logic

A chapter in the forthcoming Volume VI of
Handbook of Logic in Computer Science
S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum (eds),
Oxford University Press

Andrew M. Pitts
Cambridge University Computer Laboratory
Cambridge CB2 3QG, UK
⟨ap@cl.cam.ac.uk⟩

18 May 1995

Categorical Logic

Andrew M. Pitts

Contents

1 Introduction

This chapter provides an introduction to the interaction between category theory and mathematical logic. Category theory describes properties of mathematical structures via their transformations, or 'morphisms'. On the other hand, mathematical logic provides languages for formalizing properties of structures directly in terms of their constituent parts—elements of sets, functions between sets, relations on sets, and so on. It might seem that the kind of properties that can be described purely in terms of morphisms and their composition would be quite limited. However, beginning with the attempt of Lawvere [1964, 1966, 1969, 1970] to reformulate the foundations of mathematics using the language of category theory the development of categorical logic over the last three decades has shown that this is far from true. Indeed it turns out that many logical constructs can be characterized in terms of relatively few categorical ones, principal among which is the concept of *adjoint functor*. In this chapter we will see such categorical characterizations for, amongst other things, the notions of variable, substitution, propositional connectives and quantifiers, equality, and various type-theoretic constructs. We assume that the reader is familiar with some of the basic notions of category theory, such as functor, natural transformation, (co)limit, and adjunction: see Poigné's chapter on *Basic Category Theory* in Vol. I of this handbook, or any of the several introductions to category theory slanted towards computer science which now exist, such as [Barr and Wells, 1990], [Pierce, 1991].

Overview

There are three recurrent themes in the material we present.

Categorical semantics. Many systems of logic can only be modelled in a sufficiently complete way by going beyond the usual, set-based structures of classical model theory. Categorical logic introduces the idea of a *structure valued in a category* $\mathcal{C}$, with the classical model-theoretic notion of structure [Chang and Keisler, 1973] appearing as the special case when $\mathcal{C}$ is the category of sets and functions. For a particular logical concept, one seeks to identify what properties (or extra structure) are needed in an arbitrary category to interpret the concept in a way that respects given logical axioms and rules. A well-known example is the interpretation of simply typed lambda calculus in cartesian closed categories (see Section 3.3).

Such categorical semantics can provide a completely general and often quite simple formulation of what is required to model (a theory in) the logic. This has proved useful in cases where more traditional, set-theoretic methods of defining a notion of model either lack generality, or are inconveniently complicated to describe, or both. Seely's [1987] modelling of various impredicative type theories, such as the Girard-Reynolds polymorphic lambda calculus [Girard, 1986], [Reynolds, 1983] is an example of this:

see [Reynolds and Plotkin, 1993] and [Pitts, 1987, 1989] for instances of the use of categorical models in this case.

Internal languages. Category theory has evolved a characteristic form of proof by 'diagram-chasing' to establish properties expressible in category theoretic terms. In complex cases, such arguments can be difficult to construct and hard to follow, because of the rather limited forms of expression of purely category-theoretic language. Categorical logic enables the use of richer (and more familiar) forms of expression for establishing properties of particular kinds of category. One first defines a suitable 'internal language' naming the relevant constituents of the category and then applies a categorical semantics to turn assertions in a suitable logic over the internal language into corresponding categorical statements. Such a procedure has become most highly developed in the theory of toposes, where the internal language of a topos, coupled with the semantics of intuitionistic higher order logic in toposes, enables one to reason about the objects and morphisms of a topos 'as though they were sets and functions' (provided one reasons constructively): see [Mac Lane and Moerdijk, 1992, VI.5], [Bell, 1988], or [McLarty, 1992, Chp. 14]. This technique has been particularly helpful for working with toposes that contain 'sets' that have properties incompatible with classical logic: cases in point are the modelling of the untyped lambda calculus in terms of objects that retract onto their own function space by D. Scott [1980], and the Moggi-Hyland modelling of the Girard-Reynolds polymorphic lambda calculus by an internal full subcategory of the effective topos of Hyland [1988].

Term-model constructions. In many cases, the categorical semantics of a particular kind of logic provides the basis for a correspondence between formal theories in the logic and instances of the appropriate kind of category. In one direction, the correspondence associates to such a category the theory induced by its internal language. In the other direction one has to give a 'term-model' construction manufacturing an instance of the required categorical structure out of the syntax of the theory. An example of such a correspondence is that between theories in $\beta\eta$-equational logic over simply typed lambda calculus and cartesian closed categories. Another example is the correspondence between theories in extensional higher order intuitionistic logic and toposes. These are both examples in which a variety of category theory was discovered to correspond to a pre-existing type of logic, with interesting consequences for both sides of the equation. On the other hand, some varieties of category, embodying important mathematical constructs, have required the invention of new types of logic to establish such a theory–category correspondence. A prime example of this is the notion of category with finite limits (variously termed a *cartesian*, or a *lex* category), for which a number of different logics have been devised: the *essentially algebraic* theories of Freyd [1972], the *lim-theories* of Coste [1979, Section 2], and the *generalized algebraic* theories of Cartmell (see Section 6) can all be used for this purpose, although each has its drawbacks.

Categories arising from theories via term-model constructions can usually be characterized up to equivalence by a suitable universal property (cf. Theorem 4.2.5, for example). This has enabled meta-theoretic properties of logics to be proved via categorical algebra. For example, Freyd's proof of the existence and disjunction properties in intuitionistic logic is a case in point (see [Freyd and Scedrov, 1990, Appendix B.32]). It makes use of a categorical construction variously called *sconing*, or *glueing*. The same construction was used by Lafont to prove a strong conservativity result for the simply typed lambda calculus generated by an algebraic theory: see [Crole, 1993, Section 4.10]. A categorical construction closely related to glueing was used in [Crole and Pitts, 1992] to prove an existence property for a logic for fixed point recursion. One of the strengths of the categorical approach to logic is that the same categorical construction can sometimes be applied to obtain results for a number of different logics.

Contents of this chapter

We will not attempt a complete survey of all aspects of categorical logic that have been used, or might be used, in computer science applications. The author doubts that this can be done within the confines of a single chapter whilst avoiding superficiality. Instead we present a basic core of material in some detail.

Section 1 explains the categorical semantics of many-sorted equational logic in a category with finite products. Although the material involved is quite straightforward, we treat it carefully and in some detail since it provides the foundation for what we do in the rest of the chapter. Even at this simple level some important general features emerge—such as the role of finite products for modelling terms with several variables (independently of the presence of product types in the logic). Section 2 considers the categorical semantics of some simple type constructors. We attempt to demonstrate that the necessary categorical structure for a particular type constructor can be derived from its introduction, elimination and equality rules in a systematic way, given the basic framework of Section 2—a 'do-it-yourself categorical logic', to echo the title of [Backhouse *et al.*, 1989]. Then in Section 4 we consider a term-model construction for equational logic (with or without simple types) and the resulting theory–category correspondence.

The rest of the chapter builds on this material in two, orthogonal directions. In Section 5 we consider the categorical semantics of first order predicate logic, using a version of Lawvere's 'hyperdoctrines'. In Section 6 we present the author's distillation of several related approaches to equational logic with dependent types. Finally, Section 7 gives pointers to the literature on aspects of categorical logic not treated here.

2 Equational Logic

It has proved useful in many situations to generalize the usual notion of algebraic structure by endowing the carrier set (or sets, in the many-sorted case) with some extra mathematical structure and insisting that the operations of the algebra preserve that structure. An example from the realm of programming semantics is the use of algebras whose carriers are domains and whose operations are continuous—see [Goguen *et al.*, 1977]. Organizing the relevant mathematical structures and structure-preserving functions into a category $\mathcal{C}$, algebras with such extra structure can be viewed as a natural generalization of the usual notion of algebra: instead of using algebras valued in the category $\mathcal{S}et$ of sets and functions, one is using algebras valued in the particular category $\mathcal{C}$. To make sense of this notion for a general category, it is sufficient that it possess finite products. To motivate the interpretation of algebraic structure at this level of generality, let us see how to recast the usual, set-based semantics in a way that emphasises functions between sets rather than elements of sets.

Suppose then that we are given a many-sorted *signature Sg*, consisting of some *sort symbols*, σ, and some typed *function symbols*

$$F : \sigma_1, \ldots, \sigma_n \longrightarrow \tau \ .$$

Fixing disjoint, countably infinite sets of variables for each sort symbol, the (open) terms over Sg and their sorts are defined in the usual way: if x is a variable of sort σ, then x is a term of sort σ; and if $F : \sigma_1, \ldots, \sigma_n \longrightarrow \tau$ is a function symbol and $M_1, \ldots, M_n$ are terms of sorts $\sigma_1, \ldots, \sigma_n$ respectively, then $F(M_1, \ldots, M_n)$ is a term of sort σ. (In case $n = 0$, F is just a constant and we abbreviate $F()$ to F.) We will write $M : \sigma$ to indicate that M is a term of sort σ.

Recall that a set-valued structure for Sg is specified by giving a set $[\![\sigma]\!]$ for each sort symbol σ and a function $[\![F]\!] : [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!] \longrightarrow [\![\tau]\!]$ for each function symbol $F : \sigma_1, \ldots, \sigma_n \longrightarrow \tau$. The set $[\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!]$ is the cartesian product of the sets $[\![\sigma_i]\!]$ and so consists of n-tuples $(a_1, \ldots, a_n)$ with $a_i \in [\![\sigma_i]\!]$. In the case $n = 0$, this cartesian product contains just one element, the 0-tuple $()$, and so specifying the function $[\![F]\!]$ amounts to picking a particular element of $[\![\sigma]\!]$.

Given such a structure for Sg, the usual environment-style semantics for terms is defined by structural induction:

$$\begin{array}{rcl} [\![x]\!]\rho & \overset{\text{def}}{=} & \rho(x) \\ [\![F(M_1, \ldots, M_n)]\!]\rho & \overset{\text{def}}{=} & [\![F]\!]([\![M_1]\!]\rho, \ldots, [\![M_n]\!]\rho) \end{array}$$

where the 'environment' ρ is a finite partial function assigning elements of the structure to variables (respecting their sorts). Then an equation between two terms, $M = N$, is satisfied by the structure if for all environments defined on the variables occurring in M and N, $[\![M]\!]\rho$ and $[\![N]\!]\rho$ are equal elements of the structure

This explanation of the meaning of terms and equations is couched in terms of *elements of sets*. However, we can reformulate it in terms of *functions between sets* and using only category-theoretic properties of $\mathcal{S}et$. This reformulation will allow us to replace the category $\mathcal{S}et$ of sets and functions by other suitable categories. First note that for any fixed list of distinct variables $[x : \sigma_1, \ldots, x_n : \sigma_n]$, the set of environments defined just on this set of variables is in bijection with the cartesian product $[\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!]$. Then the mapping $\rho \mapsto [\![M]\!]\rho$ determines a function $[\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!] \longrightarrow [\![\tau]\!]$ (assuming M has sort τ) which captures the meaning of the term M in the structure. In particular, the satisfaction of an equation amounts to the equality of the corresponding functions. So we can get a semantics expressed via functions rather than elements if we give meaning not to a naked term, but rather to a 'term-in-context', that is, to a term together with a list of distinct variables containing at least those which actually occur in the term. This also allows us to drop the syntactic convention that occurrences of variables in terms be explicitly typed, by recording the sort of a variable just once, in the context. The next section summarizes this approach to the syntax.

2.1 Syntactic Considerations

Suppose given a many-sorted signature *Sg* as above and a fixed, countably infinite set of *variables*, *Var*.

Definition 2.1.1. A *context*, Γ, is a finite list $[x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ of (variable, sort)-pairs, subject to the condition that $x_1, \ldots, x_n$ are *distinct*; we will write $\mathit{var}(\Gamma)$ for the finite set $\{x_1, \ldots, x_n\}$ of variables in Γ. We will write $\Gamma, x : \sigma$ to indicate the result of extending Γ by assigning the sort σ to a variable $x \notin \mathit{var}(\Gamma)$; similarly, the result of appending two contexts whose sets of variables are disjoint will be indicated by Γ, Γ'.

A *term-in-context* takes the form

$$M : \sigma \; [\Gamma] \tag{2.1}$$

where M is a term, σ is a sort, and Γ is a context over the given signature *Sg*. The well-formed terms-in-context are inductively generated by the two rules

$$\frac{}{x : \sigma \; [\Gamma, x : \sigma, \Gamma']} \tag{2.2}$$

$$\frac{M_1 : \sigma_1 \; [\Gamma] \cdots M_n : \sigma_n \; [\Gamma]}{F(M_1, \ldots, M_n) : \tau \; [\Gamma]} \tag{2.3}$$

$$\frac{}{M = M : \sigma\ [\Gamma]} \tag{2.6}$$

$$\frac{M = M' : \sigma\ [\Gamma]}{M' = M : \sigma\ [\Gamma]} \tag{2.7}$$

$$\frac{M = M' : \sigma\ [\Gamma] \quad M' = M'' : \sigma\ [\Gamma]}{M = M'' : \sigma\ [\Gamma]} \tag{2.8}$$

$$\frac{M = M' : \sigma\ [\Gamma] \quad N = N' : \tau\ [\Gamma, x : \sigma, \Gamma']}{N[M/x] = N'[M'/x] : \tau\ [\Gamma, \Gamma']} \tag{2.9}$$

Fig. 1. Equational Logic

where in (2.3), $F : \sigma_1, \ldots, \sigma_n \longrightarrow \tau$ is any function symbol in the given signature.

A rule for substitution is derivable from rules (2.2) and (2.3):

$$\frac{M : \sigma\ [\Gamma] \quad N : \tau\ [\Gamma, x : \sigma, \Gamma']}{(N[M/x]) : \tau\ [\Gamma, \Gamma']} \tag{2.4}$$

where $N[M/x]$ denotes the result of substituting M for x in N. A rule for weakening contexts is a special case of (2.4):

$$\frac{M : \sigma\ [\Gamma]}{M : \sigma\ [\Gamma, \Gamma']} \ . \tag{2.5}$$

An *equation-in-context* takes the form $M = M' : \sigma\ [\Gamma]$ where Γ is a context, σ is a sort and M, M' are terms satisfying $M : \sigma\ [\Gamma]$ and $M' : \sigma\ [\Gamma]$. We will consider a (many-sorted) *algebraic theory*, *Th*, to consist of a signature *Sg* together with a collection of equations-in-context over *Sg*, called the *axioms* of *Th*. The *theorems* of *Th* are the equations-in-context over *Sg* that are provable from the axioms of *Th* using the rules of equational logic. These rules are shown in Fig. 1.

2.2 Categorical Semantics

Throughout this section $\mathcal{C}$ will be a fixed category with finite products. The product of a finite list $[X_1, \ldots, X_n]$ of objects in $\mathcal{C}$ will be denoted by $X_1 \times \cdots \times X_n$, with product projection morphisms

$$\pi_j : X_1 \times \cdots \times X_n \longrightarrow X_j .$$

Given morphisms $f_i : Y \longrightarrow X_i$,

$$\langle f_1, \ldots, f_n \rangle : Y \longrightarrow X_1 \times \cdots \times X_n$$

will denote the unique morphism whose compositions with each π_i is f_i. For definiteness we will assume that the product $X_1 \times \cdots \times X_n$ is defined by induction on the length of the list $[X_1, \ldots, X_n]$ using a terminal object, 1, and binary products, $- \times -$. Thus the product of the empty list is 1; and, inductively, the product of a list $[X_1, \ldots, X_n, X_{n+1}]$ of length $n+1$ is given by the binary product $(X_1 \times \cdots \times X_n) \times X_{n+1}$.

A *structure* in $\mathcal{C}$ for a given signature Sg is specified by giving an object $[\![\sigma]\!]$ in $\mathcal{C}$ for each sort σ, and a morphism $[\![F]\!] : [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!] \longrightarrow [\![\tau]\!]$ in $\mathcal{C}$ for each function symbol $F : \sigma_1, \ldots, \sigma_n \longrightarrow \tau$. (In case $n = 0$, this means that a structure assigns a global element $[\![c]\!] : 1 \longrightarrow [\![\tau]\!]$ to a constant $c : \tau$.) Given such a structure, for each context $\Gamma = x_1 : \sigma_1, \ldots, x_n : \sigma_n$, term M and sort τ for which $M : \tau\ [\Gamma]$ holds, we will define a morphism in $\mathcal{C}$:

$$[\![M : \tau[\Gamma]]\!] : [\![\Gamma]\!] \longrightarrow [\![\tau]\!]$$

where $[\![\Gamma]\!]$ denotes the product $[\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!]$. Note that the rules for deriving well-formed terms-in-context are such that for each Γ, M and τ, there is at most one way to derive $M : \tau[\Gamma]$. The definition of $[\![M : \tau[\Gamma]]\!]$ can therefore be given by induction on the structure of this derivation. Since it is also the case that τ is uniquely determined by Γ and M, we will abbreviate $[\![M : \tau[\Gamma]]\!]$ to $[\![M[\Gamma]]\!]$. The definition has two clauses, corresponding to rules (2.2) and (2.3):

$$\begin{aligned} [\![x_i[x_1 : \sigma_1, \ldots, x_n : \sigma_n]]\!] &\overset{\text{def}}{=} \pi_i \\ [\![F(M_1, \ldots, M_n)[\Gamma]]\!] &\overset{\text{def}}{=} [\![F]\!] \circ \langle [\![M_1[\Gamma]]\!], \ldots [\![M_n[\Gamma]]\!] \rangle \end{aligned}$$

Lemma 2.2.1 (Semantics of substitution). *In the categorical semantics of terms-in-context, substitution of a term for a variable in a term is interpreted via composition in the category. Specifically, if* $M_i : \sigma_i\ [\Gamma']$ *for* $i = 1, \ldots, n$, *and if* $N : \tau\ [\Gamma]$ *with* $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$, *then*

$$[\![(N[\vec{M}/\vec{x}])[\Gamma']]\!] = [\![N[\Gamma]]\!] \circ \langle [\![M_1[\Gamma']]\!], \ldots, [\![M_n[\Gamma']]\!] \rangle$$

where $N[\vec{M}/\vec{x}]$ *denotes the result of simultaneously substituting* M_i *for* x_i *(*$i = 1, \ldots, n$*) in* N*. (Note that* $(N[\vec{M}/\vec{x}]) : \tau\ [\Gamma']$ *holds by repeated use of (2.4).)*

The lemma is proved by induction on the structure of N. By taking the terms M_i to be suitable variables one obtains the following result as a special case.

Corollary 2.2.2 (Semantics of weakening). *Suppose that* $N : \tau\ [\Gamma]$ *and that* Γ' *is another context that contains* $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ *as a sub-list. Then* $[\![N[\Gamma']]\!] = [\![N[\Gamma]]\!] \circ \pi$, *where* $\pi = \langle \pi_{m(1)}, \ldots, \pi_{m(n)} \rangle$ *with* $m(i)$ *the position in* Γ' *of the variable* x_i.

Suppose $M = M' : \sigma\ [\Gamma]$ is an equation-in-context over a given signature, *Sg*. Since $M : \sigma\ [\Gamma]$ and $M : \sigma\ [\Gamma]$ are required to hold, a structure in $\mathcal{C}$ for *Sg* gives rise via the above definition, to morphisms $[\![M[\Gamma]]\!], [\![M'[\Gamma]]\!] : [\![\Gamma]\!] \longrightarrow [\![\sigma]\!]$. The structure is said to *satisfy* the equation-in-context if these morphisms are equal. If *Th* is an algebraic theory over *Sg*, then the structure is a *Th-algebra in* $\mathcal{C}$ if it satisfies all the axioms of *Th*. There are very many different categories with finite products, and algebras for an algebraic theory in one may have very different detailed structure from algebras for the same theory in another category. Nevertheless, the following proposition shows that whatever the underlying category, we can still use the familiar kind of equational reasoning embodied by the rules in Fig. 1 whilst preserving satisfaction of equations.

Theorem 2.2.3 (Soundness). *Let $\mathcal{C}$ be a category with finite products and Th an algebraic theory. Then a Th-algebra in $\mathcal{C}$ satisfies any equation-in-context which is a theorem of Th.*

Proof. The properties of equality of morphisms in $\mathcal{C}$ imply that the collection of equations-in-context satisfied by the *Th*-algebra is closed under rules (2.6), (2.7) and (2.8) in Fig. 1. Closure under rule (2.9) is a consequence of Lemma 2.2.1. ∎

The converse of this theorem, namely the completeness of the categorical semantics for equational logic, will be a consequence of the material in Section 4.2.

Summary. We conclude this section by summarizing the important features of the categorical semantics of terms and equations in a category with finite products.

- Sorts are interpreted as objects.
- A term is only interpreted in a context (an assignment of sorts to finitely many variables) containing at least the variables mentioned in the term, and such a term-in-context is interpreted as a morphism with:
  - the codomain of the morphism determined by the sort of the term;
  - the domain of the morphism determined by the context;
  - variables interpreted as product projection morphisms (identity morphisms being a special case of these);
  - substitution of terms for variables interpreted via composition and pairing;
  - weakening of contexts interpreted via composition with a product projection morphism.
- An equation is only considered in a context (containing at least the variables mentioned), and such an equation-in-context is satisfied if

the two morphisms interpreting the equated terms-in-context are actually equal in the category.

2.3 Internal Languages

The interpretation of equational logic described in the the previous section provides a means of describing certain properties of a category $\mathcal{C}$ with finite products *as though its objects were sets of elements and its morphisms functions between those sets.* To do this we use the terms over a many-sorted signature $Sg_{\mathcal{C}}$ naming the objects and morphisms of $\mathcal{C}$. The sorts of $Sg_{\mathcal{C}}$ are the objects of $\mathcal{C}$; and for each non-empty list $X_1, \dots, X_n, Y$ of objects, the function symbols of type $X_1, \dots, X_n \longrightarrow Y$ are the morphisms $f : X_1 \times \cdots \times X_n \longrightarrow Y$ in $\mathcal{C}$. Of course the size of this signature depends on the size of $\mathcal{C}$: it will have a set of symbols rather than a proper class, only if $\mathcal{C}$ is small (i.e. has only a set of morphisms). For the sake of clarity, we are not making a notational distinction between a symbol in $Sg_{\mathcal{C}}$ and the element of $\mathcal{C}$ that it names; as a result a morphism $f : X_1 \times \cdots \times X_n \longrightarrow Y$ occurs in $Sg_{\mathcal{C}}$ both as an n-ary function symbol of type $X_1, \dots, X_n \longrightarrow Y$ and as a unary one of type $X_1 \times \cdots \times X_n \longrightarrow Y$.

The terms-in-context over this signature constitute a language for describing morphisms of $\mathcal{C}$—a so-called *internal language* for $\mathcal{C}$. The connection between these terms-in-context and the morphisms they describe is given by means of the evident structure for $Sg_{\mathcal{C}}$ in $\mathcal{C}$ which maps symbols in $Sg_{\mathcal{C}}$ to the corresponding objects and morphisms in $\mathcal{C}$. Given $M : Y\ [x_1 : X_1, \dots, x_n : X_n]$, the definitions of Section 2.2 applied to this structure yield a morphism

$$[\![M[x_1 : X_1, \dots, x_n : X_n]]\!] : X_1 \times \cdots \times X_n \longrightarrow Y$$

in $\mathcal{C}$.

We will not refer to this structure for $Sg_{\mathcal{C}}$ in $\mathcal{C}$ by name, but simply say that '$\mathcal{C}$ satisfies $M = N : X\ [\Gamma]$' if the structure satisfies this equation-in-context. The following results are easy exercises in the use of the categorical semantics.

Proposition 2.3.1.

(i) *Two parallel morphisms in* $\mathcal{C}$, $f, g : X \longrightarrow X$, *are equal if and only if* $\mathcal{C}$ *satisfies* $f(x) = g(x) : X\ [x : X]$.

(ii) *A morphism* $f : X \longrightarrow X$ *in* $\mathcal{C}$ *is the identity on* X *if and only if* $\mathcal{C}$ *satisfies* $f(x) = x : X\ [x : X]$.

(iii) *A morphism* $f : X \longrightarrow Z$ *is the composition of* $g : X \longrightarrow Y$ *and* $h : Y \longrightarrow Z$ *if and only if* $\mathcal{C}$ *satisfies* $f(x) = h(g(x)) : Z\ [x : X]$.

(iv) *An object* T *is a terminal object in* $\mathcal{C}$ *if and only if there is some morphism* $t : 1 \longrightarrow T$ *satisfying* $x = t : T\ [x : T]$.

(v) $X \xleftarrow{p} Z \xrightarrow{q} Y$ *is a binary product diagram in* $\mathcal{C}$ *if and only if there*

is some morphism $r : X \times Y \longrightarrow Z$ *satisfying*

$$\begin{array}{rcl} p(r(x,y)) & = & x : X\ [x : X, y : Y] \\ q(r(x,y)) & = & y : Y\ [x : X, y : Y] \\ r(p(z), q(z)) & = & z : Z\ [z : Z]. \end{array}$$

The internal language of $\mathcal{C}$ makes it look like a category of sets and functions: given an object X, its 'elements' in the language are the terms M of sort X; and a morphism $f : X \longrightarrow Y$ yields a function sending 'elements' M of sort X to 'elements' $f(M)$ of sort Y. However, in the internal language the 'elements' of X depend upon the context; and these elements are interpreted as morphisms to X in $\mathcal{C}$ whose domain is the interpretation of the context. Thus we can think of morphisms $x : I \longrightarrow X$ in $\mathcal{C}$ as *generalized elements of* X *at stage* I and will write $x \in_I X$ to indicate this.

A particular case is when I is the terminal object *1*: the generalized elements of an object at stage *1* are more normally called its *global elements*. In general, it is not sufficient to restrict attention to global elements in order to describe properties of $\mathcal{C}$. For example, each morphism $f : X \longrightarrow Y$ yields via composition a function taking generalized elements of X at each stage I, $x \in_I X$, to generalized elements of Y at the same stage, $(f \circ x) \in_I Y$. Two such morphisms $f, g : X \longrightarrow Y$ are equal just in case they determine equal functions on generalized elements *at all stages*, but are not necessarily equal if they determine the same functions just on global elements. Call a category with terminal object *well-pointed* if for all $f, g : X \longrightarrow Y$ in $\mathcal{C}$, $f = g$ whenever $f \circ x = g \circ x$ for all global elements $x : 1 \longrightarrow X$. Typically, categories of sets equipped with extra structure and structure-preserving functions are well-pointed, and algebraic structures in such categories are closely related to corresponding set-valued structures via the use of global elements. If we only consider algebras in such well-pointed categories, we lose the ability to model certain phenomena. Here is a little example stolen from [Goguen and Meseguer, 1985], to do with 'empty sorts'.

Example 2.3.2. Consider the usual algebraic theory of boolean algebras (with sort *bool* and operations $T, F, \cong, \vee, \neg$) augmented with a second sort *hegel* and a function symbol $C : \mathit{hegel} \longrightarrow \mathit{bool}$ satisfying the axiom $\neg(C(x)) = C(x) : \mathit{bool}\ [x : \mathit{hegel}]$. Then $T = F : \mathit{bool}\ [x : \mathit{hegel}]$ is a theorem of this theory, but $T = F : \mathit{bool}\ []$ is not. Any set-valued model of the theory that assigns a non-empty set to *hegel* must identify T and F. However, there are models in non-well-pointed categories for which *hegel* is non-empty (in the sense of not being an initial object).

For example, consider the category $[\omega, \mathcal{S}et]$ of set-valued functors and natural transformations from the ordinal ω (which is a poset, hence a category). A more concrete description of $[\omega, \mathcal{S}et]$ is as the category of 'sets evolving through (discrete) time'. The objects are sets X equipped with a function $E : X \longrightarrow \omega$ recording that $x \in X$ exists at time $E(x) \in \omega$, together with a function $(-)^{+} : X \longrightarrow X$ describing how elements evolve from one instant to the next, and hence which is required to satisfy $E(x^{+}) = E(x)+1$. A morphism between two such objects, $f : X \longrightarrow Y$, is a function between the sets which preserves existence ($E(f(x)) = E(x)$) and evolution ($f(x^{+}) = f(x)^{+}$). The global sections of X are in bijection with the elements which exist at the beginning of time, $\{x \in X \mid E(x) = 0\}$. So this category is easily seen to not be well-pointed. One can model the algebraic theory of the previous paragraph in this category by interpreting *bool* as the set having two distinct elements at time 0 that evolve to a unique element at time 1, and interpreting *hegel* as the set with no elements at time 0 and a single element thereafter. (This interpretation for *hegel* is different from the initial object of $[\omega, \mathcal{S}et]$, which has no elements at any time.)

The internal language of a category with just finite products is of rather limited usefulness, because of the very restricted forms of expression and judgement in equational logic. However, internal languages over richer logics can be used as powerful tools for proving complicated 'arrow-theoretic' results by more familiar looking 'element-theoretic' arguments. See [Bell, 1988], for extended examples of this technique in the context of topos theory and higher order predicate logic. The use of categorical semantics in non-well-pointed categories gives us an increase in ability to construct useful models compared with the more traditional 'sets-and-elements' approach. The work of Reynolds and Oles [1985] on the semantics of block structure using functor categories is an example. (As the example above shows, functor categories are not well-pointed in general.) See [Mitchell and Scott, 1989] for a comparison of the categorical and set-theoretic approaches in the case of the simply typed lambda calculus.

3 Categorical Datatypes

This section explores the categorical treatment of various type constructors. We postpone considering types that depend upon variables until section 6, and concentrate here upon 'simple' types with no such dependencies. We will consider properties of these types expressible within the realm of equational logic; so from the work in the previous section we know that at the categorical level we must start with a category with finite products and ask what more it should satisfy in order to model a particular type constructor. We hope to demonstrate that the necessary categorical structure for a particular type constructor can be derived from its introduction, elimination and equality rules in a systematic way, given the basic framework of the previous section.

In fact what emerges is an interplay between type theory and categorical structure. To model the term-forming operations associated with a type constructor one needs certain structures in a category; but then the characterization (up to isomorphism) of these structures via categorical properties (typically, adjointness properties) sometimes suggests extra, semantically meaningful rules to add to the logic.

Meta-linguistic conventions

The term-forming operations we will consider in this and subsequent sections will contain various forms of variable-binding, with associated notions of free and bound variables and substitution. To deal with all this in a uniform manner, we adopt a higher-order meta-language of 'arities and expressions' as advocated by Martin-Löf. See [Nordström *et al.*, 1990, Chapter 3] for an extended discussion of this approach. For our purposes it is sufficient to take this meta-theory to be a simply typed lambda calculus with α-, β-, and η-conversion, whose terms will be called *meta-expressions*. The result of applying meta-expression e to meta-expression e' will be denoted $e(e')$, with a repeated application such as $e(e')(e'')$ abbreviated to $e(e', e'')$; in certain situations the application $e(e')$ will be written $e_{e'}$. Lambda abstraction in the meta-language will be denoted $(x)e$, with a repeated lambda abstraction such as $(x)(y)e$ abbreviated to $(x, y)e$. Equality of meta-expressions up to α-, β-, and η-conversion will be denoted $e \equiv e'$. Thus one has $(x)e \equiv (y)(e[y/x])$, $((x)e)(e') \equiv e[e'/x]$ and $(x)(e(x)) \equiv e$.

The types of the meta-language will be called *arities*. For the moment we will assume that the arities are built up from two ground arities, $\textsc{Types}$ and $\textsc{Terms}$, standing respectively for the syntactic categories of object language types and object language terms. Higher function arities will be indicated by $\textsc{Arity} \twoheadrightarrow \textsc{Arity}'$. So for example, $\textsc{Types} \twoheadrightarrow \textsc{Terms}$ is the arity of functions assigning a term expression to each type expression. For a many-sorted signature, *Sg*, as in Section 2, we now regard the function symbols F of *Sg* are meta-constants of arity $\textsc{Terms}^n \twoheadrightarrow \textsc{Terms}$ for various n, and regard the sorts as meta-constants of arity $\textsc{Types}$. The countably infinite set *Var* of variables used in that section will be identified with the metavariables of arity $\textsc{Terms}$.

Throughout this section we will assume that such a signature *Sg* has been given. On the category-theoretic side, $\mathcal{C}$ will denote a category with at least finite products, and we assume that a structure for *Sg* in $\mathcal{C}$ has been given. The simple types and their associated terms will be built up over the sorts of *Sg*. In this context, it is more usual to refer to the sorts of *Sg* as *ground types*. As before, terms and equations will only be considered in the presence of finite contexts assigning types to variables.

Remark 3.0.1 (Uniqueness of typing derivations and uniqueness of types). The various rules we will give for forming terms-in-context and equations-in-context are meant to augment the basic ones of equational logic given in Section 2 (i.e. (2.2), (2.4) and (2.6)–(2.4)). The new rules retain the property that there is at most one derivation of a typing judgement $M : \sigma\ [\Gamma]$. Thus the definition of its meaning as a morphism $[\![M[\Gamma]]\!] : [\![\Gamma]\!] \longrightarrow [\![\sigma]\!]$ in $\mathcal{C}$ can be given unambiguously by induction on the structure of this derivation. The same is true for the rules considered in Section 5. It is only when we come to consider dependent type theories in Section 6 that this property breaks down and a more careful approach to the categorical semantics must be adopted. We will also take care to retain the property of derivable typing judgements, $M : \sigma\ [\Gamma]$, that σ is uniquely determined by the term M and the context Γ. This is the reason for the appearance of type subscripts in the syntax given below for the introduction terms of disjoint union, function and list types.

Remark 3.0.2 (Implicit contexts). In order to make the statement of rules less cluttered, from now on we will adopt the convention that a left-hand part of a context will not be shown if it is common to all the judgements in a rule. Thus for example rule (3.2) below is really an abbreviation for

$$\frac{C : \sigma + \sigma'\ [\Gamma] \quad N(x) : \tau\ [\Gamma, x : \sigma] \quad N'(x') : \tau\ [\Gamma, x' : \sigma]}{\mathsf{case}(C, N, N') : \tau\ [\Gamma]}$$

Remark 3.0.3 (Congruence rules). The introduction and elimination rules for the various datatype constructors to be given below have associated with them congruence rules expressing the fact that the term-forming constructions respect equality. For example, the congruence rules corresponding to (3.1) and (3.2) are:

$$\frac{M_1 = M_2 : \sigma}{\mathsf{inl}_{\sigma'}(M_1) = \mathsf{inl}_{\sigma'}(M_2) : \sigma + \sigma'} \qquad \frac{M'_1 = M'_2 : \sigma'}{\mathsf{inl}_{\sigma}(M'_1) = \mathsf{inl}_{\sigma}(M'_2) : \sigma + \sigma'}$$

$$\frac{C_1 = C_2 : \sigma + \sigma' \quad N_1(x) = N_2(x) : \tau\ [x : \sigma] \quad N'_1(x) = N'_2(x) : \tau\ [x : \sigma]}{\mathsf{case}(C_1, N_1, N'_1) = \mathsf{case}(C_2, N_2, N_2,) : \tau}\ .$$

We will not bother to give such congruence rules explicitly in what follows.

3.1 Disjoint union types

We will use infix notation and write $\sigma + \sigma'$ rather than $+(\sigma, \sigma')$ for the *disjoint union* of two types σ and σ'. (Thus this type constructor has arity Types $\twoheadrightarrow$ Types $\twoheadrightarrow$ Types.) The rules for introducing terms of type $\sigma+\sigma'$ are

$$\frac{M : \sigma}{\mathsf{inl}_{\sigma'}(M) : \sigma + \sigma'} \qquad \frac{M' : \sigma'}{\mathsf{inr}_{\sigma}(M') : \sigma + \sigma'} \tag{3.1}$$

and the corresponding elimination rule is

$$\frac{C : \sigma + \sigma' \quad N(x) : \tau\ [x : \sigma] \quad N'(x') : \tau\ [x' : \sigma]}{\mathsf{case}(C, N, N') : \tau} \tag{3.2}$$

What structure in a category $\mathcal{C}$ with finite products is needed to define $[\![M[\Gamma]]\!]$ in the presence of these new term-forming operations? We already have at our disposal the general framework for interpreting equational logic in categories with finite products, given in the previous section. Since this is our first example of the technique, we will proceed with some care in applying this framework.

Formation. Since types σ get interpreted as objects $X = [\![\sigma]\!]$ of $\mathcal{C}$, we need a binary operation, $X, X' \mapsto X + X'$, on objects in order to interpret the formation of disjoint union types as:

$$[\![\sigma + \sigma']\!] \overset{\text{def}}{=} [\![\sigma]\!] + [\![\sigma']\!]$$

Introduction. Since terms-in-context $M : \sigma\ [\Gamma]$ get interpreted as morphisms from $I = [\![\Gamma]\!]$ to $X = [\![\sigma]\!]$, to interpret the introduction rules (3.1) we need functions on hom-sets of the form

$$\mathcal{C}(I, X) \longrightarrow \mathcal{C}(I, X + X') \qquad \mathcal{C}(I, X') \longrightarrow \mathcal{C}(I, X + X')$$

that can be applied to $[\![M[\Gamma]]\!]$ and $[\![M']\!][\Gamma]$ respectively to give the interpretation of $\mathsf{inl}_{\sigma'}(M)$ and $\mathsf{inr}_{\sigma}(M')$. Now in the syntax, ***substitution commutes with term formation***, and since substitution is interpreted by composition in $\mathcal{C}$, it is necessary that the above functions be *natural* in I. Therefore by the Yoneda Lemma (see [Mac Lane, 1971, III.2]), there are morphisms $X \longrightarrow X + X'$ and $X' \longrightarrow X + X'$ which induce the above functions on hom-sets via composition. So associated with the object $X + X'$ we need morphisms

$$\mathit{inl}_{X,X'} : X \longrightarrow X + X' \quad \text{and} \quad \mathit{inr}_{X,X'} : X \longrightarrow X + X'$$

and can then define

$$\begin{aligned} [\![\mathsf{inl}_{\sigma'}(M)[\Gamma]]\!] &\overset{\text{def}}{=} \mathit{inl}_{[\![\sigma]\!],[\![\sigma']\!]} \circ [\![M[\Gamma]]\!] \\ [\![\mathsf{inr}_{\sigma}(M')[\Gamma]]\!] &\overset{\text{def}}{=} \mathit{inr}_{[\![\sigma]\!],[\![\sigma']\!]} \circ [\![M[\Gamma]]\!] \end{aligned}$$

Elimination. Turning now to the interpretation of the elimination rule (3.2), we need a ternary function on hom-sets of the form

$$\mathit{case}_I : \mathcal{C}(I, X + X') \times \mathcal{C}(I \times X, Y) \times \mathcal{C}(I \times X', Y) \longrightarrow \mathcal{C}(I, Y)$$

and as before, it must be natural in I because of the way that substitution is modelled by composition in $\mathcal{C}$. Naturality here means that for each $g : I' \longrightarrow I$

$$case_I(c, n, n') \circ g = case_{I'}(c \circ g, n \circ (g \times id), n' \circ (g \times id)).$$

Applying this with π_2, $n \circ (\pi_1 \times id)$, and $n' \circ (\pi_1 \times id)$ for c, n, and n' respectively, gives

$$case_I(c, n, n') = case_{I \times (X+X')}(\pi_2, n \circ (\pi_1 \times id), n' \circ (\pi_1 \times id)) \circ \langle id, c \rangle$$

since $\pi_2 \circ \langle id, c\rangle = c$ and $(\pi_1 \times id) \circ (\langle id, c \rangle \times id) = id$. So in fact $case_I(c, n, n')$ can be expressed more fundamentally as a composition $\{n|n'\}_I \circ \langle id, c\rangle$ where

$$\{-|-\}_I : \mathcal{C}(I \times X, Y) \times \mathcal{C}(I \times X', Y) \longrightarrow \mathcal{C}(I \times (X + X'), Y) \qquad (3.3)$$

is a family of functions, natural in I. Thus the semantics of case-terms becomes:

$$[\![\mathsf{case}(C, N, N')[\Gamma]]\!] \stackrel{\text{def}}{=} \{[\![N(x)[\Gamma, x : \sigma]]\!] | [\![N'(x')[\Gamma, x' : \sigma']]\!]\}_{[\![\Gamma]\!]} \circ \langle id, [\![C[\Gamma]]\!]\rangle.$$

Equality. Next we consider equality rules for $\sigma + \sigma'$. Consider first the familiar rules:

$$\frac{M : \sigma \quad N(x) : \tau\ [x : \sigma] \quad N'(x') : \tau\ [x' : \sigma']}{\mathsf{case}(\mathsf{inl}_{\sigma'}(M), N, N') = N(M) : \tau} \qquad (3.4)$$

$$\frac{M' : \sigma' \quad N(x) : \tau\ [x : \sigma] \quad N'(x') : \tau\ [x' : \sigma']}{\mathsf{case}(\mathsf{inr}_{\sigma}(M'), N, N') = N'(M') : \tau} \qquad (3.5)$$

Writing $m : I \longrightarrow X$ for $[\![M[\Gamma]]\!]$, $m' : I \longrightarrow X'$ for $[\![M'[\Gamma]]\!]$, $n : I \times X \longrightarrow Y$ for $[\![N(x)[\Gamma, x : \sigma]]\!]$ and $n' : I \times X' \longrightarrow Y$ for $[\![N'(x')[\Gamma, x' : \sigma']]\!]$, then from above we have that $[\![\mathsf{case}(\mathsf{inl}_{\sigma'}(M), N, N')[\Gamma]]\!]$ is $\{n|n'\}_I \circ \langle id, inl_{X,X'} \circ m\rangle$ and $[\![\mathsf{case}(\mathsf{inr}_{\sigma}(M), N, N')[\Gamma]]\!]$ is $\{n|n'\}_I \circ \langle id, inr_{X,X'} \circ m\rangle$. From Lemma 2.2.1 giving the semantics of substitution, we also have that $[\![N(M)[\Gamma]]\!]$ is $n \circ \langle id, m\rangle$ and that $[\![N(M')[\Gamma]]\!]$ is $n' \circ \langle id, m\rangle$. Consequently the structure on $\mathcal{C}$ for interpreting terms associated with disjoint union types described above, is sound for the rules (3.4) and (3.5) provided the equations

$$\begin{aligned} \{n|n'\}_I \circ \langle id, inl_{X,X'} \circ m\rangle &= n \circ \langle id, m\rangle \\ \{n|n'\}_I \circ \langle id, inr_{X,X'} \circ m\rangle &= n' \circ \langle id, m\rangle \end{aligned}$$

hold for all m, n and n' (with appropriate domains and codomains). Clearly for this it is sufficient to require for all n and n' that

$$\{n|n'\}_I \circ \langle id, inl_{X,X'}\rangle = n \qquad (3.6)$$

$$\{n|n'\}_I \circ \langle id, inr_{X,X'}\rangle \quad = \quad n' \tag{3.7}$$

and the naturality of $\{-|-\}_I$ ensures that (3.6) and (3.7) are also necessary for the previous two equations to hold for all m, n and n'.

So we require that $\{-|-\}_{-}$ provides a natural left inverse to the natural transformation

$$\langle (id \times inl_{X,X'})^*, (id \times inr_{X,X'})^* \rangle : \\ \mathcal{C}(- \times (X + X'), Y) \longrightarrow \mathcal{C}(- \times X, Y) \times \mathcal{C}(- \times X', Y) \tag{3.8}$$

which sends f to $(f \circ (id \times inl_{X,X'}), f \circ (id \times inr_{X,X'}))$. If furthermore, we require that it provide a two-sided inverse, so that

$$f = \{f \circ (id \times inl_{X,X'}) | f \circ (id \times inr_{X,X'})\}_I \tag{3.9}$$

then the categorical semantics becomes sound for a further rule for $\sigma + \sigma'$, namely:

$$\frac{C : \sigma + \sigma' \quad F(z) : \tau \; [z : \sigma + \sigma']}{\mathsf{case}(C, (x)F(\mathsf{inl}_{\sigma'}(x)), (x')F(\mathsf{inr}_{\sigma}(x'))) = F(C) : \tau} \tag{3.10}$$

(Recall that an expression like $(x)F(\mathsf{inl}_{\sigma'}(x))$ denotes a lambda abstraction in the meta-language.)

If we think of (3.4) and (3.5) as the 'β-rules' for disjoint union, then (3.10) is an 'η-rule'. As we will see, the rule is necessary if we wish $\sigma + \sigma'$ to be characterized up to isomorphism by σ and σ'.

Now taking the component of (3.8) at the terminal object *1* and using the canonical isomorphisms $\pi_2 : 1 \times (-) \cong (-)$, gives that

$$\langle inl^*_{X,X'}, inr^*_{X,X'} \rangle : \mathcal{C}(X + X', Y) \longrightarrow \mathcal{C}(X, Y) \times \mathcal{C}(X', Y)$$

is a bijection for each Y. By definition, this means that

$$X \overset{inl_{X,X'}}{\longrightarrow} X + X' \overset{inr_{X,X'}}{\longleftarrow} X'$$

is a binary coproduct diagram in $\mathcal{C}$. We will denote the inverse bijection to $\langle inl^*_{X,X'}, inr^*_{X,X'} \rangle$ at a pair of morphisms $f : X \longrightarrow Y, f' : X' \longrightarrow Y$ by

$$\begin{pmatrix} f \\ f' \end{pmatrix} : X + X' \longrightarrow Y.$$

So we are led to require that $\mathcal{C}$ have binary coproducts. But this is not all: we can now compose the the natural bijection (3.8) with

$$\mathcal{C}(- \times X, Y) \times \mathcal{C}(- \times X', Y) \cong \mathcal{C}((- \times X) + (- \times X'), Y)$$

to deduce that for each I, X, X', composition with the canonical morphism

$$d \stackrel{\text{def}}{=} \left(\begin{array}{c} id \times inl_{X,X'} \\ id \times inr_{X,X'} \end{array} \right) : (I \times X) + (I \times X') \longrightarrow I \times (X + X') \qquad (3.11)$$

induces a bijection on hom-sets, and hence by the Yoneda Lemma, that d must be an isomorphism. In this case we say that *binary products distribute over binary coproducts* in $\mathcal{C}$, or simply that $\mathcal{C}$ has *stable* binary coproducts. Thus the construct $\{-|-\}_{-}$ takes $n : I \times X \longrightarrow Y$ and $n' : I \times X' \longrightarrow Y$ to

$$\{n|n'\}_I \stackrel{\text{def}}{=} \left(\begin{array}{c} n \\ n' \end{array} \right) \circ d^{-1} : I \times (X + X') \longrightarrow Y$$

Defining satisfaction of equations-in-context over this richer collection of terms just as in Section 2.2, the Soundness Theorem 2.2.3 extends to include the equality rules for disjoint unions, i.e. *if $\mathcal{C}$ is a category with finite products and stable binary coproducts, then the equations-in-context that are satisfied by a structure in $\mathcal{C}$ are closed under the rules of Equational Logic augmented by rules* (3.4), (3.5) *and* (3.10). As with all categorical structure defined by universal properties, coproducts in a category are unique up to isomorphism. So we have that the structure needed in a category to interpret disjoint unions with all three equality rules is essentially unique. If we do not wish to model the third equality rule (3.10), then we have seen that a weaker structure than stable coproducts is sufficient—just $inl_{X,X'}$, $inr_{X,X'}$ and an operation $\{-|-\}_{-}$ satisfying the two equations (3.6) and (3.7), together with the naturality condition:

$$\{n \circ (g \times id) | n' \circ (g \times id)\}_{I'} = \{n|n'\}_I \circ (g \times id).$$

(This naturality is automatic in the presence of the uniqueness rule (3.10).) However, in this case there may be several non-isomorphic 'disjoint unions' for X and X' in $\mathcal{C}$. Put another way, provided the third equality rule (3.10) is included, the rules for disjoint unions *characterize this type constructor up to bijection.* This kind of observation seems important from a foundational point of view, and arises very naturally from the categorical approach to logic.

The stability of binary coproducts in $\mathcal{C}$ says precisely that for each object I, the functor $I \times (-) : \mathcal{C} \longrightarrow \mathcal{C}$ preserves binary coproducts. Since coproducts, being colimits, are preserved by any functor with a right adjoint, stability is automatic if each $I \times (-) : \mathcal{C} \longrightarrow \mathcal{C}$ has a right adjoint. The value of this right adjoint at an object X is by definition the *exponential object* $I \rightarrow X$. In particular *if $\mathcal{C}$ is a cartesian closed category, then coproducts in $\mathcal{C}$ are automatically stable*, since all exponential objects exist in this case by definition. We will see in Section 3.3 that exponential objects model function types. So if one is only concerned with modelling disjoint sums in a higher-order setting, one need only verify the presence of coproducts in the category. Here is an example of a category with non-stable coproducts (and hence without exponentials).

Example 3.1.1. Recall that the category of complete, atomic boolean algebras and boolean homomorphisms is equivalent to the opposite of the category of sets and functions. In $\mathcal{S}et$ one generally does not have

$$I + (X \times X') \cong (I + X) \times (I + X') \ .$$

(Consider the case $I = X = X' = 1$, for example.) Thus in the opposite of the category of sets, binary coproducts are not in general stable, and hence nor are they stable in the category of complete atomic boolean algebras. Therefore, even though this category has coproducts (it has all small limits and colimits), we cannot soundly interpret disjoint union types in it.

Remark 3.1.2 (Empty type). We have treated binary disjoint union types. The 'nullary' version of this is the empty type *null*, with rules:

$$\frac{C : \mathit{null}}{\mathsf{empty}_\tau(C) : \tau} \qquad \frac{C : \mathit{null} \quad F(z) : \tau \ [z : \mathit{null}]}{\mathsf{empty}_\tau(C) = F(C) : \tau}$$

These rules can be soundly interpreted in $\mathcal{C}$ if it has a *stable initial object*, 0. This means that not only is 0 initial (i.e. for each object X, there is a unique morphism $0 \longrightarrow X$), but so also is $I \times 0$ for any object I. This latter condition is equivalent to requiring that $\pi_2 : I \times 0 \longrightarrow 0$ is an isomorphism. As before, this stability condition is automatic if the functors $I \times (-) : \mathcal{C} \longrightarrow \mathcal{C}$ have right adjoints, and thus in particular, an initial object in a cartesian closed category is automatically stable.

3.2 Product types

One could be forgiven for jumping straight to the conclusion that product types $\sigma \times \sigma'$ should be interpreted by binary categorical product in a category $\mathcal{C}$. However, as we have seen in Section 2, the finite product structure of $\mathcal{C}$ is there primarily to enable terms involving multiple variables to be interpreted. Also, we will use the elimination rule for products that is derived systematically from its formation and introduction rule, rather than the more familiar formulation in terms of first and second projection. (See [Backhouse *et al.*, 1989] for comments on this issue in general.) Nevertheless, with a full set of equality rules including the analogue of the 'surjective pairing' rule, binary products in $\mathcal{C}$ are exactly what is required to model product types. If one does not wish to model the surjective pairing rule, then a weaker structure on $\mathcal{C}$ (not determined uniquely up to isomorphism) suffices.

Recall that the introduction rule and corresponding elimination rule for product types are

$$\frac{M : \sigma \quad M' : \sigma'}{\mathsf{pair}(M, M') : \sigma \times \sigma'} \tag{3.12}$$

$$\frac{P : \sigma \times \sigma' \quad N(x, x') : \tau\ [x : \sigma, x' : \sigma']}{\mathsf{split}(P, N) : \tau} \tag{3.13}$$

(together with associated congruence rules—cf. Remark 3.0.3). By a similar analysis to that in Section 3.1, one finds that the following structure on $\mathcal{C}$ is needed to interpret these new terms:

- for all objects X, X', an object $X * X'$;
- for all objects X, X', a morphism $pair_{X,X'} : X \times X' \longrightarrow X * X'$;
- for all objects I, X, X', Y, a function

$$split_I : \mathcal{C}(I \times X \times X', Y) \longrightarrow \mathcal{C}(I \times (X * X'), Y)$$

  that is natural in I (so that $split_{I'}(n \circ (g \times id)) = split_I(n) \circ (g \times id)$ holds for all $n : I \times X \times X' \longrightarrow Y$ and $g : I' \longrightarrow I$).

The semantics of product types and terms in $\mathcal{C}$ are then:

$$\begin{aligned}
[\![\sigma \times \sigma']\!] &\overset{\text{def}}{=} [\![\sigma]\!] * [\![\sigma']\!] \\
[\![\mathsf{pair}(M, M')[\Gamma]]\!] &\overset{\text{def}}{=} pair_{[\![\sigma]\!],[\![\sigma']\!]} \circ \langle [\![M[\Gamma]]\!], [\![M'[\Gamma]]\!] \rangle \\
[\![\mathsf{split}(P, N)[\Gamma]]\!] &\overset{\text{def}}{=} split_{[\![\Gamma]\!]}([\![N(x, x')[\Gamma, x : \sigma, x' : \sigma']]\!]) \circ \langle id, [\![P[\Gamma]]\!] \rangle
\end{aligned}$$

The equality rules for product types are:

$$\frac{M : \sigma \quad M' : \sigma' \quad N(x, x') : \tau\ [x : \sigma, x' : \sigma']}{\mathsf{split}(\mathsf{pair}(M, M'), N) = N(M, M') : \tau} \tag{3.14}$$

$$\frac{P : \sigma \times \sigma' \quad N(z) : \tau\ [z : \sigma \times \sigma']}{\mathsf{split}(P, (x, x')N(\mathsf{pair}(x, x'))) = N(P) : \tau} \tag{3.15}$$

For the above semantics to be sound for (3.14), one needs that that for each $n : I \times X \times X' \longrightarrow Y$ in $\mathcal{C}$

$$split_I(n) \circ (id \times pair_{X,X'}) = n.$$

In other words, the natural transformation induced by pre-composition with $(id \times pair_{X,X'})$

$$(id \times pair)^*_{X,X'} : \mathcal{C}(- \times (X * X'), Y) \longrightarrow \mathcal{C}(- \times X \times X', Y) \tag{3.16}$$

should have left inverse given by $n \mapsto split_I(n)$. This condition does not suffice to determine $X * X'$ uniquely up to isomorphism: there may be several different such 'weak product' structures on $\mathcal{C}$. However, if we insist that (3.15) is also satisfied, i.e. that

$$split_I(n \circ (id \times pair_{X,X'})) = n,$$

then $split\_$ becomes a two-sided inverse for (3.16). By considering the component of this natural isomorphism at the terminal object, one has that pre-composition with $pair : X \times X' \longrightarrow X * X'$ induces a bijection on hom-sets, and hence is itself an isomorphism. Consequently to interpret product types satisfying both equality rules, the structure needed on $\mathcal{C}$ must be essentially the existing binary product structure (and so in particular, is unique up to isomorphism). Thus the semantics of product types and terms in $\mathcal{C}$ simplifies to:

$$\begin{array}{rcl} [\![\sigma \times \sigma']\!] & \stackrel{\text{def}}{=} & [\![\sigma]\!] \times [\![\sigma']\!] \\ [\![\mathsf{pair}(M, M')[\Gamma]]\!] & \stackrel{\text{def}}{=} & \langle [\![M[\Gamma]]\!], [\![M'[\Gamma]]\!] \rangle \\ [\![\mathsf{split}(P, N)[\Gamma]]\!] & \stackrel{\text{def}}{=} & [\![N(x, x')[\Gamma, x : \sigma, x' : \sigma']]\!] \circ \langle id, [\![P[\Gamma]]\!] \rangle. \end{array}$$

In particular, the definable first and second projections

$$\begin{array}{rcl} \mathsf{fst}(z) & \stackrel{\text{def}}{=} & \mathsf{split}(z, (x, x')x) \\ \mathsf{snd}(z) & \stackrel{\text{def}}{=} & \mathsf{split}(z, (x, x')x') \end{array}$$

when interpreted give the first and second projection morphisms as one might hope:

$$\begin{array}{rcl} [\![\mathsf{fst}(z)[z : \sigma \times \sigma']]\!] & = & \pi_1 : [\![\sigma]\!] \times [\![\sigma']\!] \longrightarrow [\![\sigma]\!] \\ [\![\mathsf{snd}(z)[z : \sigma \times \sigma']]\!] & = & \pi_2 : [\![\sigma]\!] \times [\![\sigma']\!] \longrightarrow [\![\sigma']\!]. \end{array}$$

Remark 3.2.1 (One-element type). The 'nullary' version of binary product types is the one-element type *unit* with the following rules (together with associated congruence rules). As for product types, we use the possibly less familiar elimination rule derived systematically from the form of formation and introduction.

$$\frac{}{\star : unit} \qquad \frac{U : unit \quad N : \tau}{\mathsf{proj}(U, N) : \tau}$$

$$\frac{U : unit \quad N(z) : \tau \ [z : unit]}{\mathsf{proj}(U, N(\star)) = N(U) : \tau}$$

The last rule is the analogue of rule (3.15)—in other words, it is the 'η-rule' for this type. The 'β-rule' for *unit* (analogous to (3.14)) is

$$\frac{N : \tau}{\mathsf{proj}(\star, N) = N : \tau}$$

but it is easy to see that in fact this rule is derivable from the above rules, as is the rule

$$\frac{U : \mathit{unit}}{U = \star : \mathit{unit}}$$

Using an argument similar to that for product types, one finds that these rules can be interpreted soundly in $\mathcal{C}$ provided it has a terminal object.

3.3 Function types

Since types σ get interpreted as objects $X = [\![\sigma]\!]$ of $\mathcal{C}$, we need a binary operation, $X, X' \mapsto X \to X'$, on objects in order to interpret *function types* $\sigma \to \sigma'$ as follows:

$$[\![\sigma \to \sigma']\!] \stackrel{\text{def}}{=} [\![\sigma]\!] \to [\![\sigma']\!]$$

Introduction. The introduction rule for function types is:

$$\frac{F(x) : \tau \; [x : \sigma]}{\lambda_\sigma(F) : \sigma \to \tau} \; . \tag{3.17}$$

To interpret this rule in a category $\mathcal{C}$ with finite products we need a function on hom-sets of the form

$$\mathit{cur}_I : \mathcal{C}(I \times X, Y) \longrightarrow \mathcal{C}(I, X \to Y)$$

in order to define

$$[\![\lambda_\sigma(F)[\Gamma]]\!] \stackrel{\text{def}}{=} \mathit{cur}_{[\![\Gamma]\!]}([\![F(x)[\Gamma, x : \sigma]]\!]).$$

In order to preserve the basic property of the semantics that substitution is modelled by composition, we require the above functions to be natural in I, meaning

$$\mathit{cur}_{I'}(f \circ (g \times \mathit{id})) = \mathit{cur}_I(f) \circ g$$

holds for all $f : I \times X \longrightarrow Y$ and $g : I' \longrightarrow I$.

Elimination. We will use the familiar elimination rule involving application rather that the rule systematically derived from formation and introduction, which involves judgements with higher-order contexts (see [Nordström *et al.*, 1990, Section 7.2]).

$$\frac{M : \sigma \to \tau \quad N : \sigma}{\mathsf{ap}(M, N) : \tau} \tag{3.18}$$

To interpret this rule, we need a function on hom-sets of the form

$$\mathcal{C}(I, X{\to}Y) \times \mathcal{C}(I, X) \longrightarrow \mathcal{C}(I, Y)$$

which is natural in I (in order to respect the semantics of substitution in terms of composition). Naturality in I of these functions implies that they are induced by a morphism

$$ap_{X,Y} : (X{\to}Y) \times X \longrightarrow Y$$

and we define

$$[\![\mathsf{ap}(M,N)[\Gamma]]\!] \stackrel{\text{def}}{=} ap_{[\![\sigma]\!],[\![\sigma']\!]} \circ \langle [\![M[\Gamma]]\!], [\![N[\Gamma]]\!] \rangle.$$

Equality. The rule of β-conversion is

$$\frac{F(x) : \tau\ [x : \sigma] \quad N : \sigma}{\mathsf{ap}(\lambda_\sigma(F), N) = F(N) : \tau} \tag{3.19}$$

The structure in $\mathcal{C}$ we have described for interpreting function types is sound for this rule provided

$$ap_{X,Y} \circ \langle cur_I(f), n \rangle = f \circ \langle id, n \rangle$$

holds for all $f : I \times X \longrightarrow Y$ and $n : I \longrightarrow X$. For this it is sufficient, and because of the naturality of *cur* also necessary that

$$ap_{X,Y} \circ (cur_I(f) \times id) = f \tag{3.20}$$

hold for all $f : I \times X \longrightarrow Y$.

The rule for η-conversion is

$$\frac{M : \sigma{\to}\tau}{\lambda_\sigma((x)\mathsf{ap}(M,x)) = M : \sigma{\to}\tau} \tag{3.21}$$

The semantics is sound for this rule provided that for all $m : I \longrightarrow (X{\to}Y)$

$$cur_I(ap_{X,Y} \circ (m \times id)) = m. \tag{3.22}$$

In fact the naturality of $cur_{\_}$ implies that this holds if and only if for each X and Y

$$cur_I(ap_{X,Y}) = id_{X\to Y}.$$

Now (3.20) and (3.22) say precisely that the natural transformation

$$\mathcal{C}(- \times X, Y) \longrightarrow \mathcal{C}(-, X \to Y)$$

given by $f \mapsto ap_{X,Y} \circ (f \times id)$ is a bijection with inverse cur_{-}. Thus, by definition, $X \to Y$ is the *exponential* of Y by X in the category $\mathcal{C}$, with evaluation morphism $ap_{X,Y} : (X \to Y) \times X \longrightarrow Y$. Recall that by definition, a cartesian closed category has all finite products and exponentials. Thus we have: *function types satisfying β- and η-conversion can be soundly interpreted in $\mathcal{C}$ provided it is cartesian closed.*

3.4 Inductive types

In this section we consider the categorical interpretation of inductively defined datatypes (such as the natural numbers, lists and trees) using initial algebras. For definiteness we will concentrate on the case of the list type constructor, $\sigma \mapsto \sigma\,list$.

Recall that the introduction rules and corresponding elimination rule for $\sigma\,list$ are:

$$\frac{}{\mathsf{nil}_\sigma : \sigma\,list} \qquad \frac{M : \sigma \quad L : \sigma\,list}{\mathsf{cons}(M, L) : \sigma\,list} \tag{3.23}$$

$$\frac{F(x,l,x') : \sigma' \;[x : \sigma, l : \sigma\,list, x' : \sigma'] \quad L : \sigma\,list \quad M' : \sigma'}{\mathsf{listrec}(F, L, M') : \sigma'} \tag{3.24}$$

By a process of analysis that one hopes is becoming familiar by now, one finds that the following structure on a category $\mathcal{C}$ with finite products is needed to interpret these new syntax.

- For each object X, an object LX equipped with morphisms $nil_X : 1 \longrightarrow LX$ and $cons_X : X \times LX \longrightarrow LX$.
- For all objects I, X, X', an operation sending a morphism $f : I \times X \times LX \times X' \longrightarrow X'$ to a morphism $listrec_I(f) : I \times LX \times X' \longrightarrow X'$, which is natural in I (so that for all $g : I' \longrightarrow I$, $listrec_{I'}(f \circ (g \times id \times id \times id)) = listrec_I(f) \circ (g \times id \times id)$).

The semantics of $\sigma\,list$ and its associated terms in $\mathcal{C}$ is then:

$$\begin{aligned}
[\![\sigma\,list]\!] &\overset{\text{def}}{=} L[\![\sigma]\!] \\
[\![\mathsf{nil}_\sigma[\Gamma]]\!] &\overset{\text{def}}{=} nil_{[\![\sigma]\!]} \circ \langle\,\rangle_{L[\![\sigma]\!]} \\
[\![\mathsf{cons}(M,L)[\Gamma]]\!] &\overset{\text{def}}{=} cons_{[\![\sigma]\!]} \circ \langle [\![M[\Gamma]]\!], [\![L[\Gamma]]\!] \rangle \\
[\![\mathsf{listrec}(F,L,M')[\Gamma]]\!] &\overset{\text{def}}{=} listrec_{[\![\Gamma]\!]}([\![F(x,l,x')[\Gamma, x : \sigma, l : \sigma\,list, x' : \sigma']]\!]) \\
&\qquad \circ \langle id, [\![L[\Gamma]]\!], [\![M'[\Gamma]]\!] \rangle.
\end{aligned}$$

(In the clause for nil_σ, $\langle\,\rangle_{L[\![\sigma]\!]}$ denotes the unique morphism from $L[\![\sigma]\!]$ to the terminal object *1*.)

Now consider the usual equality rules for $\sigma list$:

$$\frac{F(x,l,x') : \sigma' \ [x:\sigma, l:\sigma list, x':\sigma'] \quad M' : \sigma'}{\mathsf{listrec}(F, \mathsf{nil}_\sigma, M') = M' : \sigma'} \tag{3.25}$$

$$\frac{F(x,l,x') : \sigma' \ [x:\sigma, l:\sigma list, x':\sigma'] \quad M:\sigma \quad L:\sigma list \quad M' : \sigma'}{\mathsf{listrec}(F, \mathsf{cons}(M,L), M') = F(M,L,\mathsf{listrec}(F,L,M')) : \sigma'} \tag{3.26}$$

For the semantics to be sound for these two rules one needs that for each $F : I \times X \times LX \times X' \longrightarrow X'$ in $\mathcal{C}$

$$listrec_I(f) \circ \langle \pi_1, nil_X \circ \langle\,\rangle, \pi_2\rangle = \pi_2 : I \times X' \longrightarrow X' \tag{3.27}$$

$$\begin{aligned} & listrec_I(f) \circ \langle \pi_1, cons_X \circ \langle \pi_2, \pi_3\rangle, \pi_4\rangle = \\ & \quad f \circ \langle \pi_1, \pi_2, \pi_3, listrec_I(f) \circ \langle \pi_1, \pi_3, \pi_4\rangle\rangle \\ & \quad : I \times X \times LX \times X' \longrightarrow X' \end{aligned} \tag{3.28}$$

We can go one step further and insist that given f, $listrec_I(f)$ is the *unique* morphism satisfying (3.27) and (3.28). In this case the structure on $\mathcal{C}$ is also sound for the rule

$$\frac{\begin{array}{cc} F(x,l,x') : \sigma' \ [x:\sigma, l:\sigma list, x':\sigma'] & L : \sigma list \quad M' : \sigma' \\ G(l,x') : \sigma' \ [l:\sigma list, x':\sigma'] & G(\mathsf{nil}_\sigma, x') = x' : \sigma' \ [x':\sigma'] \\ \multicolumn{2}{c}{G(\mathsf{cons}(x,l),x') = F(x,l,G(l,x')) : \sigma' \ [x:\sigma, l:\sigma list, x':\sigma']} \end{array}}{G(L,M') = \mathsf{listrec}(F,L,M') : \sigma'} \ . \tag{3.29}$$

With (3.29) and in the presence of product types, the scheme for primitive recursion becomes interdefinable with a simpler scheme of iteration, given by the following rules.

$$\frac{F(x,x') : \sigma' \ [x:\sigma, x':\sigma'] \quad M' : \sigma'}{\mathsf{listit}(F, M', l) : \sigma' \ [l : \sigma list]} \tag{3.30}$$

$$\frac{F(x,x') : \sigma' \ [x:\sigma, x':\sigma'] \quad M' : \sigma'}{\mathsf{listit}(F, M', \mathsf{nil}_\sigma) = M' : \sigma'} \tag{3.31}$$

$$\frac{F(x,x') : \sigma' \ [x:\sigma, x':\sigma'] \quad M' : \sigma'}{\mathsf{listit}(F, M', \mathsf{cons}(x,l)) = F(x, \mathsf{listit}(F,M',l)) : \sigma' \ [x:\sigma, l:\sigma list]} \tag{3.32}$$

$$\frac{\begin{array}{cc} F(x,x') : \sigma' \ [x:\sigma, x':\sigma'] & M' : \sigma' \\ G(l) : \sigma' \ [l : \sigma list] & G(\mathsf{nil}_\sigma) = M' : \sigma' \\ \multicolumn{2}{c}{G(\mathsf{cons}(x,l)) = F(x, G(l)) : \sigma' \ [x:\sigma, l:\sigma list]} \end{array}}{G(l) = \mathsf{listit}(F,M',l) : \sigma' \ [l:\sigma list]} \ . \tag{3.33}$$

For clearly iteration is a special case of primitive recursion, and in the reverse direction one can define

$$\begin{aligned}&\mathsf{listrec}(F, L, M') \overset{\mathrm{def}}{=}\\&\qquad\mathsf{fst}(\mathsf{listit}((x,z)\mathsf{pair}(F(x,\mathsf{fst}(z),\mathsf{snd}(z)),\mathsf{snd(z)}),\mathsf{pair}(\mathsf{nil}_\sigma, M'), L))\end{aligned}$$

and use (3.31), (3.32) and (3.33) to prove that this has the correct properties. To do this, one also has to prove that

$$\mathsf{snd}(\mathsf{listit}((x,z)\mathsf{pair}(F(x,\mathsf{fst}(z),\mathsf{snd}(z)),\mathsf{snd(z)}),\mathsf{pair}(\mathsf{nil}_\sigma, M'), L)) = L,$$

and this requires the uniqueness property (3.33).

In $\mathcal{C}$, the rules (3.30–3.33) correspond to the following universal property of the object LX equipped with morphisms $nil_X : 1 \longrightarrow LX$ and $cons_X : X \times LX \longrightarrow LX$: *for each pair of morphisms $f : I \times X \times X' \longrightarrow X'$ and $m' : I \longrightarrow X'$, there is a unique morphism $g : I \times LX \longrightarrow X'$ such that the following diagrams commute:*

$$\begin{tikzcd}
I \times 1 \arrow[r, "id_I \times nil_X"] \arrow[d, "\pi_2"'] & I \times LX \arrow[d, "g"] \\
1 \arrow[r, "m'"'] & X'
\end{tikzcd}$$

$$\begin{tikzcd}
I \times X \times LX \arrow[r, "id_{I\times X} \times cons_X"] \arrow[d, "\langle \pi_1, \pi_2, g \circ \langle \pi_1, \pi_3 \rangle\rangle"'] & I \times LX \arrow[d, "g"] \\
I \times X \times X' \arrow[r, "f"'] & X'
\end{tikzcd}$$

The uniqueness property of g implies that the operation sending f to g is natural in I—so it is unnecessary to state this as a separate requirement. If the category $\mathcal{C}$ is cartesian closed, then a further simplification can be made: it is sufficient to have the above universal property for $I = 1$ to deduce the general case. If $\mathcal{C}$ also has binary coproducts, then one can combine nil_X and $cons_X$ into a single morphism

$$i \overset{\mathrm{def}}{=} \begin{pmatrix} nil_X \\ cons_X \end{pmatrix} : T(LX) \longrightarrow LX$$

where $T : \mathcal{C} \longrightarrow \mathcal{C}$ is the functor $1+(X\times -)$. The universal property enjoyed by nil_X and $cons_X$ is equivalent to the following property of i: *for each* *morphism $h : T(X') \longrightarrow X'$, there is a unique morphism $g : LX \longrightarrow X'$ with $g \circ i = h \circ T(g)$.* In other words, (LX, i) is an initial object in the category of algebras for the functor T. (Recall that a T-algebra is an object X' of $\mathcal{C}$ equipped with a 'structure' morphism $T(X') \longrightarrow X'$; a morphism of T-algebras is a morphism in $\mathcal{C}$ between the underlying objects of the algebras that makes the evident square involving the structure morphisms commute.)

As usual, this initiality property characterizes LX, nil_X and $cons_X$ uniquely up to isomorphism in $\mathcal{C}$. Moreover, the structure morphism of an initial algebra is always an isomorphism: so i gives an isomorphism between $1 + (X \times LX)$ and LX. (The inverse of i is the unique T-algebra morphism from the initial algebra to $(T(LX), T(i))$.)

Taking the type σ in $\sigma\,list$ to be the one-element type *unit*, one obtains the type of natural numbers. On the categorical side, when X is the terminal object 1 in $\mathcal{C}$, the above universal property of $L1$ is that given by Lawvere in defining the notion of a *natural number object* in a category. Other inductive datatypes (equipped with iterators satisfying uniqueness conditions) can be modelled soundly by requiring $\mathcal{C}$ to have initial algebras for the 'strictly positive' functors $T : \mathcal{C} \longrightarrow \mathcal{C}$—those expressible via a combination of constant functors, products, coproducts, and exponentiation by a fixed object.

Remark 3.4.1 (Weak natural numbers object). Without the uniqueness rules (3.29) or (3.33), the scheme of iteration is weaker than that of primitive recursion. (For example, the expected properties of the predecessor function on the natural numbers cannot be derived from an iterative definition—see [Girard, 1989, page 51].) The corresponding uniqueness part of the universal property defining the notion of natural number object N, makes this notion conditional-equational rather than equational. If this uniqueness requirement is removed, N is termed a *weak* natural numbers object. Lambek [1988] and Román [1989] have studied extensions of the algebraic theory of weak natural numbers involving Mal'cev operations in which the uniqueness of the iterator can be derived.

3.5 Computation types

Moggi [1991] introduces a new approach to the denotational semantics of programming languages based on the categorical notion of a *strong monad* and the corresponding type-theoretic notion of 'computation types'. As a further illustration of the methods developed so far, we will give the semantics of Moggi's 'computational lambda calculus' in a category with finite products equipped with a strong monad.

The syntax of computation types is as follows. We are given a unary type constructor $\sigma \mapsto T\sigma$. One should think of $T\sigma$ as the type of 'computations' of elements of type σ. There are two rules for building terms of computation types:

$$\frac{M : \sigma}{\mathsf{val}(M) : T\sigma} \tag{3.34}$$

$$\frac{E : T\sigma \quad F(x) : T\sigma' \; [x : \sigma]}{\mathsf{let}(E, F) : T\sigma'} \tag{3.35}$$

(together with associated congruence rules—cf. Remark 3.0.3).

Intuitively, $\mathsf{val}(M)$ is the value M regarded as a trivial computation which immediately evaluates to itself; and $\mathsf{let}(E, F)$ denotes the sequential computation which firstly tries to evaluate E to some value $M : \sigma$ and then proceeds to evaluate $F(M)$. These intended meanings motivate the following three equality rules:

$$\frac{M : \sigma \quad F(x) : T\sigma' \; [x : \sigma]}{\mathsf{let}(\mathsf{val}(M), F) = F(M) : T\sigma'} \tag{3.36}$$

$$\frac{E : T\sigma}{\mathsf{let}(E, (x)\mathsf{val}(x)) : T\sigma} \tag{3.37}$$

$$\frac{E : T\sigma \quad F(x) : T\sigma' \; [x : \sigma] \quad G(x') : T\sigma'' \; [x' : \sigma']}{\mathsf{let}(\mathsf{let}(E, F), G) = \mathsf{let}(E, (x)\mathsf{let}(F(x), G)) : T\sigma''} \; . \tag{3.38}$$

To interpret this syntax soundly in a category $\mathcal{C}$ with finite products we need the following structure on $\mathcal{C}$:

- for each object X, an object TX
- for each object X, a morphism $\eta_X : X \longrightarrow TX$;
- for all objects I, X, X', a function

  $$\mathit{lift}_I : \mathcal{C}(I \times X, T(X')) \longrightarrow \mathcal{C}(I \times TX, T(X'))$$

  that is natural in I (so that $\mathit{lift}_I(f) \circ (g \times \mathit{id}) = \mathit{lift}_{I'}(f \circ (g \times \mathit{id}))$ holds for all $f : I \times X \longrightarrow T(X')$ and $g : I' \longrightarrow I$).

Then the semantics of computation types and their associated terms in $\mathcal{C}$ is:

$$\begin{aligned}
[\![T\sigma]\!] &\stackrel{\text{def}}{=} T([\![\sigma]\!]) \\
[\![\mathsf{val}(M)[\Gamma]]\!] &\stackrel{\text{def}}{=} \eta_{[\![\sigma]\!]} \circ [\![M[\Gamma]]\!] \\
[\![\mathsf{let}(E, F)[\Gamma]]\!] &\stackrel{\text{def}}{=} \mathit{lift}_{[\![\Gamma]\!]}([\![F(x)[\Gamma, x : \sigma]]\!]) \circ \langle \mathit{id}, [\![E[\Gamma]]\!] \rangle
\end{aligned}$$

Satisfaction of rules 3.36, 3.37 and 3.38 requires the following equational properties of η_X and lift_I to hold for all $f : I \times X \longrightarrow T(X')$ and g :

$I \times T(X') \longrightarrow T(X'')$. (We have used the naturality of lift_I in I to state these in their simplest form.)

$$\begin{array}{rcl} \mathit{lift}_I(f \circ \langle id, \eta_X \rangle) & = & f \\ \mathit{lift}_I(\eta_X \circ \pi_1) & = & \pi_1 : I \times X \longrightarrow X \\ \mathit{lift}_I(g) \circ \langle \pi_1, \mathit{lift}_I(f) \rangle & = & \mathit{lift}_I(\mathit{lift}_I(g) \circ \langle \pi_1, f \rangle) \end{array}$$

Specifying the structure (T, η, lift) with the above properties is in fact equivalent to specifying a monad on the category $\mathcal{C}$ together with a 'tensorial strength' for the monad: we refer the reader to [Moggi, 1991] and the references therein for more details. Note that unlike the other categorical datatypes considered in this section, for a given category $\mathcal{C}$ the structure of a strong monad is not determined up to isomorphism—there may be many different ones on $\mathcal{C}$. (See *loc. cit.* for several examples with computational significance.)

4 Theories as Categories

Sections 2 and 3 have shown how to model an equational theory, possibly involving some simple datatypes, in a suitably structured category in a way which respects the rules of Equational Logic. The purpose of this section is to demonstrate that this categorical semantics is completely general, in the sense that the category theory provides an alternative, but equivalent formulation of the concepts embodied in the logic. We will see that there is a correspondence between theories and categories, which enables one to view the latter as giving an abstract, or presentation-free notion of theory.

Such a theory–category correspondence relies upon the fact that the categorical formulation of semantics allows us to consider models of a theory in *different* categories. For a given theory *Th*, amongst all these different categories it is possible to construct one which contains the 'most general' model of *Th*. This category is called the 'classifying category' of *Th* and the model it contains is called the 'generic' model. They are characterized by a certain category-theoretic, universal property which depends upon the ability to compare algebras in different categories by transporting them along functors preserving the relevant categorical structure. So we begin by describing this process.

Until Section 4.4 we will confine our attention to algebraic theories versus categories with finite products. *Th* will denote a fixed, many-sorted algebraic theory (as defined in Section 2.1), with underlying signature *Sg*.

4.1 Change of category

Recall that a functor $T : \mathcal{C} \longrightarrow \mathcal{D}$ is said to *preserve* the finite product $X_1 \times \cdots \times X_n$ if the morphisms $T(\pi_i) : T(X_1 \times \cdots \times X_n) \longrightarrow T(X_i)$ make

$T(X_1 \times \cdots \times X_n)$ into the product of the $T(X_i)$ in $\mathcal{D}$. If $\mathcal{D}$ itself has finite products, then this is equivalent to requiring that

$$a \stackrel{\text{def}}{=} \langle T(\pi_1), \ldots, T(\pi_n) \rangle : T(X_1 \times \cdots \times X_n) \longrightarrow T(X_1) \times \cdots \times T(X_n)$$

be an isomorphism. (A stronger condition would be to demand that a be an *identity* morphism—in which case we say that T *strictly* preserves the finite product $X_1 \times \cdots \times X_n$.)

If $\mathcal{C}$ and $\mathcal{D}$ are categories with finite products and $T : \mathcal{C} \longrightarrow \mathcal{D}$ is a functor preserving them, then every structure S for *Sg* in $\mathcal{C}$ gives rise to a structure $T(S)$ in $\mathcal{D}$ defined as follows:

$$\begin{array}{rcl} T(S)[\![\sigma]\!] & \stackrel{\text{def}}{=} & T(S[\![\sigma]\!]) \\ T(S)[\![F]\!] & \stackrel{\text{def}}{=} & T(S[\![F]\!]) \circ a^{-1} \ . \end{array}$$

for all sorts σ and function symbols $F : \sigma_1, \ldots, \sigma_n \longrightarrow \tau$. Since T preserves finite products and the semantics of terms-in-context is defined in terms of these, the meaning of a term in S is mapped by T to its meaning in $T(S)$. More precisely, if $M : \tau\ [\Gamma]$ then

$$\begin{tikzcd} T(S)[\![\Gamma]\!] \arrow[r, phantom, "\overset{a^{-1}}{\cong}"] \arrow[d, "T(S)[\![M[\Gamma]]\!]"'] & T(S[\![\Gamma]\!]) \arrow[d, "T(S[\![M[\Gamma]]\!])"] \\ T(S)[\![\tau]\!] \arrow[r, phantom, "="] & T(S[\![\tau]\!]) \end{tikzcd}$$

commutes in $\mathcal{D}$. Consequently, if S satisfies $M = M' : \tau\ [\Gamma]$, then so does $T(S)$. Thus *finite product preserving functors preserve the satisfaction of equations*. In particular, if S is an algebra in $\mathcal{C}$ for a theory *Th*, then $T(S)$ is an algebra for *Th* in $\mathcal{D}$.

Example 4.1.1. If $\mathcal{C}$ is locally small, then for each object I in $\mathcal{C}$ the hom-functor $\mathcal{C}(I, \_) : \mathcal{C} \longrightarrow \mathcal{S}et$ preserves products (but not strictly so, in general). Thus every algebra S in $\mathcal{C}$ gives rise to an ordinary, set-valued algebra $\mathcal{C}(I, S)$, whose elements are the 'generalized elements of S at stage I' in the sense of Section 2.3.

4.2 Classifying category of a theory

Two contexts over the signature *Sg* are *α-equivalent* if they differ only in their variables; in other words, the list of sorts occurring in each context are equal (and in particular, the contexts are of equal length). Clearly this gives an equivalence relation on the collection of contexts and the set of α-equivalence classes of contexts is in bijection with the set of lists of sorts in *Sg*. Assuming a fixed enumeration $Var = \{v_1, v_2, \ldots\}$ of the set of variables (i.e. of the set of meta-variables of arity TERMS), we can pick a canonical representative context $[v_1 : \sigma_1, \ldots, v_n : \sigma_n]$ for the α-equivalence class corresponding to each list $\sigma_1, \ldots, \sigma_n$ of sorts. We will not distinguish notationally between a context and the α-equivalence class it determines.

Given contexts Γ and $\Gamma' = [y_1 : \tau_1, \ldots, y_n : \tau_m]$, a *context morphism* from Γ to Γ'

$$\gamma : \Gamma \longrightarrow \Gamma'$$

is specified by a list $\gamma = [N_1, \ldots, N_m]$ of terms over *Sg* such that $N_j : \tau_j\ [\Gamma]$ holds for each $j = 1, \ldots, m$. Note that Γ' could be replaced by any α-equivalent context without affecting this definition.

Given another such morphism $\gamma' = [N'_1, \ldots, N'_m]$, we will write

$$\gamma =_{Th} \gamma' : \Gamma \longrightarrow \Gamma'$$

to indicate the judgement that γ and γ' are context morphisms from Γ to Γ' that are *Th-provably equal*: by definition, this means that for each $j = 1, \ldots, m$, $N_j = N'_j : \tau\ [\Gamma]$ is a theorem of *Th*. Rules (2.6), (2.7) and (2.8) imply that *Th*-provable equality of context morphisms (between two given contexts) is an equivalence relation. Moreover, rule (2.9) shows that changing Γ up to α-equivalence will not change the class of γ under this equivalence relation.

The *composition* of context morphisms $\gamma : \Gamma \longrightarrow \Gamma'$ and $\gamma' : \Gamma' \longrightarrow \Gamma''$ is the context morphism $\gamma' \circ \gamma : \Gamma \longrightarrow \Gamma''$ formed by making substitutions:

$$\gamma' \circ \gamma \stackrel{\text{def}}{=} [M_1[\vec{N}/\vec{y}], \ldots, M_n[\vec{N}/\vec{y}]]$$

when $\Gamma' = [y_1 : \tau_1, \ldots, y_n : \tau_m]$, $\gamma = [N_1, \ldots, N_m]$ and $\gamma' = [M_1, \ldots, M_n]$. The fact that $\gamma' \circ \gamma$ does constitute a context morphism from Γ to Γ'' follows from rule (2.4); and rule (2.9) implies that composition respects *Th*-provable equality:

$$\frac{\gamma_1 =_{Th} \gamma_2 : \Gamma \longrightarrow \Gamma' \quad \gamma'_1 =_{Th} \gamma'_2 : \Gamma' \longrightarrow \Gamma''}{\gamma'_1 \circ \gamma_1 =_{Th} \gamma'_2 \circ \gamma_2 : \Gamma \longrightarrow \Gamma''}\ .$$

The usual properties of term substitution imply that composition of context morphisms is associative:

$$\gamma'' \circ (\gamma' \circ \gamma) =_{Th} (\gamma'' \circ \gamma') \circ \gamma : \Gamma \longrightarrow \Gamma'''$$

when $\gamma : \Gamma \longrightarrow \Gamma'$, $\gamma' : \Gamma' \longrightarrow \Gamma''$ and $\gamma'' : \Gamma'' \longrightarrow \Gamma'''$. Moreover, the operations of composition possess units: the *identity* context morphism for $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$,

$$id_\Gamma : \Gamma \longrightarrow \Gamma$$

is given by the list $id_\Gamma = [x_1, \ldots, x_n]$ of variables in Γ; clearly one has

$$\begin{array}{lll} \gamma \circ id_\Gamma & =_{Th} & \gamma : \Gamma \longrightarrow \Gamma' \\ id_{\Gamma'} \circ \gamma & =_{Th} & \gamma : \Gamma \longrightarrow \Gamma' \ . \end{array}$$

We now have all the ingredients necessary to define a category. Specifically, the *classifying category*, $\mathcal{C}\ell(Th)$, of an algebraic theory Th is defined as follows.

Objects of $\mathcal{C}\ell(Th)$ are α-equivalence classes of contexts (or, if you prefer, finite lists of sorts) over the signature of Th.

Morphisms of $\mathcal{C}\ell(Th)$ from one object Γ to another Γ' are equivalence classes of context morphisms for the equivalence relation which identifies $\gamma : \Gamma \longrightarrow \Gamma'$ with $\gamma' : \Gamma \longrightarrow \Gamma'$ just in case $\gamma =_{Th} \gamma' : \Gamma \longrightarrow \Gamma'$. Composition of morphisms in $\mathcal{C}\ell(Th)$ is induced by composition of context morphisms and identities are equivalence classes of identity context morphisms. We will not distinguish notationally between a context morphism and the morphism of $\mathcal{C}\ell(Th)$ which it determines.

Proposition 4.2.1. *$\mathcal{C}\ell(Th)$ has finite products.*

Proof. Clearly, for each context Γ the empty list of terms is the unique context morphism $\Gamma \longrightarrow [\,]$, and so its equivalence class is the unique morphism from Γ to $[\,]$ in $\mathcal{C}\ell(Th)$. Thus the (α-equivalence class of the) empty context $[\,]$ is a terminal object in $\mathcal{C}\ell(Th)$.

Given contexts $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ and $\Gamma' = [y_1 : \tau_1, \ldots, y_m : \tau_m]$, we make use of the given enumeration $Var = \{v_1, v_2, \ldots\}$ of the set of variables to define a context

$$\Gamma \times \Gamma' \overset{\text{def}}{=} [v_1 : \sigma_1, \ldots, v_n : \sigma_n, v_{n+1} : \tau_1, \ldots, v_{n+m} : \tau_m]$$

Then $[v_1, \ldots, v_n]$ and $[v_{n+1}, \ldots, v_{n+m}]$ determine morphisms $\pi_1 : \Gamma \times \Gamma' \longrightarrow \Gamma$ and $\pi_2 : \Gamma \times \Gamma' \longrightarrow \Gamma'$ in $\mathcal{C}\ell(Th)$ which make $\Gamma \times \Gamma'$ into the product of Γ and Γ' in $\mathcal{C}\ell(Th)$. For if $\gamma = [M_1, \ldots, M_n] : \Delta \longrightarrow \Gamma$ and $\gamma' = [N_1, \ldots, N_m] : \Delta \longrightarrow \Gamma'$ then $[M_1, \ldots, M_n, N_1, \ldots, N_m]$ determines a context morphism $\langle \gamma, \gamma' \rangle : \Delta \longrightarrow \Gamma \times \Gamma'$ which is unique up to Th-provability with the property that $\pi_1 \circ \langle \gamma, \gamma' \rangle = \gamma$ and $\pi_2 \circ \langle \gamma, \gamma' \rangle = \gamma'$. ∎

Remark 4.2.2. We could have defined $\mathcal{C}\ell(Th)$ by taking its objects to be just contexts rather than α-equivalence classes of contexts; this would have resulted in a larger category, equivalent to the one defined above (and hence with the same categorical properties). One advantage of the definition we gave is that the particular choice of finite products given in the proof of

Proposition 4.2.1 is *strictly associative* and *strictly unital*: this means that for all objects Γ, Γ' and Γ'', the morphisms

$$\begin{aligned} \langle\langle \pi_1, \pi_1 \circ \pi_2\rangle, \pi_2 \circ \pi_2\rangle &: \Gamma \times (\Gamma' \times \Gamma'') \longrightarrow (\Gamma \times \Gamma') \times \Gamma'' \\ \pi_1 &: \Gamma \times 1 \longrightarrow \Gamma \\ \pi_2 &: 1 \times \Gamma \longrightarrow \Gamma \end{aligned}$$

are not merely isomorphisms (as is always the case), but actually *identity* morphisms.

Remark 4.2.3. There is a close relationship between $\mathcal{C}\ell(Th)$ and the *free Th*-algebras in $\mathcal{S}et$ generated by finitely many indeterminates. Writing Γ for $[x_1 : \sigma_1, \dots, x_n : \sigma_n]$, let $F_{Th}(\Gamma)$ denote the free algebra generated by finitely many indeterminates $x_1, \dots, x_n$ of sorts $\sigma_1, \dots, \sigma_n$ respectively. Then $F_{Th}(\Gamma)$ can be constructed by taking its underlying set at a sort τ to consist of the set of terms M satisfying $M : \tau\ [\Gamma]$, quotiented by the equivalence relation which identifies M with M' just in case $M = M' : \tau\ [\Gamma]$ is a theorem of *Th*. This quotient set is precisely the set of morphisms $\Gamma \longrightarrow [v_1 : \tau]$ in $\mathcal{C}\ell(Th)$. Thus the hom-sets of $\mathcal{C}\ell(Th)$ can be used to construct the free finitely generated *Th*-algebras in $\mathcal{S}et$. Conversely, it can be shown that the category $\mathcal{C}\ell(Th)$ is equivalent to the opposite of the category whose objects are the free, finitely generated *Th*-algebras in $\mathcal{S}et$, and whose morphisms are all *Th*-algebra homomorphisms.

Next we describe the 'generic' model of *Th* in the classifying category. Each sort σ of the underlying signature *Sg* of the theory *Th* determines determines an object in the classifying category of *Th* represented by the context $[v_1 : \sigma]$. We will denote this object of $\mathcal{C}\ell(Th)$ by $G[\![\sigma]\!]$. If $F : \sigma_1, \dots, \sigma_n \longrightarrow \tau$ is a function symbol in *Sg*, then since

$$F(v_1, \dots, v_n) : \tau\ [v_1 : \sigma_1, \dots, v_n : \sigma_n]$$

we have that $[F(v_1, \dots, v_n)]$ is a context morphism from $[v_1 : \sigma_1, \dots, v_n : \sigma_n]$ to $[v_1 : \tau]$. Now the proof of Proposition 4.2.1 shows that in $\mathcal{C}\ell(Th)$ the object $[v_1 : \sigma_1, \dots, v_n : \sigma_n]$ is the product of the objects $[v_i : \sigma_i] = G[\![\sigma_i]\!]$; hence $[F(v_1, \dots, v_n)]$ determines a morphism $G[\![F]\!] : G[\![\sigma_1]\!] \times \cdots \times G[\![\sigma_n]\!] \longrightarrow G[\![\tau]\!]$ in $\mathcal{C}\ell(Th)$. Altogether then, G is a structure for *Sg* in the category $\mathcal{C}\ell(Th)$.

Lemma 4.2.4. *The structure G has the following properties.*

(i) *For each context Γ, the object $G[\![\Gamma]\!]$ in $\mathcal{C}\ell(Th)$ is (the α-equivalence class of) Γ;*

(ii) *If $M : \tau\ [\Gamma]$ then the morphism $G[\![M[\Gamma]]\!] : G[\![\Gamma]\!] \longrightarrow G[\![\tau]\!]$ in $\mathcal{C}\ell(Th)$ is that determined by the context morphism $[M] : \Gamma \longrightarrow [v_1 : \tau]$;*

(iii) *$M = M' : \tau\ [\Gamma]$ is a theorem of Th if and only if $G[\![M[\Gamma]]\!] = G[\![M'[\Gamma]]\!]$. In other words, G satisfies an equation-in-context just*

in case it is a theorem of Th.

Proof. (i) We have already observed that $[v_1 : \sigma_1, \ldots, v_n : \sigma_n]$ is the product $G[\![\sigma_1]\!] \times \cdots \times G[\![\sigma_n]\!]$ in $\mathcal{C}\ell(\mathit{Th})$; but recall from Section 2.2 that this is also the definition of $G[\![[v_1 : \sigma_1, \ldots, v_n : \sigma_n]]\!]$.

(ii) This follows by induction on the structure of M using the explicit description of the product structure of $\mathcal{C}\ell(\mathit{Th})$ given in the proof of Proposition 4.2.1.

(iii) By part (ii), $G[\![M[\Gamma]]\!] = G[\![M'[\Gamma]]\!]$ holds if and only if $[M]$ and $[M']$ determine equal morphisms in $\mathcal{C}\ell(\mathit{Th})$, which by definition means that $M = M' : \tau\ [\Gamma]$ is a theorem of *Th*.

∎

Part (ii) of the above lemma implies in particular that G satisfies the axioms of *Th* and hence is a *Th*-algebra. G will be called the *generic Th-algebra.* It enjoys the following universal property.

Theorem 4.2.5 (Universal property of the generic algebra). *For each category $\mathcal{C}$ with finite products, any Th-algebra S in $\mathcal{C}$ is equal to $\overline{S}(G)$ for some finite product preserving functor $\overline{S} : \mathcal{C}\ell(\mathit{Th}) \longrightarrow \mathcal{C}$. Moreover, the functor $\overline{S}$ is uniquely determined up to (unique) natural isomorphism by the algebra S. ($\overline{S}$ is called the functor classifying the algebra S.)*

Proof. Define $\overline{S}$ on objects Γ by

$$\overline{S}(\Gamma) \stackrel{\text{def}}{=} S[\![\Gamma]\!]$$

and on morphisms $\gamma = [N_1, \ldots, N_m] : \Gamma \longrightarrow \Gamma'$ by

$$\overline{S}(\gamma) \stackrel{\text{def}}{=} \langle S[\![N_1[\Gamma]]\!], \ldots, S[\![N_m[\Gamma]]\!]\rangle : S[\![\Gamma]\!] \longrightarrow S[\![\Gamma']\!]$$

Then the fact that $\overline{S}$ is a functor preserving finite products follows easily from the definition of $\mathcal{C}\ell(\mathit{Th})$ in Section 4.2. Applying $\overline{S}$ to G, one has for a sort σ that $\overline{S}(G)\sigma = \overline{S}(G[\![\sigma]\!]) = S[\![[v_1 : \sigma]]\!] = S[\![\sigma]\!]$, and similarly for a function symbol F that $\overline{S}(G)F = \overline{S}(G[\![F]\!]) = S[\![F]\!]$. Hence $\overline{S}(G) = S$.

Now suppose that $T : \mathcal{C}\ell(\mathit{Th}) \longrightarrow \mathcal{C}$ is another product-preserving functor and that there is an isomorphism, $h : T(G) \cong S$, of *Th*-algebras in $\mathcal{C}$. For each object $\Gamma = [v_1 : \sigma_1, \ldots, v_n : \sigma_n]$ in $\mathcal{C}\ell(\mathit{Th})$, since $\Gamma \cong G[\![\sigma_1]\!] \times \cdots \times G[\![\sigma_n]\!]$, one gets isomorphisms

$$\begin{array}{rcl} T(\Gamma) & \cong & T(G[\![\sigma_1]\!] \times \cdots \times G[\![\sigma_n]\!]) \\ & \cong & T(G)[\![\sigma_1]\!] \times \cdots \times T(G)[\![\sigma_n]\!] \\ & \cong & S[\![\sigma_1]\!] \times \cdots \times S[\![\sigma_n]\!] \\ & \stackrel{\text{def}}{=} & \overline{S}(\Gamma) \end{array}$$

which are the components of a natural isomorphism $k : T \cong \overline{S}$. Finally, note that since T and $\overline{S}$ preserve finite products and every object in $\mathcal{C}\ell(\mathit{Th})$ is a finite product of objects of the form $G[\![\sigma]\!]$, k is uniquely determined by the condition that it induces h. ∎

Corollary 4.2.6.

(i) The classifying category of Th is determined uniquely up to equivalence, and the generic Th-algebra uniquely up to isomorphism by the universal property in Theorem 4.2.5.

(ii) The operation of evaluating a finite product preserving functor from $\mathcal{C}\ell(\mathit{Th})$ to $\mathcal{C}$ at the generic algebra G is the object part of an equivalence of categories:

$$FP(\mathcal{C}\ell(\mathit{Th}),\mathcal{C}) \simeq \mathit{Th}\text{-}ALG(\mathcal{C}) \tag{4.1}$$

where $FP(\mathcal{C}\ell(\mathit{Th}),\mathcal{C})$ is the category of finite product preserving functors and natural transformations from $\mathcal{C}\ell(\mathit{Th})$ to $\mathcal{C}$, and $\mathit{Th}\text{-}ALG(\mathcal{C})$ is the category of Th-algebras and homomorphisms in $\mathcal{C}$.

Proof. Part (i) is a routine consequence of the universal property, but part (ii) deserves further comment.

The statement of Theorem 4.2.5 says that the functor $T \mapsto T(G)$ is essentially surjective, and full and faithful for isomorphisms; hence it gives an equivalence between the category of functors and natural isomorphisms and the category of algebras and algebra isomorphisms. Since this is true for any $\mathcal{C}$ with finite products, we can replace $\mathcal{C}$ by its arrow category $\mathcal{C}^{\cdot\to\cdot}$ (whose objects are the morphisms of $\mathcal{C}$ and whose morphisms are commutative squares in $\mathcal{C}$), which certainly has finite products when $\mathcal{C}$ does. The equivalence for objects and isomorphisms in this case implies that the original functor $T \mapsto T(G)$ is full and faithful, and hence that (4.1) holds. ∎

4.3 Theory-category correspondence

Let $Sg_{\mathcal{C}}$ be the signature defined from a category $\mathcal{C}$ with finite products, as in Section 2.3. As we noted in that section, there is a canonical structure for $Sg_{\mathcal{C}}$ in $\mathcal{C}$. Define $Th_{\mathcal{C}}$ to be the algebraic theory over $Sg_{\mathcal{C}}$ whose axioms are all equations-in-context which are satisfied by this structure. Then the structure is automatically an algebra for this theory, and hence by Theorem 4.2.5 corresponds to a finite product preserving functor $T : \mathcal{C}\ell(Th_{\mathcal{C}}) \longrightarrow \mathcal{C}$. The definition of $Sg_{\mathcal{C}}$ (which names the various objects and morphisms in $\mathcal{C}$) and $Th_{\mathcal{C}}$ (which identifies terms which name the same things in $\mathcal{C}$) entails that T is full, faithful and essentially surjective—and hence is an equivalence of categories. Thus we have that *every category with finite products is equivalent to the classifying category of some many-sorted algebraic theory.* As we noted in Section 2.3, $Th_{\mathcal{C}}$ will have a set of symbols and axioms (rather than a proper class of them) only in the case that $\mathcal{C}$ is a small category.

Starting with $\mathcal{C}$, forming $\mathit{Th}_{\mathcal{C}}$ and then $\mathcal{C}\ell(\mathit{Th}_{\mathcal{C}})$ one obtains back not $\mathcal{C}$ itself, but an equivalent category. What about the reverse process: how are the theories Th and $\mathit{Th}_{\mathcal{C}\ell(\mathit{Th})}$ related? To answer this question, we need an appropriate notion of morphism, or *translation* between algebraic theories. A rather general notion is obtained by declaring a translation $\mathit{Th} \longrightarrow \mathit{Th}'$ to be an algebra for Th in the classifying category of Th'. The syntactic nature of the construction of $\mathcal{C}\ell(\mathit{Th}')$ permits one to give a purely syntactic explanation of this notion of translation, which we omit here. In fact we can form a 2-category of many-sorted algebraic theories, $\mathcal{A}\mathit{lg}$, whose hom-categories are the categories of algebras and homomorphisms in classifying categories:

$$\mathcal{A}\mathit{lg}(\mathit{Th}, \mathit{Th}') \stackrel{\text{def}}{=} \mathit{Th}\text{-}\mathcal{A}\mathit{lg}(\mathcal{C}\ell(\mathit{Th}')) \ .$$

If $\mathcal{F}p$ denotes the 2-category whose objects are categories with finite products and whose hom-categories consist of finite product preserving functors and all natural transformations, then the classifying-category construction is the object part of a 2-functor

$$\mathcal{C}\ell : \mathcal{A}\mathit{lg} \longrightarrow \mathcal{F}p \ .$$

This 2-functor is 'full and faithful' in the sense that it is an equivalence on hom-categories (using (4.1)), and is 'essentially surjective' in the sense that every object of $\mathcal{F}p$ is equivalent to one in the image of $\mathcal{C}\ell$. Thus $\mathcal{C}\ell$ induces an equivalence between $\mathcal{A}\mathit{lg}$ and $\mathcal{F}p$ in an appropriate, up-to-equivalence sense.

Remarks 4.3.1. We close this section by mentioning some consequences of the correspondence.

(i) We can view a particular small category with finite products $\mathcal{C}$ as specifying a (many-sorted) algebraic theory independently of any particular presentation in terms of a signature and axioms: there may be many such presentations whose syntactic details are different, but which nevertheless specify the 'same' theory, in the sense that their classifying categories are equivalent to the given category $\mathcal{C}$. Indeed, once one has this categorical view of algebraic theories, one sees that there are many 'naturally occurring' algebraic theories which do not arise in terms of a presentation with operators and equations. For example, the category whose objects are the finite cartesian powers of the set of natural numbers $\mathbb{N}$ and whose morphisms $\mathbb{N}^m \longrightarrow \mathbb{N}^n$ are n-tuples of m-ary primitive recursive functions, is a category with finite products and hence an algebraic theory. This category is a paradigmatic example of the notion of 'iteration theory' introduced by Elgot:

see the book by Bloom and Esik [1993]. (A reader who takes this advice should be warned that [Bloom and Ésik, 1993] adopts a not uncommon viewpoint that algebraic theories can be identified with categories with finite *coproducts*. Since the 2-category of categories with finite coproducts and functors preserving such is equivalent (under the 2-functor taking a category to its opposite category) to the 2-category of categories with finite products, this viewpoint is formally equivalent to the one presented here. However it does not sit well with the intuitively appealing picture we have built up of the sorts of a theory (objects of a category) as generalized sets and the terms-in-context (morphisms) as generalized functions.)

(ii) If $T : \mathbf{C} \longrightarrow \mathbf{D}$ is a morphism in $\mathcal{F}p$ between small categories, then for any $\mathcal{C} \in \mathcal{F}p$ with small colimits, it can be shown that the functor $T^* : \mathcal{F}p(\mathbf{D}, \mathcal{C}) \longrightarrow \mathcal{F}p(\mathbf{C}, \mathcal{C})$ induced by composition with T has a left adjoint, given by left Kan extension along T. Since T corresponds to a translation between algebraic theories and T^* is the functor restricting algebras along T, this left adjoint provides 'relatively free' constructions on algebras for a wide variety of situations (depending upon the nature of the translation).

(iii) Free constructions (indeed weighted colimits in general) in $\mathcal{F}p$ can be constructed via appropriate syntactic constructions on algebraic theories and translations between them.

4.4 Theories with datatypes

In this section we will examine the effect on the classifying category $\mathcal{C}\ell(\mathit{Th})$ of a theory *Th* when we enrich equational logic with the various datatype constructions considered in Section 3, together with their associated introduction, elimination and equality rules. We will look at product, disjoint union and function types. In each case the classifying category turns out to possess the categorical structure that was used in Section 3 to interpret these datatype constructs. (Similar considerations apply to the other type forming operations considered in that section, namely list types and 'computation' types.)

Product types. (Cf. Section 3.2.) In this case, for each pair of types σ and σ' there is a binary product diagram in the classifying category $\mathcal{C}\ell(\mathit{Th})$ of the form

$$\begin{tikzcd}[x : \sigma] & {[z : \sigma \times \sigma']} \arrow[l, "{[\mathsf{fst}(z)]}"'] \arrow[r, "{[\mathsf{snd}(z)]}"] & {[x' : \sigma']}\end{tikzcd} .$$

Similarly, in the presence of a one-element type *unit* (cf. Remark 3.2.1), then $[z : \mathit{unit}]$ is a terminal object in $\mathcal{C}\ell(\mathit{Th})$. It follows that in the presence of these type-forming constructs, $\mathcal{C}\ell(\mathit{Th})$ is equivalent to the full subcategory whose objects are represented by contexts of length one.

Disjoint union types. (Cf. Section 3.1.) In this case, for each pair of types σ and σ' and each context $\Gamma = [x_1 : \sigma_1, \dots, x_n : \sigma_n]$, there is a coproduct diagram in the classifying category $\mathcal{C}\ell(\mathit{Th})$ of the form

$$\begin{tikzcd}[\Gamma, x : \sigma] \arrow[r, "{[\vec{x}, \mathsf{inl}_{\sigma'}(x)]}"] & {[\Gamma, z : \sigma + \sigma']} & {[\Gamma, x' : \sigma']} \arrow[l, "{[\vec{x}, \mathsf{inr}_{\sigma}(x')]}"']\end{tikzcd}$$

(where x, x', z are chosen to be distinct from $\vec{x} = x_1, \dots, x_n$). In case $\Gamma = \emptyset$, we have that $[z : \sigma + \sigma']$ is the binary coproduct of $[x : \sigma]$ and $[x : \sigma]$ in $\mathcal{C}\ell(\mathit{Th})$. Given the description of products in the category $\mathcal{C}\ell(\mathit{Th})$ in Proposition 4.2.1, it also follows that product with an arbitrary object $[\Gamma]$ distributes over this coproduct, i.e. it is a stable coproduct(cf. 3.1). Similarly, in the presence of an empty type (cf. Remark 3.1.2), $[z : \mathit{null}]$ is a stable initial object in $\mathcal{C}\ell(\mathit{Th})$.

In the presence of product and one-element types we have seen that every object is isomorphic to one of the form $[x : \sigma]$. In this case the presence of disjoint union and empty types, we can conclude that $\mathcal{C}\ell(\mathit{Th})$ has all stable finite coproducts.

Function types. (Cf. Section 3.3.) In this case, for each pair of types σ and σ', the object $[f : \sigma{\to}\sigma']$ is the exponential of $[x' : \sigma']$ by $[x : \sigma]$ in $\mathcal{C}\ell(\mathit{Th})$, with associated evaluation morphism

$$[f : \sigma{\to}\sigma'] \times [x : \sigma] = [f : \sigma{\to}\sigma', x : \sigma] \xrightarrow{[\mathsf{ap}(f, x)]} [x' : \sigma'] \ .$$

Unlike the case for disjoint union types, it is not necessary to assume the presence of product types in order to conclude that the classifying category possesses exponentials for any pair of objects. For, given any objects $\Gamma = [x_1 : \sigma_1, \dots, x_n : \sigma_n]$ and $\Gamma' = [x'_1 : \sigma'_1, \dots, x'_n : \sigma'_m]$, the exponential $[\Gamma]{\to}[\Gamma']$ in $\mathcal{C}\ell(\mathit{Th})$ is given by the object

$$[f_1 : \vec{\sigma}{\to}\sigma'_1, \dots, f_m : \vec{\sigma}{\to}\sigma'_m]$$

where for each $i = 1, \dots, m$

$$\vec{\sigma}{\to}\sigma'_i \quad \stackrel{\text{def}}{=} \quad \sigma_1{\to}(\sigma_2{\to}\cdots(\sigma_n{\to}\sigma'_i)\cdots) \ .$$

Thus in the presence of function types, the classifying category $\mathcal{C}\ell(\mathit{Th})$ is a cartesian closed category, whether or not we assume the theory *Th* involves product types.

Remark 4.4.1 (The generic model of a theory). Suppose we consider equational theories *Th* in the equational logic of product, disjoint union and function types. We have seen that $\mathcal{C}\ell(\mathit{Th})$ is a cartesian closed category with finite coproducts. (Recall that the stability of coproducts is automatic in the presence of exponentials.) Such a category is sometimes called *bicartesian closed.* We can define a structure G in $\mathcal{C}\ell(\mathit{Th})$ for the underlying signature of Th just as in Section 4.2 and satisfying Lemma 4.2.4. Thus the structure G in $\mathcal{C}\ell(\mathit{Th})$ is a model of Th (i.e. satisfies the axioms of Th) and indeed satisfies exactly the equations that are theorems of Th. An immediate corollary of this property is the *completeness* of the categorical semantics: *A equation is derivable from the axioms of Th using the equational logic of product, disjoint union and function types if (and only if) it is satisfied by all models of Th in bicartesian closed categories.*

Indeed, Theorem 4.2.5 extends to yield a universal property characterizing G as the *generic model* of Th in bicartesian closed categories: given any other model S of Th in a bicartesian closed category $\mathcal{C}$, then the functor $\overline{S} : \mathcal{C}\ell(\mathit{Th}) \longrightarrow \mathcal{C}$ defined in the proof of Theorem 4.2.5 is a morphism of bicartesian closed categories, i.e. preserves finite products, finite coproducts and exponentials. It also maps G to S and is unique up to unique isomorphisms with these properties.

5 Predicate Logic

In previous sections we have considered the categorical interpretation of properties of structures that can be expressed equationally. Now we consider what it means for a structure in a category to satisfy properties expressible in the (first order) predicate calculus, using logical connectives and quantifiers. As in section 3, adjunctions play an important role: we will see that the propositional connectives, the quantifiers, and equality can be characterized in terms of their adjointness properties. First we set up the basic framework of predicate logic and its categorical semantics.

5.1 Formulas and sequents

Let us augment the notion of many-sorted signature considered in section 2 by allowing not only sort and function symbols, but also *relation symbols*, R. We assume that each relation symbol comes equipped with a typing of the form $R \subseteq \sigma_1, \ldots, \sigma_n$, indicating both the number and the sorts of the terms to which R can be applied. To judgements of the form 2.1 asserting that an expression is a well-formed term of a given sort in a given context, we add judgements of the form

$$\phi \; \mathit{prop} \; [\Gamma]$$

asserting that the expression ϕ is a well-formed formula in context Γ. The *atomic* formulas over the given signature are introduced by the rules

$$\frac{M_1 : \sigma_1 \; [\Gamma] \quad \cdots \quad M_n : \sigma_n \; [\Gamma]}{R(M_1, \ldots, M_n) \; \mathit{prop} \; [\Gamma]} \; (\text{where } R \subseteq \sigma_1, \ldots, \sigma_n) \tag{5.1}$$

with one such rule for each relation symbol R. Later we will consider compound formulas built up from the atomic ones using various proposition forming operations. The logical properties of these operations will be specified in terms of *sequents* of the form

$$\Phi \vdash \psi \; [\Gamma] \tag{5.2}$$

where $\Phi = \phi_1, \ldots, \phi_n$ is a finite list of formulas, ψ is a formula and the judgements $\phi_i \, prop \; [\Gamma]$ and $\psi \, prop \; [\Gamma]$ are derivable. The intended meaning of the sequent (5.2) is that the joint validity of all the formulas in Φ logically entails the validity of the formula ψ.

Fig. 2 gives the basic, structural rules for deriving sequents. Since we wish to consider predicate logic as an extension of the rules for equational logic given in section 2.1, Fig. 2 includes a rule (*Subst*) permitting the substitution in a sequent of a term for a provably equal one. The rule (*Weaken*) allows for weakening of contexts. The other possible form of weakening is to add formulas to the left-hand side of a sequent:

$$\frac{\Phi \vdash \psi}{\Phi, \Phi' \vdash \psi}$$

This is derivable from the rules in Fig. 2 because of the form of rule (*Id*). Note that in stating these rules we continue to use the convention established in Remark 3.0.2 that the left-hand part of a context is not shown if it is common to all the judgements in a rule. Thus for example, the full form of rule (*Subst*) is really

$$\frac{M = M' : \sigma \; [\Gamma'] \quad \Phi(x) \vdash \psi(x) \; [\Gamma', x : \sigma, \Gamma]}{\Phi(M) \vdash \psi(M') \; [\Gamma', \Gamma]}$$

Let us enrich the notion of *theory* used in section 2 by permitting the signature of the theory to contain typed relation symbols in addition to sorts and typed function symbols, and by permitting the axioms of the theory to be sequents as well as equations-in-context. The *theorems* of such a theory will now be those judgements that can be derived from the axioms using the rules in Figures 1 and 2 (together with the rules for the various rules for proposition-forming operations to be considered below, as appropriate).

5.2 Hyperdoctrines

If the judgement $\phi \; prop \; [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ over a given signature is derivable, then $\{x_1, \ldots, x_n\}$ contains the variables occurring in the formula ϕ. Intuitively such a formula ϕ describes a subset of the product of the sorts $\sigma_1, \ldots, \sigma_n$. Indeed, suppose we are given a set-valued structure for

$$\frac{\Phi, \psi, \psi \vdash \theta}{\Phi, \psi \vdash \theta}\,(\mathit{Contract}) \qquad \frac{\Phi, \psi, \psi', \Phi' \vdash \theta}{\Phi, \psi', \psi, \Phi' \vdash \theta}\,(\mathit{Exchange})$$

$$\frac{}{\Phi, \psi \vdash \psi}\,(\mathit{Id}) \qquad \frac{\Phi \vdash \psi \quad \Phi', \psi \vdash \psi'}{\Phi, \Phi' \vdash \psi'}\,(\mathit{Cut})$$

$$\frac{\Phi \vdash \psi}{\Phi \vdash \psi\ [\Gamma]}\,(\mathit{Weaken}) \qquad \frac{M = M' : \sigma \quad \Phi(x) \vdash \psi(x)\ [x : \sigma, \Gamma]}{\Phi(M) \vdash \psi(M')\ [\Gamma]}\,(\mathit{Subst})$$

Fig. 2. Structural rules

the signature: such a structure assigns a set $[\![\sigma]\!]$ to each sort σ, a function $[\![F]\!] : [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!] \longrightarrow [\![\tau]\!]$ to each function symbol $F : \sigma_1, \ldots, \sigma_n \longrightarrow \tau$ and a subset $[\![R]\!] \subseteq [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!]$ to each relation symbol $R \subseteq \sigma_1, \ldots, \sigma_n$. Then each formula-in-context $\phi\ \mathit{prop}\ [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ gives rise to the subset of $[\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!]$ consisting of those n-tuples $(a_1, \ldots, a_n)$ for which the sentence $\phi[a_1/x_1, \ldots, a_n/x_n]$ is satisfied in the classical, Tarskian sense (see [Chang and Keisler, 1973, 1.3]). Our aim is to explain how this traditional notion of satisfaction can be generalized to give an interpretation of formulas-in-context in categories other than the category of sets. To do this one needs a categorical treatment of the notion of 'subset' and the various operations performed on subsets when interpreting logical connectives and quantifiers. For this we will use the 'hyperdoctrine' approach originated by Lawvere [1969, 1970], whereby the 'subsets' of an object are given abstractly via additional structure on a category.

Definition 5.2.1. A *prop-category*, $\mathcal{C}$, is a category possessing finite products and equipped with the following structure:

- For each object X in $\mathcal{C}$, a partially ordered set $\mathit{Prop}_{\mathcal{C}}(X)$, whose elements will be called *$\mathcal{C}$-properties of X*. The partial order on $\mathit{Prop}_{\mathcal{C}}(X)$ will be denoted by $\leq$.

- For each morphism $f : Y \longrightarrow X$ in $\mathcal{C}$, an operation assigning to each $\mathcal{C}$-property A of X, a $\mathcal{C}$-property f^*A of Y called the *pullback of A along f*. These operations are required to be

  monotone: if $A \leq A'$, then $f^*A \leq f^*A'$;
  functorial: $\mathit{id}^*A = A$ and $g^*(f^*A) = (f \circ g)^*A$.

Thus a prop-category is nothing other than a category with finite products, $\mathcal{C}$, together with a contravariant functor $Prop_{\mathcal{C}}(-)$ from $\mathcal{C}$ to the category of posets and monotone functions. $Prop_{\mathcal{C}}(-)$ is a particular instance of a Lawvere 'hyperdoctrine', which in general is category- rather than poset-valued (and pseudofunctorial rather than functorial). The reader should be warned that the term 'prop-category' is not standard; indeed there is no standard terminology for the various kinds of hyperdoctrine which have been used in categorical logic.

Here are some important examples of prop-categories from the worlds of set theory, domain theory and recursion theory.

Example 5.2.2 (Subsets). The category $\mathcal{S}et$ of sets and functions supports the structure of a prop-category in an obvious way by taking the $\mathcal{S}et$-properties of a set X to be the ordinary subsets of X. Given a function $f : Y \longrightarrow X$, the operation f^* is that of inverse-image, taking a subset $A \subseteq X$ to the subset $\{y \mid f(y) \in A\} \subseteq Y$.

Example 5.2.3 (Inclusive subsets of cpos). Let $\mathcal{C}po$ denote the category whose objects are posets possessing joins of all ω-chains and whose morphisms are the ω-continuous functions (i.e. monotone functions preserving joins of ω-chains). This category has products, created by the forgetful functor to $\mathcal{S}et$. We make it into a prop-category by taking $Prop_{\mathcal{C}po}(X)$ to consist of all *inclusive* subsets of X, i.e. those that are closed under joins of ω-chains in X. The partial order on $Prop_{\mathcal{C}po}(X)$ is inclusion of subsets. Given $f : Y \longrightarrow X$, since f is ω-continuous, the inverse image of an inclusive subset of X along f is an inclusive subset of Y; this defines the operation $f^* : Prop_{\mathcal{C}po}(X) \longrightarrow Prop_{\mathcal{C}po}(Y)$.

Example 5.2.4 (Realizability). The prop-category $\mathcal{K}\ell$ has as underlying category the category of sets and functions. For each set X, the poset $Prop_{\mathcal{K}\ell}(X)$ is defined as follows. Let $X \rightarrow P(\mathbb{N})$ denote the set of functions from X to the powerset of the set of natural numbers. Let $\leq$ denote the binary relation on this set defined by: $p \leq q$ if and only if there is a partial recursive function $\varphi : \mathbb{N} \rightharpoonup \mathbb{N}$ such that for all $x \in X$ and $n \in \mathbb{N}$, if $n \in p(x)$ then φ is defined at n and $\varphi(n) \in q(x)$. Since partial recursive functions are closed under composition and contain the identity function, $\leq$ is transitive and reflexive, i.e. it is a preorder. Then $Prop_{\mathcal{K}\ell}(X)$ is the quotient of $X \rightarrow P(\mathbb{N})$ by the equivalence relation generated by $\leq$; the partial order between equivalence classes $[p]$ is that induced by $\leq$. Given a function $f : Y \longrightarrow X$, the operation $f^* : Prop_{\mathcal{K}\ell}(X) \longrightarrow Prop_{\mathcal{K}\ell}(Y)$ sends $[p]$ to $[p \circ f]$; it is easily seen to be well-defined, monotonic and functorial.

Example 5.2.5 (Subobjects). A important class of examples of prop-categories is provided by using the category-theoretic notion of a *subobject* of an object. Recall that a morphism $f : Y \longrightarrow X$ in a category $\mathcal{C}$ is a monomorphism if for all parallel pairs of morphisms g, h with codomain

Y, $g = h$ whenever $f \circ g = f \circ h$. The collection of monomorphisms in $\mathcal{C}$ with codomain X can be preordered by declaring that $f \leq f'$ if and only if $f = f' \circ g$ for some (mono)morphism g. Then a subobject of X is an equivalence class of monomorphisms with codomain X under the equivalence relation generated by $\leq$. If for each X, there are only a set of such equivalence classes, $\mathcal{C}$ is said to be *well-powered*.

If $\mathcal{C}$ has all finite limits and is well-powered, we can make it into a prop-category by taking $\mathit{Prop}_{\mathcal{C}}(X)$ to be the set of subobjects of X with partial order that induced by $\leq$. Given $f : Y \longrightarrow X$ in $\mathcal{C}$, the operation f^* on subobjects is that induced by pullback along f. (Recall that a pullback of a monomorphism is again a monomorphism.) When $\mathcal{C} = \mathcal{S}\mathit{et}$, we obtain an example that is isomorphic to example 5.2.2 above. (Subobjects of X in $\mathcal{S}\mathit{et}$ are in natural bijection with subsets of X.) But note that under a similar isomorphism for $\mathcal{C} = \mathcal{C}\mathit{po}$, the structure in example 5.2.3 picks out a restricted collection subobjects. (Not every subobject of X in $\mathcal{C}\mathit{po}$ corresponds to an inclusive subset X—consider for example the subobject determined by identity function from the unordered two element set to the ordered two element set.)

Much of the development of the categorical approach to predicate logic (such as the classic text by Makkai and Reyes [1977]) has restricted its attention to interpretations in this class of examples, where formulas (in context) are interpreted as subobjects in categories.

Given a signature Sg of sort, function and relation symbols as in section 5.1, a *structure* for Sg in a prop-category $\mathcal{C}$ is specified by giving an object $[\![\sigma]\!]$ in $\mathcal{C}$ for each sort σ, a morphism $[\![F]\!] : [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!] \longrightarrow [\![\tau]\!]$ in $\mathcal{C}$ for each function symbol $F : \sigma_1, \ldots, \sigma_n \longrightarrow \tau$, and a $\mathcal{C}$-property $[\![R]\!] \in \mathit{Prop}_{\mathcal{C}}([\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!])$ for each relation symbol $R \subseteq \sigma_1, \ldots, \sigma_n$. For each context $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ and formula ϕ over the signature for which the judgement $\phi\,\mathit{prop}\ [\Gamma]$ is derivable, we will define a $\mathcal{C}$-property

$$[\![\phi[\Gamma]]\!] \in \mathit{Prop}_{\mathcal{C}}([\![\Gamma]\!])$$

where $[\![\Gamma]\!]$ denotes the product $[\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!]$. The definition will be given by induction on the derivation of $\phi\,\mathit{prop}\ [\Gamma]$, since there will be at most one way to derive the well-formedness of a formula in a given context. At the moment we only have one rule for forming formulas, namely rule (5.1) for forming atomic formulas. To define $[\![\phi[\Gamma]]\!]$ in this case we use the semantics of terms-in-context described in Section 2 to assign meanings $[\![M_i[\Gamma]]\!]$ in the structure for the terms M_i and then define

$$[\![R(M_1, \ldots, M_n)[\Gamma]]\!] \overset{\text{def}}{=} \langle [\![M_1[\Gamma]]\!], \ldots [\![M_n[\Gamma]]\!] \rangle^* ([\![R]\!]) \ .$$

The following lemma is a consequence of Lemma 2.2.1 specifying the behaviour of the categorical semantics with respect to substitution of terms for variables in terms.

Lemma 5.2.6 (Semantics of substitution and weakening). *In the categorical semantics of formulas in context, substitution of a term for a variable in a formula is interpreted via the pullback operations. Specifically, if $M_i : \sigma_i$ $[\Gamma]$ for $i = 1, \dots, n$, and if ϕ prop $[\Gamma']$ with $\Gamma' = x_1 : \sigma_1, \dots, \sigma_n$, then*

$$[\![(\phi[\vec{M}/\vec{x}])[\Gamma]]\!] = \langle [\![M_1[\Gamma]]\!], \dots, [\![M_n[\Gamma]]\!] \rangle^*([\![\phi[\Gamma']]\!])$$

where $\phi[\vec{M}/\vec{x}]$ denotes the result of simultaneously substituting M_i for x_i $(i = 1, \dots, n)$ in ϕ.

Similarly, weakening of contexts is interpreted via pullback along a product projection: if ϕ prop $[\Gamma]$, then

$$[\![\phi[\Gamma, x : \sigma]]\!] = \pi_1^*([\![\phi[\Gamma]]\!])$$

where $\pi_1^* : [\![\Gamma, x : \sigma]\!] = [\![\Gamma]\!] \times [\![\sigma]\!] \longrightarrow [\![\Gamma]\!]$ is the product projection in $\mathcal{C}$.

5.3 Satisfaction

Suppose that $\phi_1, \dots, \phi_n \vdash \psi$ $[\Gamma]$ is a sequent over a given signature *Sg*. Given a structure for *Sg* in a prop-category $\mathcal{C}$, what should it mean for the structure to *satisfy* the sequent? Since ϕ_i *prop* $[\Gamma]$ and ψ *prop* $[\Gamma]$ are required to hold, we get $\mathcal{C}$-properties $[\![\phi_i[\Gamma]]\!]$ and $[\![\psi[\Gamma]]\!]$ of the object $[\![\Gamma]\!]$. In the case $n = 1$ when the sequent contains a single antecedent formula, it seems natural to define the satisfaction of the sequent to mean that $[\![\phi_i[\Gamma]]\!] \leq [\![\psi[\Gamma]]\!]$ holds in the poset $Prop_{\mathcal{C}}([\![\Gamma]\!])$. For the general case, let us suppose that each poset $Prop_{\mathcal{C}}(X)$ comes equipped with a distinguished element $\top_X$ and a binary operation $A, A' \mapsto A \wedge A'$. Then we can define $\Phi \vdash \psi$ $[\Gamma]$ to be satisfied if

$$[\![\Phi[\Gamma]]\!] \leq [\![\psi[\Gamma]]\!]$$

where $[\![\Phi[\Gamma]]\!]$ is defined by induction on the length of the list Φ:

$$\begin{aligned} [\![\emptyset[\Gamma]]\!] &\stackrel{\text{def}}{=} \top_X \\ [\![\Phi, \psi[\Gamma]]\!] &\stackrel{\text{def}}{=} [\![\Phi[\Gamma]]\!] \wedge [\![\psi[\Gamma]]\!]. \end{aligned}$$

We require this definition of satisfaction be sound for the rules in Fig. 2: if the hypotheses of a rule are satisfied so should be the conclusion. For (*Cut*) to be sound in the case that $\Phi' = \emptyset$, it is sufficient that for all $A, B, C \in Prop_{\mathcal{C}}(X)$, if $A \leq B$ and $\top_X \wedge B \leq C$ then $A \leq C$. For (*Id*) to be sound, it is sufficient that $A \wedge B \leq B$. From these two properties plus the fact that $\leq$ is a partial order, it follows that $\top_X \wedge A = A$. Using this, the soundness of (*Exchange*) amounts to requiring $A \wedge B = B \wedge A$. But then we have both $A \wedge B \leq B$ and $A \wedge B = B \wedge A \leq A$; so $A \wedge B$ is a lower bound for A and B. In particular for all A, $A = \top_X \wedge A \leq \top_X$, so that $\top_X$ is the greatest element of $\mathit{Prop}_{\mathcal{C}}(X)$. In fact $A \wedge B$ has to be the greatest lower bound of A and B: for the soundness of (*Contract*) requires $C \leq C \wedge C$, for all C; and the general form of (*Cut*) (when Φ' is nonempty) requires for all $A, A', B, C \in \mathit{Prop}_{\mathcal{C}}(X)$, if $A \leq B$ and $A' \wedge B \leq C$ then $A \wedge A' \leq C$. The latter property implies that $\wedge$ is monotone, and hence $C \leq A$ and $C \leq B$ imply $C \leq C \wedge C \leq A \wedge B$. So all in all, we need each poset $\mathit{Prop}_{\mathcal{C}}(X)$ to have finite meets. But that is not all: in view of Lemma 5.2.6, for the soundness of rules (*Weaken*) and (*Subst*) we will need that these finite meets be preserved by the pullback operations f^* in $\mathcal{C}$. Therefore we are led to the following definition.

Definition 5.3.1. A prop-category $\mathcal{C}$ *has finite meets* if for each object X in $\mathcal{C}$ the poset $\mathit{Prop}_{\mathcal{C}}(X)$ possesses all finite meets and these are preserved by the pullback operations f^*. Thus for each X there is $\top_X \in \mathit{Prop}_{\mathcal{C}}(X)$ satisfying

- $A \leq \top_X$, for all $A \in \mathit{Prop}_{\mathcal{C}}(X)$
- $f^*(\top_X) = \top_Y$, for all $f : Y \longrightarrow X$;

and for all $A, B \in \mathit{Prop}_{\mathcal{C}}(X)$, there is an element $A \wedge B \in \mathit{Prop}_{\mathcal{C}}(X)$ satisfying

- for all $C \in \mathit{Prop}_{\mathcal{C}}(X)$, $C \leq A \wedge B$ if and only if $C \leq A$ and $C \leq B$
- $f^*(A \wedge B) = f^*A \wedge f^*B$, for all $f : Y \longrightarrow X$.

Given a structure in $\mathcal{C}$ for a signature *Sg*, we say that a sequent $\Phi \vdash \psi\ [\Gamma]$ over *Sg* is *satisfied* by the structure if

$$\textstyle\bigwedge_{\phi \in \Phi} [\![\phi[\Gamma]]\!] \leq [\![\psi[\Gamma]]\!] \tag{5.3}$$

where the left-hand side indicates the (finite) meet of the elements $[\![\phi[\Gamma]]\!]$.

Extending Theorem 2.2.3 we have:

Theorem 5.3.2 (Soundness). *Let $\mathcal{C}$ be a prop-category that has finite meets and let Th be a theory in the sense of Section 5.1. Then any structure in $\mathcal{C}$ for the underlying signature of Th that satisfies the axioms of Th also satisfies its theorems.*

Examples 5.3.3. The prop-categories of Examples 5.2.2–5.2.5 all have finite meets. Meets in *Set* (respectively *Cpo*) are given by set-theoretic intersection of subsets (respectively inclusive subsets)—these are clearly preserved by the pullback operations since these are given by taking inverse images of subsets along functions (respectively ω continuous functions).

Finite meets in $\mathcal{K}\ell$ can be calculated as follows. Choose some recursive bijection $pr : \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ and define a binary operation on subsets $A, B \subseteq \mathbb{N}$ by

$$A \wedge B \overset{\text{def}}{=} \{pr(n,m) \mid n \in A \text{ and } m \in B\}$$

Then one can show that for any pair of $\mathcal{K}\ell$-properties of a set X represented by functions $p, q : X \longrightarrow P(\mathbb{N})$ say, the meet of $[p]$ and $[q]$ in $\mathit{Prop}_{\mathcal{K}\ell}(X)$ is represented by the function $n \mapsto p(n) \wedge q(n)$. The greatest element $\top_X \in \mathit{Prop}_{\mathcal{K}\ell}(X)$ is represented by the function $n \mapsto \mathbb{N}$. Since the pullback operations f^* in $\mathcal{K}\ell$ are given by pre-composition with f, it is clear from these descriptions that finite meets are preserved by f^*.

Finally, let $\mathcal{C}$ be a category with finite limits, made into a prop-category as in Example 5.2.5 by taking the $\mathcal{C}$-properties of an object X to be its subobjects. Then the top subobject $\top_X$ is represented by the identity on X; clearly this is stable under pullback. Given two subobjects represented by monomorphisms $A \longrightarrow X, B \longrightarrow X$ say, their binary meet is represented by the composition $A \wedge B \longrightarrow A \longrightarrow X$, where $A \wedge B \longrightarrow A$ is the categorical pullback of $B \longrightarrow X$ along $A \longrightarrow X$. Elementary properties of pullbacks in categories guarantee that this meet operation is stable under pullback.

5.4 Propositional connectives

Given the well-known connection between intuitionistic propositional logic and Heyting algebras (see [Dummett, 1977, Chp. 5], for example), it is not surprising that to model these propositional connectives in a prop-category we will need each poset $\mathit{Prop}_{\mathcal{C}}(X)$ to be a Heyting algebra. However, we also require that the Heyting algebra structure be preserved by the pullback operations f^*. This is because these operations will be used in the categorical semantics to interpret the operation of substituting terms for variables in a formula.

Definition 5.4.1. Let $\mathcal{C}$ be a prop-category.

(i) *$\mathcal{C}$ has finite joins* if for each object X in $\mathcal{C}$ the poset $\mathit{Prop}_{\mathcal{C}}(X)$ possesses all finite joins and these are preserved by the pullback operations f^*. Thus for each X there is $\bot_X \in \mathit{Prop}_{\mathcal{C}}(X)$ satisfying

  * $\bot_X \leq A$, for all $A \in \mathit{Prop}_{\mathcal{C}}(X)$
  * $f^*(\bot_X) = \bot_Y$, for all $f : Y \longrightarrow X$;

  and for all $A, B \in \mathit{Prop}_{\mathcal{C}}(X)$, there is an element $A \vee B \in \mathit{Prop}_{\mathcal{C}}(X)$ satisfying

  * for all $C \in \mathit{Prop}_{\mathcal{C}}(X)$, $A \vee B \leq C$ if and only if $A \leq C$ and $B \leq C$
  * $f^*(A \vee B) = f^*A \vee f^*B$, for all $f : Y \longrightarrow X$.

(ii) *$\mathcal{C}$ has Heyting implications* (also called 'relative pseudocomplements') if it has finite meets (cf. Definition 5.3.1) and for each object X and all $A, B \in \mathit{Prop}_{\mathcal{C}}(X)$, there is an element $A \rightarrow B \in \mathit{Prop}_{\mathcal{C}}(X)$ satisfying

  * for all $C \in \mathit{Prop}_{\mathcal{C}}(X)$, $C \leq A \rightarrow B$ if and only if $C \wedge A \leq B$
  * $f^*(A \rightarrow B) = f^*A \rightarrow f^*B$, for all $f : Y \longrightarrow X$.

Given a signature Sg as in Section 5.1, consider compound formulas built up from the atomic ones and the propositional constant *false* using the propositional connectives & (conjunction), $\vee$ (disjunction), and $\Rightarrow$ (implication). The rules of formation are

$$\frac{}{\mathit{false}\,\mathit{prop}\ [\Gamma]} \tag{5.4}$$

$$\frac{\phi\ \mathit{prop}\ [\Gamma] \quad \phi'\ \mathit{prop}\ [\Gamma]}{\phi \# \phi'\ \mathit{prop}\ [\Gamma]}\ (\text{where } \# \text{ is one of } \&, \vee, \Rightarrow) \tag{5.5}$$

Negation, $\neg\phi$, will be treated as an abbreviation of $\phi \Rightarrow \mathit{false}$; truth, *true*, will be treated as an abbreviation of $\neg\mathit{false}$; bi-implication, $\phi \Leftrightarrow \phi'$, will be treated as an abbreviation of $(\phi \Rightarrow \phi')\,\&(\phi' \Rightarrow \phi)$.

Given a structure for Sg in a prop-category $\mathcal{C}$ with finite meets, finite joins and Heyting implications, we can interpret formulas-in-context $\phi[\Gamma]$ as $\mathcal{C}$-properties $[\![\phi[\Gamma]]\!] \in \mathit{Prop}_{\mathcal{C}}([\![\Gamma]\!])$, by induction on the derivation of $\phi\ \mathit{prop}\ [\Gamma]$. Atomic formulas are interpreted as in Section 5.3, and compound formulas as follows:

$$\begin{array}{rcl}
[\![\mathit{false}[\Gamma]]\!] & \stackrel{\text{def}}{=} & \bot_{[\![\Gamma]\!]} \\
[\![\phi\,\&\,\phi'[\Gamma]]\!] & \stackrel{\text{def}}{=} & [\![\phi[\Gamma]]\!] \wedge [\![\phi'[\Gamma]]\!] \\
[\![\phi \vee \phi'[\Gamma]]\!] & \stackrel{\text{def}}{=} & [\![\phi[\Gamma]]\!] \vee [\![\phi'[\Gamma]]\!] \\
[\![\phi \Rightarrow \phi'[\Gamma]]\!] & \stackrel{\text{def}}{=} & [\![\phi[\Gamma]]\!] \rightarrow [\![\phi'[\Gamma]]\!]
\end{array}$$

In this way the notion given in Section 5.3 of satisfaction of a sequent by a structure applies to sequents involving the propositional connectives.

Theorem 5.4.2 (Soundness). *Given a structure in a prop-category $\mathcal{C}$ with finite meets, finite joins and Heyting implications, the collection of sequents $\Phi \vdash \psi\ [\Gamma]$ that are satisfied by the structure (cf. 5.3) is closed under the usual introduction and elimination introduction rules for the propositional connectives in Gentzen's Natural Deduction formulation of the intuitionistic sequent calculus, set out in Fig. 3.*

Recall that a poset $\mathcal{P}$ can be regarded as a category whose objects are the elements of $\mathcal{P}$ and whose morphisms are instances of the order relation. From this point of view, meets, joins, and Heyting implications are all instances of adjoint functors. The operation of taking the meet (respectively the join) of n elements is right (respectively left) adjoint to the diagonal functor $\mathcal{P} \longrightarrow \mathcal{P}^n$; and given $A \in \mathcal{P}$, the Heyting implication operation $A \rightarrow (-) : \mathcal{P} \longrightarrow \mathcal{P}$ is right adjoint to $(-) \wedge A : \mathcal{P} \longrightarrow \mathcal{P}$. Fig. 4 gives an alternative formulation of the rules for the intuitionistic propositional connectives reflecting this adjoint formulation. The rules take the form

$$\frac{\Phi \vdash \mathit{false}}{\Phi \vdash \psi}\,(\mathit{false\text{-}Elim})$$

$$\frac{\Phi \vdash \psi \quad \Phi' \vdash \psi'}{\Phi, \Phi' \vdash \psi \mathbin{\&} \psi'}\,(\&\text{-}\mathit{Intr}) \qquad \frac{\Phi \vdash \psi \mathbin{\&} \psi'}{\Phi \vdash \psi}\,(\&\text{-}\mathit{Elim}_1) \qquad \frac{\Phi \vdash \psi \mathbin{\&} \psi'}{\Phi \vdash \psi'}\,(\&\text{-}\mathit{Elim}_2)$$

$$\frac{\Phi \vdash \psi}{\Phi \vdash \psi \vee \psi'}\,(\vee\text{-}\mathit{Intr}_1) \qquad \frac{\Phi \vdash \psi'}{\Phi \vdash \psi \vee \psi'}\,(\vee\text{-}\mathit{Intr}_2)$$

$$\frac{\Phi, \psi \vdash \theta \quad \Phi', \psi' \vdash \theta \quad \Phi'' \vdash \psi \vee \psi'}{\Phi, \Phi', \Phi'' \vdash \theta}\,(\vee\text{-}\mathit{Elim})$$

$$\frac{\Phi, \psi \vdash \psi'}{\Phi \vdash \psi \Rightarrow \psi'}\,(\Rightarrow\text{-}\mathit{Intr}) \qquad \frac{\Phi \vdash \psi \Rightarrow \psi' \quad \Phi' \vdash \psi}{\Phi, \Phi' \vdash \psi}\,(\Rightarrow\text{-}\mathit{Elim})$$

Fig. 3. Natural Deduction rules for intuitionistic propositional connectives

$$\frac{\Phi \vdash \psi \quad \Phi \vdash \psi'}{\overline{\Phi \vdash \psi \mathbin{\&} \psi'}}\,(\&\text{-}\mathit{Adj}) \qquad \frac{\Phi, \psi \vdash \theta \quad \Phi, \psi' \vdash \theta}{\overline{\Phi, \psi \vee \psi' \vdash \theta}}\,(\vee\text{-}\mathit{Adj})$$

$$\frac{\Phi, \psi \vdash \psi'}{\overline{\Phi \vdash \psi \Rightarrow \psi'}}\,(\Rightarrow\text{-}\mathit{Adj}) \qquad \frac{}{\overline{\Phi, \mathit{false} \vdash \psi}}\,(\mathit{false\text{-}Adj})$$

Fig. 4. 'Adjoint' rules for intuitionistic propositional connectives

$$\frac{\text{sequents}}{\overline{\text{sequent}}}$$

A set of sequents is closed under such a 'bi-rule' if the set contains the sequents above the double line if and only if it contains the sequent below the double line; such a rule is thus an abbreviation for several 'uni-directional' rules.

Note that the presence of the extra formulas Φ in rule $(\vee\text{-}\mathit{Adj})$ means that the character of disjunction is captured not just by the left adjointness property of binary joins—one also needs the adjoint to possess a 'stability' property which in this case is the distribution of finite meets over finite joins. (Compare this with the need identified in Section 3.1 for coproducts to be stable in order to model disjoint union types correctly.) In the presence of rule ($\Rightarrow$-*Adj*) this stability property is automatic, and we could have given the rule without the extra formulas Φ. ((*false-Adj*) is also a 'stable' left adjunction rule, but in this case the stability gives rise to no extra condition.)

The proof of the Soundness Theorem 5.4.2 amounts to showing that the rules in Fig. 3 are derived rules of Fig. 4. Conversely, it is not hard to prove that the (uni-directional versions of) the rules in Fig. 4 are derived rules of Fig. 3. In this sense, the two presentations give equivalent formulations of *provability* (i.e. what can be proved in intuitionistic propositional logic. However, the structure of *proofs* in the two formulations may be very different. One should note that the presence of rule (*Cut*) (which corresponds to the transitivity of $\leq$ in prop-categories) in Fig. 4 is apparently essential for the adjoint formulation to be equivalent to the natural deduction formulation of Fig. 3—unlike the situation for Gentzen's Sequent Calculus formulation where cuts can be eliminated (see [Dummett, 1977, 4.3]).

Remark 5.4.3. A typical feature of the categorical treatment of logical constructs is the identification of which constructs are essentially uniquely determined by their logical properties. Adjoint functors are unique up to isomorphism if they exist. So in particular the operations of Definitions 5.3.1 and 5.4.1 are uniquely determined if they exist for $\mathcal{C}$. Correspondingly, the adjoint rules of Fig. 4 make plain that up to provable equivalence, the intuitionistic connectives are uniquely determined by their logical properties. For example, there cannot be two different binary meet operations on a poset, and correspondingly if $\&'$ were another connective satisfying rule ($\& - Adj$), then $\emptyset \vdash (\phi \,\&\, \phi') \Leftrightarrow (\phi \,\&'\, \phi')$ would be provable for all ϕ, ϕ'. Such uniqueness for logical constructs does not always hold: for example the exponential operator (!) of Girard's Linear Logic [1987] does not have this property.

Example 5.4.4. The prop-category $\mathcal{S}et$ of Example 5.2.2 has finite meets and finite joins, given by set-theoretic intersection and union respectively (which are preserved by the pullback operations since these are given by taking inverse images along functions). Indeed, each $Prop_{\mathcal{S}et}(X)$ is a Boolean algebra and so the categorical semantics of formulas is sound for classical propositional logic. This semantics coincides with the usual set-theoretic one, in the sense that the subset $[\![\phi[x_1 : \sigma_1, \ldots, x_n : \sigma_n]]\!] \subseteq [\![\sigma_1]\!] \times \cdots \times [\![\sigma_n]\!]$ consists of those n-tuples $(a_1, \ldots, a_n)$ for which the sentence $\phi[a_1/x_1, \ldots, a_n/x_n]$ is satisfied in the classical, Tarskian sense (see [Chang and Keisler, 1973, 1.3]).

Example 5.4.5. Turning to the prop-category $\mathcal{C}po$ of Example 5.2.3, first note that each poset $Prop_{\mathcal{C}po}(X)$ does have meets, joins and Heyting implications. Meets (even infinite ones) are given by set-theoretic intersection. Finite joins are given by set-theoretic union. The Heyting implication $A \to B$ of inclusive subsets $A, B \in \mathit{Prop}_{\mathcal{C}po}(X)$ is given by taking the intersection of all inclusive subsets of X that contain $A \setminus B = \{x \in X \mid x \notin A \text{ or } x \in B\}$.

The operation f^* of taking the inverse image of an inclusive subset along an ω-continuous function $f : Y \longrightarrow X$ preserves meets and finite joins, but it does *not* preserve Heyting implications. (For example, take X to be the successor ordinal ω^+, Y to be the discrete ω-cpo with the same set of elements, and f to be the identity function; when $A = \{\omega\}$ and $B = \emptyset$, one has $f^*(A \to B) = f^*(X) = Y \neq \omega = f^*A \to f^*B$.) Thus $\mathcal{C}po$ supports the interpretation of conjunction and disjunction, but not implication.

Example 5.4.6. The prop-category $\mathcal{K}\ell$ of Example 5.2.4 has finite meets, finite joins and Heyting implications. To see this one needs to consider numerical codes for partial recursive functions. Let $n \cdot x$ denote the result, if defined, of applying the nth partial recursive function (in some standard enumeration) to x. The notation $\{n\}(x)$ is traditional for $n \cdot x$. We will often write nx for $n \cdot x$; a multiple application $(n \cdot x) \cdot y$ will be written nxy, using the convention that $- \cdot -$ associates to the left.

So for each partial recursive function $\varphi : \mathbb{N} \rightharpoonup \mathbb{N}$ there is some $n \in \mathbb{N}$ such that $nx \asymp \varphi(x)$ for all $x \in \mathbb{N}$. Here '$e \asymp e$' means '$e\Downarrow$ *if and only if* $e'\Downarrow$, *in which case* $e = e'$', and '$e\Downarrow$' means '*the expression* e *is defined*'. The key requirement of this enumeration of partial recursive functions is that the partial binary operation $n, m \mapsto n \cdot m$ makes $\mathbb{N}$ into a 'partial combinatory algebra'—i.e. there should be $\mathsf{K}, \mathsf{S} \in \mathbb{N}$ satisfying for all $x, y, z \in \mathbb{N}$ that

$$\mathsf{K}x\Downarrow \text{ and } \mathsf{K}x \asymp x$$

$$\mathsf{S}x\Downarrow, \mathsf{S}xy\Downarrow \text{ and } \mathsf{S}xyz \asymp xz(yz)$$

In particular K and S can be used to define $\mathsf{P}, \mathsf{P}_0, \mathsf{P}_1 \in \mathbb{N}$ so that $(x, y) \mapsto \mathsf{P}xy$ defines a recursive bijection $\mathbb{N} \times \mathbb{N} \cong \mathbb{N}$ with inverse $z \mapsto (\mathsf{P}_0 z, \mathsf{P}_1 z)$.

Now define three binary operations on subsets $A, B \subseteq \mathbb{N}$ as follows:

$$\begin{aligned}
A \wedge B &\stackrel{\text{def}}{=} \{\mathsf{P}xy \mid x \in A \text{ and } y \in B\}\\
A \vee B &\stackrel{\text{def}}{=} \{\mathsf{P}_0 x \mid x \in A\} \cup \{\mathsf{P}_1 y \mid y \in A\}\\
A \to B &\stackrel{\text{def}}{=} \{n \mid \text{for all } x \in A,\ nx\Downarrow \text{ and } nx \in B\}
\end{aligned}$$

Using elementary properties of the partial combinatory algebra $(\mathbb{N}, \cdot)$ one can show that for any pair of $\mathcal{K}\ell$-properties of a set X, represented by functions $p, q : X \longrightarrow P(\mathbb{N})$ say, the meet of $[p]$ and $[q]$ in $\mathit{Prop}_{\mathcal{K}\ell}(X)$ is represented by the function $n \mapsto p(n) \wedge q(n)$. Similarly their join is represented by the function $n \mapsto p(n) \vee q(n)$ and their Heyting implication is represented by the function $n \mapsto p(n) \to q(n)$. The greatest and least elements $\top_X, \bot_X \in Prop_{\mathcal{K}\ell}(X)$ are represented by the functions $n \mapsto \mathbb{N}$ and $n \mapsto \emptyset$ respectively. Since the pullback operations f^* in $\mathcal{K}\ell$ are given by pre-composition with f, it is clear from these descriptions that finite meets, finite joins and Heyting implications are all preserved by f^*.

So the prop-category $\mathcal{K}\ell$ supports an interpretation of the propositional connectives that is sound for intuitionistic logic. Combining the definition of the categorical semantics with the above description of finite meets, finite joins and Heyting implications in $\mathcal{K}\ell$ one finds that this interpretation is in fact Kleene's '1945-realizability' explanation of the intuitionistic propositional connectives (see [Dummett, 1977, 6.2]), in the sense that the relation '$n \in [\![\phi[\,]]\!]$' coincides with the relation 'n *realizes* ϕ'.

Remark 5.4.7 (Classical logic). We can obtain classical propositional logic by adding the rule

$$\frac{\Phi \vdash \neg\neg\psi}{\Phi \vdash \psi}$$

to those in Fig. 3. Recalling that $\neg\psi$ is an abbreviation for $\psi \Rightarrow \mathit{false}$, one has that $[\![\neg\psi[\Gamma]]\!]$ is the pseudocomplement of $[\![\psi[\Gamma]]\!]$ in the Heyting algebra $Prop_{\mathcal{C}}([\![\Gamma]\!])$. For the above rule to be sound in $\mathcal{C}$ we need $((A \to \bot_X) \to \bot_X) \leq A$ to hold for each $A \in Prop_{\mathcal{C}}(X)$. In other words the pseudocomplement of every $\mathcal{C}$-property must actually be a complement. So to obtain a sound (and in fact complete) semantics for classical propositional logic we require that each $Prop_{\mathcal{C}}(X)$ is a Boolean algebra and that each pullback operation f^* preserve finite meets and joins (and hence also complements).

5.5 Quantification

The rules for forming quantified formulas are

$$\frac{\phi(x)\ \mathit{prop}\ [x : \sigma]}{\forall_\sigma(\phi)\ \mathit{prop}} \qquad \frac{\phi(x)\ \mathit{prop}\ [x : \sigma]}{\exists_\sigma(\phi)\ \mathit{prop}} .$$

The usual Natural Deduction rules for introducing and eliminating quantifiers are given in Fig. 5. Modulo the structural rules of Fig. 2, these Natural Deduction rules are inter-derivable with an 'adjoint' formulation given by the bi-directional rules of Fig. 6. It should be noted that some side conditions are implicit in these rules because of the well-formedness conditions mentioned after (5.2) that are part of the definition of a sequent. Thus x does not occur free in Φ in ($\forall$-*Intro*) or ($\forall$-*Adj*); and it does not occur free in Φ, Φ', θ in ($\exists$-*Elim*) or ($\exists$-*Adj*).

What structure in a prop-category $\mathcal{C}$ with finite meets is needed to soundly interpret quantifiers? Clearly we need functions

$$\frac{\Phi \vdash \psi(x)\ [x:\sigma]}{\Phi \vdash \forall_\sigma(\psi)}\ (\forall\text{-}\mathit{Intr}) \qquad \frac{\Phi \vdash \forall_\sigma(\psi) \quad M:\sigma}{\Phi \vdash \psi(M)}\ (\forall\text{-}\mathit{Elim})$$

$$\frac{\Phi \vdash \psi(M)}{\Phi \vdash \exists_\sigma(\psi)}\ (\exists\text{-}\mathit{Intr}) \qquad \frac{\Phi \vdash \exists_\sigma(\psi) \quad \Phi', \psi(x) \vdash \theta\ [x:\sigma]}{\Phi, \Phi' \vdash \theta}\ (\exists\text{-Elim})$$

Fig. 5. Natural Deduction rules for quantifiers

$$\frac{\Phi \vdash \psi(x)\ [x:\sigma]}{\overline{\Phi \vdash \forall_\sigma(\psi)}}\ (\forall\text{-}\mathit{Adj})$$

$$\frac{\Phi, \psi(x) \vdash \theta\ [x:\sigma]}{\overline{\Phi, \exists_\sigma(\psi) \vdash \theta}}\ (\exists\text{-}\mathit{Adj})$$

Fig. 6. 'Adjoint' rules for quantifiers

$$\bigwedge_{I,X}, \bigvee_{I,X} \quad : \quad \mathit{Prop}_{\mathcal{C}}(I \times X) \longrightarrow \mathit{Prop}_{\mathcal{C}}(I)$$

so that we can define

$$\begin{aligned} [\![\forall_\sigma(\phi)[\Gamma]]\!] &\stackrel{\text{def}}{=} \bigwedge_{[\![\Gamma]\!],[\![\sigma]\!]}([\![\phi(x)[\Gamma, x:\sigma]]\!]) \\ [\![\exists_\sigma(\phi)[\Gamma]]\!] &\stackrel{\text{def}}{=} \bigvee_{[\![\Gamma]\!],[\![\sigma]\!]}([\![\phi(x)[\Gamma, x:\sigma]]\!]) \ . \end{aligned}$$

In order to retain the soundness of the structural rules (*Weaken*) and (*Subst*), we must require that these operations be natural in I, i.e. for any $f : I' \longrightarrow I$ we require

$$f^*(\bigwedge_{I,X} A) = \bigwedge_{I',X}((f \times id_X)^* A) \tag{5.6}$$

$$f^*(\bigvee_{I,X} A) = \bigvee_{I',X}((f \times id_X)^* A) \ . \tag{5.7}$$

Given the semantics of weakening in Lemma 5.2.6, for the soundness of ($\forall$-*Adj*) we need

$$A \leq \bigwedge_{I,X}(B) \quad \Leftrightarrow \quad \pi_1^*(A) \leq B \tag{5.8}$$

for all $A \in \mathit{Prop}_{\mathcal{C}}(I)$ and $B \in \mathit{Prop}_{\mathcal{C}}(I \times X)$. In other words $\bigwedge_{I,X}$ provides a right adjoint to the monotone function

$$\pi_1^* : \mathit{Prop}_{\mathcal{C}}(I) \longrightarrow \mathit{Prop}_{\mathcal{C}}(I \times X) \ .$$

Note that this requirement determines the function $\bigwedge_{I,X}$ uniquely and implies in particular that it is a monotone function.

Given the definition of satisfaction of sequents in Section 5.3, for the soundness of ($\exists$-*Adj*) we need

$$A \wedge \bigvee_{I,X}(B) \leq C \quad \Leftrightarrow \quad \pi_1^*(A) \wedge B \leq \pi_1^*(C)$$

for all $A, C \in \mathit{Prop}_{\mathcal{C}}(I)$ and $B \in \mathit{Prop}_{\mathcal{C}}(I \times X)$. We can split this requirement into two: first that $\bigvee_{I,X}$ provides a left adjoint to π_1^*, and secondly a 'stability' condition for the left adjoint with respect to meets:

$$\begin{aligned} \bigvee_{I,X}(B) \leq C \quad &\Leftrightarrow \quad B \leq \pi_1^*(C) && (5.9) \\ A \wedge \bigvee_{I,X}(B) \quad &= \quad \bigvee_{I,X}(\pi_1^*(A) \wedge B) \ . && (5.10) \end{aligned}$$

Definition 5.5.1. Let $\mathcal{C}$ be a prop-category with finite meets. We say that *$\mathcal{C}$ has universal quantification* if for each product projection $\pi_1 : I \times X \longrightarrow X$ in $\mathcal{C}$, the pullback function π_1^* possesses a right adjoint $\bigwedge_{I,X}$ (5.6) and these right adjoints are natural in I (5.6).

We say that *$\mathcal{C}$ has existential quantification* if each π_1^* possesses a left adjoint $\bigvee_{I,X}$ (5.9) and these left adjoints are both natural in I (5.7) and satisfy the stability condition (5.10).

Thus we have shown that *if $\mathcal{C}$ is a prop-category with finite meets and both universal and existential quantification, then the sequents that are satisfied by a structure in $\mathcal{C}$ are closed under the quantifier rules in Fig. 5.*

Remark 5.5.2. Conditions (5.6) and (5.7) are instances of what Lawvere [1969] termed the 'Beck-Chevalley Condition': see Remark 5.6.6. Condition (5.10) was called by him 'Frobenius Reciprocity'. It is somewhat analogous to the stability condition on coproducts needed in Section 3.1 to soundly interpret disjoint union types. Just as there we noted that the existence of exponentials makes stability of coproducts automatic, so here it is the case that condition (5.10) holds automatically if $\mathcal{C}$ has Heyting implications (see Definition 5.4.1).

Example 5.5.3. The prop-cat $\mathcal{S}et$ of Example 5.2.2 possesses adjoints for all pullback operations and in particular for pullback along projections. Specifically, for $A \subseteq I \times X$ we have

$$\begin{aligned} \bigwedge_{I,X}(A) \quad &= \quad \{i \in I \mid \forall x \in X.(i,x) \in A\} \\ \bigvee_{I,X}(A) \quad &= \quad \{i \in I \mid \exists x \in X.(i,x) \in A\} \ . \end{aligned}$$

These adjoints are easily seen to be natural in I, and since $\mathcal{S}et$ has Heyting (indeed Boolean) implications, as noted above the Frobenius Reciprocity condition (5.10) holds automatically.

Thus $\mathcal{S}et$ has both universal and existential quantification. Given the above formulas for the adjoints, it is clear that the remark in Example 5.4.4 about the coincidence categorical semantics in $\mathcal{S}et$ with the usual, Tarskian semantics of formulas continues to hold in the presence of quantified formulas.

Example 5.5.4. The prop-category $\mathcal{K}\ell$ of Example 5.2.2 also has universal and existential quantification. Given $A \in \mathit{Prop}_{\mathcal{K}\ell}(I \times X)$, suppose A is represented by the function $p : I \times X \longrightarrow P(\mathbb{N})$. Then $\bigwedge_{I,X}(A)$ and $\bigvee_{I,X}(A)$ can be represented respectively by the functions

$$\alpha_{I,X}(p)\ ,\ \epsilon_{I,X}(p) : I \longrightarrow P(\mathbb{N})$$

defined by

$$\alpha_{I,X}(p)(i) = \bigcap_{x \in X} p(i,x) \quad \text{and} \quad \epsilon_{I,X}(p)(i) = \bigcup_{x \in X} p(i,x)\ .$$

Some calculations with partial recursive functions show that these formulas do indeed yield the required adjoints, and that they are natural in I. Since we noted in Example 5.4.6 that $\mathcal{K}\ell$ has Heyting implications, the Frobenius Reciprocity condition (5.10) holds automatically.

Example 5.5.5. The prop-category $\mathcal{C}po$ of Example 5.2.3 possesses natural right adjoints to pulling back along projections: they are given just as for $\mathcal{S}et$ by the formula (5.11), since this is an inclusive subset when A is. $\mathcal{C}po$ also possesses left adjoint to pulling back along projections: $\bigvee_{I,X}(A)$ is given by the smallest inclusive subset of I containing $\{i \in I \mid \exists x \in X.(i,x) \in A\}$ (i.e. by the intersection of all inclusive subsets containing that set). However, these left adjoints do not satisfy the Beck-Chevalley condition (5.7). For if this condition did hold, for any $i \in I$ we could apply it with I' a one-element ω-cpo and $f : I' \longrightarrow I$ the function mapping the unique element of I' to i, to conclude that $i \in \bigvee_{I,X}(A)$ if and only if $(i,x) \in A$ for some $x \in X$. In other words the set in (5.11) would already be inclusive when A is inclusive. But this is by no means the case in general. (For example consider when I is the successor ordinal ω^+ and X is the discrete ω-cpo with the same set of elements. Then $A = \{(m,n) \mid m \leq n\}$ is an inclusive subset of $I \times X$, but (5.11) is ω, which is not an inclusive subset of ω^+.)

Thus the prop-category $\mathcal{C}po$ has universal quantification, but not existential quantification.

5.6 Equality

We have seen that the categorical semantics of the propositional connectives and quantifiers provides a characterization of these logical operations in terms of various categorical adjunctions. In this section we show that,

$$\frac{M : \sigma}{\Phi \vdash (M =_\sigma M)}\ (=\text{-}\mathit{Intr})$$

$$\frac{\Phi \vdash (M =_\sigma M') \quad \Phi' \vdash \psi(x,x)\ [x : \sigma]}{\Phi, \Phi' \vdash \psi(M, M')}\ (=\text{-}\mathit{Elim})$$

Fig. 7. Natural Deduction rules for equality

$$\frac{\Phi \vdash \psi(x,x)\ [x : \sigma]}{\Phi, (x =_\sigma x') \vdash \psi(x,x')\ [x : \sigma, x' : \sigma]}\ (=\text{-}\mathit{Adj})$$

Fig. 8. 'Adjoint' rule for equality

within the context of first-order logic, the same is true of equality predicates. (As with so much of categorical logic, this observation originated with Lawvere [1970].) The formation rule for equality formulas is

$$\frac{M : \sigma \quad M' : \sigma}{(M =_\sigma M')\ \mathit{prop}}\ .$$

Natural Deduction rules for introducing and eliminating such formulas are given in Fig. 7 (cf. [Nordström *et al.*, 1990, Section 8.1]). The usual properties of equality, such as reflexivity, symmetry, and transitivity, can be derived from these rules. It is not hard to see that modulo the structural rules of Fig. 2, the Natural Deduction rules are inter-derivable with an 'adjoint' formulation given by the bi-directional rule of Fig. 8.

Suppose $\mathcal{C}$ is a prop-category with finite meets. To interpret equality formulas whilst respecting the structural rules (*Weaken*) and (*Subst*) of Fig. 2, for each $\mathcal{C}$-object X we need a $\mathcal{C}$-property

$$\mathit{Eq}_X \in \mathit{Prop}_{\mathcal{C}}(X \times X)$$

so that we can define

$$[\![(M =_\sigma M')[\Gamma]]\!] \stackrel{\text{def}}{=} \langle [\![M[\Gamma]]\!], [\![M'[\Gamma]]\!] \rangle^*(\mathit{Eq}_{[\![\sigma]\!]})\ . \tag{5.11}$$

Given the definition of satisfaction of sequents in Section 5.3, for the soundness of $(=\text{-}\mathit{Adj})$ we need

$$A \leq (\mathit{id}_I \times \Delta_X)^*(B) \quad \Leftrightarrow \quad (\mathit{id}_I \times \pi_1)^*(A) \wedge \pi_2^*(\mathit{Eq}_X) \leq B \tag{5.12}$$

for all $\mathcal{C}$-objects I, X and all $\mathcal{C}$-properties $A \in \mathit{Prop}_{\mathcal{C}}(I \times X)$ and $B \in \mathit{Prop}_{\mathcal{C}}(I \times (X \times X))$. Here Δ_X is the diagonal morphism $\langle id_X, id_X \rangle : X \longrightarrow X \times X$, and π_1 and π_2 are the product projections $X \times X \longrightarrow X$ and $I \times (X \times X) \longrightarrow X \times X$.

Condition (5.12) says that for each $A \in \mathit{Prop}_{\mathcal{C}}(I \times X)$, the value of the left adjoint to $(id_I \times \Delta_X)^*$ exists at A and is equal to $(id_I \times \pi_1)^*(A) \wedge \pi_2^*(Eq_X)$. In particular, taking I to be the terminal object in $\mathcal{C}$ and $A = \top_{I \times X}$, we have that Eq_X is the value of the left adjoint to Δ_X^* at the top element $\top_X \in \mathit{Prop}_{\mathcal{C}}(X)$, i.e.

$$\top_X \leq \Delta_X^*(B) \quad \Leftrightarrow \quad Eq_X \leq B \tag{5.13}$$

for all $B \in \mathit{Prop}_{\mathcal{C}}(X \times X)$. Note that the $\mathcal{C}$-properties Eq_X are uniquely determined by this requirement.

Definition 5.6.1. Let $\mathcal{C}$ be a prop-category with finite meets. We say that $\mathcal{C}$ *has equality* if for each $\mathcal{C}$ object X, the value Eq_X of the left adjoint to Δ_X^* at the top element $\top_X \in \mathit{Prop}_{\mathcal{C}}(X)$ exists and satisfies (5.12)

Thus we have shown that *if $\mathcal{C}$ is a prop-category with finite meets and equality, then the sequents that are satisfied by a structure in $\mathcal{C}$ are closed under the equality rules in Fig. 7.*

Remark 5.6.2. In fact when $\mathcal{C}$ also has Heyting implications and universal quantification, then property (5.12) is automatic if Eq_X exists satisfying (5.13). This observation corresponds to the fact that in the presence of implication and universal quantification, the rule (=-*Adj*) is inter-derivable with a simpler rule without 'parameters':

$$\frac{\emptyset \vdash \psi(x,x)\ [x : \sigma]}{(x =_\sigma x') \vdash \psi(x,x')\ [x : \sigma, x' : \sigma]} \ .$$

Example 5.6.3. The prop-category $\mathcal{S}et$ of Example 5.2.2 has equality. Since we have seen in previous sections that $\mathcal{S}et$ has implication and universal quantification, by Remark 5.6.2 it suffices to establish the existence of the left adjoint to Δ_X^* at $\top_X$. But for this, clearly we can take $Eq_X \subseteq X \times X$ to be the identity relation $\{(x,x) \mid x \in X\}$.

Similarly for the prop-category $\mathcal{K}\ell$ of Example 5.2.4, since it has implication and universal quantification, to see that it also has equality we just have to verify (5.13). This can be done by taking taking Eq_X to be the $\mathcal{K}\ell$-property represented by the function $\delta_X : X \times X \longrightarrow P(\mathbb{N})$ defined by

$$\delta_X(x,x') \ \stackrel{\text{def}}{=}\ \begin{cases} \mathbb{N} & \text{if } x = x' \\ \emptyset & \text{if } x \neq x' \ . \end{cases}$$

Example 5.6.4. We saw in Example 5.4.5 that prop-category $\mathcal{C}po$ does not have Heyting implications. Hence condition (5.12) is not necessarily implied by the special case (5.13) in this case. Nevertheless, we can satisfy (5.12) by taking each Eq_X to be the inclusive subset $\{(x,x) \mid x \in X\}$ of the ω-cpo $X \times X$. Thus $\mathcal{C}po$ does have equality.

Example 5.6.5. Let $\mathcal{C}$ be a category with finite limits, made into a prop-category (with finite meets) as in Example 5.2.5. Then $\mathcal{C}$ has equality, with Eq_X the subobject of $X \times X$ represented by the diagonal monomorphism $\Delta_X = \langle id_X, id_X\rangle$. Note that in this case, definition (5.11) implies that the interpretation $[\![(M =_\sigma M')[\Gamma]]\!]$ of an equality formula is the subobject of $[\![\Gamma]\!]$ represented by the equalizer of the pair of $\mathcal{C}$-morphisms

$$[\![M[\Gamma]]\!]\ ,\ [\![M'[\Gamma]]\!] : [\![\Gamma]\!] \longrightarrow [\![\sigma]\!]\ .$$

Remark 5.6.6 (Generalized quantifiers). Suppose that $\mathcal{C}$ is a prop-category with finite meets, Heyting implications and both universal and existential quantification. Then not only do the adjoints $\bigwedge_{I,X}, \bigvee_{I,X}$ exist, but in fact for any $\mathcal{C}$ morphism $f : X \longrightarrow Y$, the monotone function $f^* : Prop_{\mathcal{C}}(Y) \longrightarrow Prop_{\mathcal{C}}(X)$ has both left and right adjoints, which will be denoted $\bigvee_f$ and $\bigwedge_f$ respectively. Indeed, for $B \in Prop_{\mathcal{C}}(Y)$ we can define

$$\begin{array}{rcl} \bigvee_f(A) & = & \bigvee_{X,Y}((f \times id_Y)^*(Eq_Y) \wedge \pi_1^*(A)) \\ \bigwedge_f(A) & = & \bigwedge_{X,Y}((f \times id_Y)^*(Eq_Y) \rightarrow \pi_1^*(A))\ . \end{array}$$

It is easier to see what these expressions mean and to prove that they have the required adjointness properties if we work in a suitable *internal language* for the prop-category $\mathcal{C}$. The signature of such an internal language is like that discussed in Section 2.3, but augmented with relation symbols $R \subseteq X_1 \times \cdots \times X_n$ for each $\mathcal{C}$-property $R \in Prop_{\mathcal{C}}(X_1 \times \cdots \times X_n)$ (for each tuple of $\mathcal{C}$-objects $X_1, \ldots, X_n$). Using the obvious structure in $\mathcal{C}$ for this signature, one can describe $\mathcal{C}$-properties using the interpretation of formulas over the signature; and relations between such $\mathcal{C}$-properties can be established by proving sequents in the predicate calculus and then appealing to the soundness results we have established.

From this perspective $\bigvee_f$ and $\bigwedge_f$ are the interpretation of formulas that are *generalized quantifiers, adjoint to substitution*:

$$\begin{array}{rcl} \bigvee_f(A) & = & [\![\exists_\sigma(x)(f(x) =_Y y \ \&\ A(x))]\!] \\ \bigwedge_f(A) & = & [\![\forall_\sigma(x)(f(x) =_Y y \Rightarrow A(x))]\!]\ . \end{array}$$

The adjointness properties

$$\bigvee_f(A) \leq B \quad \Leftrightarrow \quad A \leq f^*(B)$$

$$B \leq \bigwedge_f(A) \quad \Leftrightarrow \quad f^*(B) \leq A$$

can be deduced from the fact that the following bi-directional rules are derivable from the Natural Deduction rules for $\&, \Rightarrow, \exists, \forall, =$:

$$\frac{\exists_\sigma(x)(f(x) =_Y y \;\&\; A(x)) \vdash B(y) \; [y : Y]}{A(x) \vdash B(f(x)) \; [x : X]}$$

$$\frac{B(y) \vdash \forall_\sigma(x)(f(x) =_Y y \Rightarrow A(x)) \; [y : Y]}{B(f(x)) \vdash A(x) \; [x : X]} \quad .$$

We required the adjoints $\bigvee_{I,X}(= \bigvee_{\pi_1})$ and $\bigwedge_{I,X}(= \bigwedge_{\pi_1})$ to be natural in I in order to model the interaction between quantification and substitution correctly. More generally, the adjoints $\bigvee_f$ and $\bigwedge_f$ enjoy stability properties reflecting the interaction of the generalized quantifiers with substitution on their parameters. These stability conditions are all instances of what Lawvere [1969] termed Beck-Chevalley conditions. In general if

$$\begin{tikzcd} X \arrow[r, "h"] \arrow[d, "f"'] & W \arrow[d, "k"] \\ Y \arrow[r, "g"'] & Z \end{tikzcd}$$

is a commutative square in $\mathcal{C}$, then we say that the left adjoints to the pullback functions satisfy a *Beck-Chevalley Condition* for the above square if $g^*\bigvee_k = \bigvee_f h^*$. Note that $h^*k^* = f^*g^*$ (since $kh = gf$), $id_W \leq k^*\bigvee_k$ (since $\bigvee_k$ is left adjoint to k^*), and $\bigvee_f f^* \leq id_Y$ (since $\bigvee_f$ is left adjoint to f^*); so

$$\bigvee_f h^* \leq \bigvee_f h^* k^* \bigvee_k = \bigvee_f f^* g^* \bigvee_k \leq g^* \bigvee_k \ .$$

So the Beck-Chevalley Condition amounts to requiring the other inequality: $g^*\bigvee_k \leq \bigvee_f h^*$. Dually, $g^*\bigwedge_k \leq \bigwedge_f h^*$ holds automatically and the right adjoints are said to satisfy a Beck-Chevalley condition for the above square if the reverse inequality holds, so that $g^*\bigwedge_k = \bigwedge_f h^*$. With this terminology, it is the case that in a prop-category with finite meets, Heyting implications and quantification, the left and right adjoints to the pullback operations satisfy the Beck-Chevalley Condition for certain commutative squares which (by virtue of the finite products in $\mathcal{C}$) are pullback squares. These are the squares of the form

$$\begin{tikzcd}
I \times X \arrow[r, "\pi_1"] \arrow[d, "id_I \times f"'] & X \arrow[d, "f"] & X \arrow[r, "\langle id_X, f\rangle"] \arrow[d, "f"'] & X \times Y \arrow[d, "f \times id_Y"] \\
I \times Y \arrow[r, "\pi_1"'] & Y & Y \arrow[r, "\Delta_Y"'] & Y \times Y
\end{tikzcd} .$$

If the underlying category of $\mathcal{C}$ has all pullbacks, obviously a sufficient condition to ensure the adjoints to the pullback functions satisfy these stability conditions is that the Beck-Chevalley Condition holds for all pullback squares. (And in fact this holds for the left adjoints if and only if it holds for the right adjoints.) This is the case for the prop-categories $\mathcal{S}et$ and $\mathcal{K}\ell$ of Examples 5.2.2 and 5.2.4.

Remark 5.6.7. The version of predicate logic we have described here is somewhat unusual in that it has both equality judgements ($M = M' : \sigma\ [\Gamma]$) and equality formulas ($M =_\sigma M'$), the latter occurring as a constituent of sequent judgments ($\Phi \vdash \psi\ [\Gamma]$). Using the rule (*Subst*) from Fig. 2 and the rule (=-*Intro*) from Fig. 7, one can derive the following rule connecting the two forms of equality:

$$\frac{M = M' : \sigma}{\emptyset \vdash (M =_\sigma M')} .$$

However, for the converse of this rule to be satisfied in a prop-category with equality it would have to be the case that for any parallel pair of $\mathcal{C}$-morphisms $f, f' : I \longrightarrow X$, that if $\top_I = \langle f, f'\rangle^*(Eq_X)$ then $f = f'$. This condition holds for the examples given in this section, but not in general. (For example, consider a 'degenerate' prop-category whose underlying category is the category of sets and functions and whose poset of properties at each set is the trivial, one-element poset.)

5.7 Completeness

Let *Th* be a theory whose sequent axioms may involve the propositional connectives, quantifiers and equality. The theorems of such a theory are those judgements that can be derived from the axioms using the rules in Figs 2, 3, 5, and 7, together with the rules for equational logic given in Section 2.1.

Given a prop-category $\mathcal{C}$ possessing finite meets, finite joins, Heyting implications, quantification, and equality, a categorical model for *Th* in $\mathcal{C}$ consists of a structure for the signature of the theory with the property that each of the theory's axioms are satisfied by the structure. The soundness of the categorical semantics implies that every theorem of the theory is also satisfied by such a model. Conversely, the categorical semantics is *complete*, in the sense that a judgement over the signature of a given theory is a theorem if it is satisfied by all models of the theory in prop-categories with finite meets, finite joins, Heyting implications, quantification, and equality.

This completeness is an easy corollary of the stronger result that, given Th, there is a 'classifying' prop-category $\mathcal{C}\ell(Th)$ containing a 'generic' model G, which is in particular a structure that satisfies a sequent (or an equation) if and only if it is a theorem of Th. This classifying prop-category can be constructed via an extension of the 'term-model' construction discussed in Section 4. The underlying category of the prop-category $\mathcal{C}\ell(Th)$ is constructed just as in Section 4.2: its objects are α-equivalence classes of contexts Γ, and its morphisms are equivalence classes of context morphisms under provable equality in Th.

Then for each object Γ we defined the poset of $\mathcal{C}\ell(Th)$-properties of Γ as follows. Its underlying set is the quotient

$$Prop_{\mathcal{C}\ell(Th)}(\Gamma) \quad \stackrel{\text{def}}{=} \quad \{\phi \mid \phi\, prop\ [\Gamma]\}/\sim_{Th}$$

where the equivalence relation $\phi \sim_{Th} \phi'$ holds if and only if both $\phi \vdash \phi'\ [\Gamma]$ and $\phi' \vdash \phi\ [\Gamma]$ are theorems of Th. The partial order on $Prop_{\mathcal{C}\ell(Th)}(\Gamma)$ is that induced by Th-provable entailment: $A \leq A'$ holds if and only if for some (indeed, any) formulas ϕ and ϕ' representing the equivalence classes A and A' respectively, $\phi \vdash \phi'\ [\Gamma]$ is a theorem of Th.

To complete the definition of the prop-category structure (cf. Definition 5.2.1), we have to define the action of pulling back a $\mathcal{C}\ell(Th)$-property along a morphism. This is induced by the operation of substituting terms for variables in formulas. Given $A \in Prop_{\mathcal{C}\ell(Th)}(\Gamma)$ and $\gamma : \Gamma' \longrightarrow \Gamma$ with

$$\begin{array}{rcl} A & = & [\phi] \\ \Gamma & = & [x_1 : \sigma_1, \ldots, x_n : \sigma_n] \\ \gamma & = & [M_1, \ldots, M_n] \end{array}$$

then

$$\gamma^* A \quad \stackrel{\text{def}}{=} \quad [\phi[M_1/x_1, \ldots, M_n/x_n]] \ .$$

The properties of monotonicity and functoriality of $(-)^*$ in Definition 5.2.1 are easily verified (using the definition of identities and composition in $\mathcal{C}\ell(Th)$ given in Section 4.2.)

Proposition 5.7.1. *The classifying prop-category $\mathcal{C}\ell(Th)$ has finite meets, finite joins, Heyting implication, universal and existential quantification, and equality.*

Proof. The propositional operations are induced by the corresponding logical connectives. Thus for any object Γ and $\mathcal{C}\ell(Th)$-properties $A = [\phi]$ and $A' = [\phi']$ of Γ, we have:

$$\begin{array}{rcl} A \wedge A' & = & [\phi \,\&\, \phi'] \\ A \vee A' & = & [\phi \vee \phi'] \\ A \to A' & = & [\phi \Rightarrow \phi'] \\ \top_\Gamma & = & [\mathit{true}] \\ \bot_\Gamma & = & [\mathit{false}] \ . \end{array}$$

Similarly, quantification is induced by the corresponding operation on formulas. Thus if $A = [\phi] \in \mathit{Prop}_{\mathcal{C}\ell(\mathit{Th})}(\Delta \times \Gamma)$, so that $\phi\, \mathit{prop}\, [\Delta, \Gamma]$, then

$$\bigwedge\nolimits_{\Delta,\Gamma}(A) \quad = \quad [\forall_{\sigma_1}(x_1) \cdots \forall_{\sigma_n}(x_n)\phi]$$

if $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$ say.

Finally, for each object $\Gamma = [x_1 : \sigma_1, \ldots, x_n : \sigma_n]$, the equality property $\mathit{Eq}_\Gamma \in \mathit{Prop}(\Gamma \times \Gamma')$ can be represented by the formula

$$(x_1 =_{\sigma_1} x_1') \,\&\, \cdots \,\&(x_n =_{\sigma_n} x_n')$$

where the variables $x_1', \ldots, x_n'$ are chosen to be distinct from $x_1, \ldots, x_n$ (and each other). ∎

Turning now to the *generic model* G of Th in $\mathcal{C}\ell(\mathit{Th})$, its underlying structure is defined for the types and function symbols of Th as in Section 4.2, and for relation symbols $R \subseteq \sigma_1, \ldots, \sigma_n$ by

$$G[\![R]\!] \overset{\text{def}}{=} [R(x_1, \ldots, x_n)] \in \mathit{Prop}_{\mathcal{C}\ell(\mathit{Th})}([x_1 : \sigma_1, \ldots, x_n : \sigma_n]) \ .$$

Extending the properties of G on types, terms and equations given in Lemma 4.2.4, we also have the following properties for formulas and sequents.

Lemma 5.7.2. *The structure G has the following properties.*

(i) If ϕ prop $[\Gamma]$, then the interpretation $G[\![\phi[\Gamma]]\!] \in \mathit{Prop}_{\mathcal{C}\ell(\mathit{Th})}(\Gamma)$ of the formula ϕ in the structure G is just the $\mathcal{C}\ell(\mathit{Th})$-property represented by the formula ϕ.

(ii) $\Phi \vdash \psi\ [\Gamma]$ is a theorem of Th if and only if

$$\bigwedge\nolimits_{\phi \in \Phi} G[\![\phi[\Gamma]]\!] \leq G[\![\psi[\Gamma]]\!] \ .$$

In other words, G satisfies a sequent just in case it is a theorem of Th. In particular G satisfies the axioms of Th and hence is a model of Th.

Corollary 5.7.3 (Completeness). *A judgement over the signature of a given theory is a theorem if it is satisfied by all models of the theory in* *prop-categories with finite meets, finite joins, Heyting implications, quantification, and equality.*

Stronger forms of completeness hold with models of *Th* restricted to particular classes of prop-category. For example, Kripke's completeness theorem for intuitionistic predicate logic (see [Dummett, 1977, Chp. 5]) can be paraphrased in category-theoretic terms as asserting the completeness of the collection of prop-categories of the kind given in Example 5.2.5, but with the category $\mathcal{C}$ restricted to be a category of set-valued presheaves on a poset.

Remark 5.7.4. The classifying prop-category $\mathcal{C}\ell(Th)$ and the generic algebra G of a predicate theory have a universal property analogous to that given for algebraic theories in Theorem 4.2.5 (but with respect to a suitable notion of morphism of prop-categories). This universal property forms the basis of an equivalence between theories in predicate logic (and translations between them) and suitably structured prop-categories (and morphisms between them)—cf. Section 4.3.

6 Dependent Types

This section considers the categorical semantics of type theories involving types that depend upon terms. Martin-Löf's type theory (see [Nordström *et al.*, 1990]) provides an example of such a system, as does the Girard-Reynolds second order lambda calculus (see [Girard, 1989]). Here we will just consider a fundamental example, namely the dependently typed analogue of the algebraic theories of Section 2. So types and terms will be built up from a signature of function symbols which now may be type-valued as well as term-valued. Once one has explained the categorical framework corresponding to such *dependently typed algebraic theories* [1], one can use it systematically to describe the categorical structure needed for any particular dependently-typed datatype constructor—much as we did for simple types in Section 3. We will just consider a single example of this, namely the categorical semantics of dependent products in Section 6.5.

Several different, but interconnected, categorical structures have been proposed for interpreting the basic framework of dependent types by Seely [1984], Cartmell [1986], Taylor [1986], Ehrhard [1988], Streicher [1989, 1991], Hyland and Pitts [1989], Obtułowicz [1989], Curien [1989], and Jacobs [1991]. This reflects the fact that the categorical interpretation of dependent types is undoubtedly more complicated than the other varieties of categorical logic explained in this chapter. This is due to the structural complications implicit in the logic. One complication is that in general one must consider the satisfaction of equations not only between terms (of the same type), but also between types. Another complication is that

[1] Cartmell [1986] calls such theories 'generalized algebraic'.

proofs of well-formedness of expressions cannot be separated from proofs of equality, and are by no means unique. If one assigns meanings to expressions by induction on the derivation of their well-formedness, one therefore has to take care that the meaning is independent of the particular derivation. Early accounts [Seely, 1984] of the categorical semantics of dependent types tended to gloss over this point. Here we will take a more careful, but necessarily more complicated approach using an adaptation of Cartmell's notion of 'category with attributes'—a simplified version of the 'contextual categories' he uses in [Cartmell, 1986]. Categories with attributes are categories equipped with some extra structure for interpreting (dependent) types: accordingly we adopt (in Definition 6.3.3) the rather more compact terminology of 'type-categories' for the categorical structure we use.

In order to motivate the notion of type-category, we will reverse the pattern set by Sects 2 and 4: having described the syntax, we will organize it into a category just as in Section 4.2, and see how the extra structure needed for a type-category emerges. We then give the interpretation of the syntax in a general type-category. The soundness of the interpretation (Theorem 6.4.5) relies on some 'strictness' conditions built into the definition of type-category to avoid the potential difficulties mentioned above to do with multiple proofs of well-formedness of judgements. The wide range of examples of type-categories which we give shows that these strictness conditions are not restrictive in practice. Without them one must resolve quite complicated coherence problems: see [Curien, 1990].

6.1 Syntactic considerations

We will continue to make use of the simply typed metalanguage introduced at the beginning of Section 3, with ground arities $\textsc{Types}$ and $\textsc{Terms}$. We assume given a *signature*, *Sg*, consisting of collections of meta-constants

- $s : \textsc{Terms}^n \twoheadrightarrow \textsc{Types}$, called n-ary *type-valued function symbols*
- $F : \textsc{Terms}^n \twoheadrightarrow \textsc{Terms}$, called n-ary *term-valued function symbols*

for each number n. The raw[2] expressions of the object language over *Sg* are then the meta-expressions built up from the symbols in *Sg* and a fixed, countably infinite set $Var = \{v_1, v_2, \ldots\}$ of meta-variables of arity $\textsc{Terms}$. In particular, the raw types over *Sg* are meta-expressions of arity $\textsc{Types}$ and the raw terms over *Sg* are meta-expressions of arity $\textsc{Terms}$. (The algebraic signatures of Sect 2.1 also contained typing information for the function symbols. The analogous information here will given as part of the axioms of a dependently-typed algebraic theory, rather than as part of its underlying signature.) Throughout Section 6, *Sg* will denote a fixed signature in the above sense.

[2]The adjective 'raw' is used to emphasize that not all expressions will be 'well-formed' in the technical sense to be explained.

Definition 6.1.1. As before, a *context*, Γ, is a finite list $\Gamma = [x_1 : \sigma_1, \dots, x_n : \sigma_n]$ of (variable,type)-pairs, but now subject to the following conditions on variables: for each $i = 1, \dots, n$

$$var(\sigma_i) \subseteq \{x_1, \dots, x_{i-1}\} \quad \text{and} \quad x_i \notin \{x_1, \dots, x_{i-1}\}$$

where $var(e)$ denotes the (finite) set of variables occurring in the expression e. Thus the variables x_i are distinct and each type σ_i only involves variables which have already been listed in the context. Notational conventions for contexts will be as in Section 2.1; in particular, $[\,]$ denotes the empty context.

The variant of dependently typed Equational Logic that we are going to describe contains rules for deriving a number of different forms of judgement, set out in Table 1. The 'secondary' judgement forms are so-called because they are in fact expressible in terms of the primary forms, modulo the rules for dependently typed equational logic given below (cf. Remark 6.1.4).

Definition 6.1.2. A *dependently typed algebraic theory* , *Th*, over the signature *Sg* is specified by the following data.

- For each type-valued function symbol s a judgement

$$s(\vec{x})\ type\ [\Gamma_s]$$

  called the *introductory axiom* of s. Here $\vec{x}$ is the list of variables of the context Γ_s, and must have length n when s has arity $\text{TERMS}^n \twoheadrightarrow \text{TYPES}$. Γ_s lists the types of the argument to which s may be applied.
- For each term-valued function symbol F a judgement

$$F(\vec{x}) : \sigma_F\ [\Gamma_F]$$

  called the *introductory axiom* of F. Once again, $\vec{x}$ is the list of variables of the context Γ_F, and must have length n when F has arity $\text{TERMS}^n \twoheadrightarrow \text{TERMS}$. Γ_F lists the types of the arguments to which F may be applied, and σ_F gives the type of the result (which may depend upon those arguments).
- A collection of judgements of the form $M = M' : \sigma\ [\Gamma]$, called the *term-equality axioms* of *Th*.
- A collection of judgements of the form $\sigma = \sigma'\ [\Gamma]$, called the *type-equality axioms* of *Th*.

Given such a theory, the *theorems* of *Th* are the judgements which are provable using the rules shown in Figs 9 and 10. In the rules, $\vec{x}$ is the list of variables of context Γ and $\vec{x}'$ is the list of variables of context Γ'; and supposing $\gamma = [M_1, \dots, M_n]$ and $\vec{x} = [x_1, \dots, x_n]$, then $\sigma[\gamma/\vec{x}]$ denotes the result of simultaneously substituting the term M_i for the variable x_i

Table 1. Forms of judgement

(Γ and Γ' are contexts, σ and σ' are raw types, M and M' are raw terms, and γ and γ' are finite lists of raw terms.)

Judgement	Intended meaning	Restriction
	Primary forms	
$\sigma\ \mathit{type}\ [\Gamma]$	'σ is a well-formed type in context Γ'	$\mathit{var}(\sigma) \subseteq \mathit{var}(\Gamma)$
$M : \sigma\ [\Gamma]$	'M is a well-formed term of type σ in context Γ'	$\mathit{var}(M, \sigma) \subseteq \mathit{var}(\Gamma)$
$\sigma = \sigma'\ [\Gamma]$	'σ and σ' are equal (well-formed) types in context Γ'	$\mathit{var}(\sigma, \sigma') \subseteq \mathit{var}(\Gamma)$
$M = M' : \sigma\ [\Gamma]$	'M and M' are equal (well-formed) terms of type σ in context Γ'	$\mathit{var}(M, M', \sigma) \subseteq \mathit{var}(\Gamma)$
	Secondary forms	
$\Gamma\ \mathit{ctxt}$	'Γ is a well-formed context'	
$\gamma : \Gamma \rightarrow \Gamma'$	'γ is a context morphism from Γ to Γ''	$\mathit{var}(\gamma) \subseteq \mathit{var}(\Gamma)$
$\Gamma = \Gamma'$	'Γ and Γ' are equal (well-formed) contexts'	Γ and Γ' of equal length
$\gamma = \gamma' : \Gamma \rightarrow \Gamma'$	'γ and γ' are equal context morphisms from Γ to Γ''	$\mathit{var}(\gamma, \gamma') \subseteq \mathit{var}(\Gamma)$ and γ, γ', and Γ' of equal length

Contexts

$$\frac{}{[\,]\ \mathit{ctxt}} \qquad \frac{\sigma\ \mathit{type}\ [\Gamma]}{[\Gamma, x : \sigma]\ \mathit{ctxt}}$$

$$\frac{}{[\,] = [\,]} \qquad \frac{\Gamma = \Gamma' \quad \sigma = \sigma'[\vec{x}/\vec{x}']\ [\Gamma]}{[\Gamma, x : \sigma] = [\Gamma', x' : \sigma']}$$

Types

$$\frac{\sigma\ \mathit{type}\ [\Gamma]}{\sigma = \sigma\ [\Gamma]} \qquad \frac{\sigma = \sigma'\ [\Gamma]}{\sigma' = \sigma\ [\Gamma]} \qquad \frac{\sigma = \sigma'\ [\Gamma] \quad \sigma' = \sigma''\ [\Gamma]}{\sigma = \sigma''\ [\Gamma]}$$

$$\frac{\gamma : \Gamma' \to \Gamma \quad \sigma\ \mathit{type}\ [\Gamma]}{\sigma[\gamma/\vec{x}]\ \mathit{type}\ [\Gamma']} \qquad \frac{\gamma = \gamma' : \Gamma' \to \Gamma \quad \sigma = \sigma'\ [\Gamma]}{\sigma[\gamma/\vec{x}] = \sigma'[\gamma'/\vec{x}]\ [\Gamma']}$$

Terms

$$\frac{[\Gamma, x : \sigma, \Gamma']\ \mathit{ctxt}}{x : \sigma\ [\Gamma, x : \sigma, \Gamma']} \qquad \frac{M : \sigma\ [\Gamma] \quad \sigma = \sigma'\ [\Gamma]}{M : \sigma'\ [\Gamma]} \qquad \frac{M = M' : \sigma\ [\Gamma] \quad \sigma = \sigma'\ [\Gamma]}{M = M' : \sigma'\ [\Gamma]}$$

$$\frac{M : \sigma\ [\Gamma]}{M = M : \sigma\ [\Gamma]} \qquad \frac{M = M' : \sigma\ [\Gamma]}{M' = M : \sigma\ [\Gamma]} \qquad \frac{M = M' : \sigma\ [\Gamma] \quad M' = M'' : \sigma\ [\Gamma]}{M = M'' : \sigma\ [\Gamma]}$$

$$\frac{\gamma : \Gamma' \to \Gamma \quad M : \sigma\ [\Gamma]}{M[\gamma/\vec{x}] : \sigma[\gamma/\vec{x}]\ [\Gamma']} \qquad \frac{\gamma = \gamma' : \Gamma' \to \Gamma \quad M = M' : \sigma\ [\Gamma]}{M[\gamma/\vec{x}] = M'[\gamma'/\vec{x}] : \sigma[\gamma/\vec{x}]\ [\Gamma']}$$

Context morphisms

$$\frac{[\Gamma]\ \mathit{ctxt}}{[\,] : \Gamma \to [\,]} \qquad \frac{\gamma : \Gamma' \to \Gamma \quad \sigma\ \mathit{type}\ [\Gamma] \quad M : \sigma[\gamma/\vec{x}]\ [\Gamma']}{[\gamma, M] : \Gamma' \to [\Gamma, x : \sigma]}$$

$$\frac{[\Gamma]\ \mathit{ctxt}}{[\,] = [\,] : \Gamma \to [\,]} \qquad \frac{\gamma = \gamma' : \Gamma' \to \Gamma \quad \sigma\ \mathit{type}\ [\Gamma] \quad M = M' : \sigma[\gamma/\vec{x}]\ [\Gamma']}{[\gamma, M] = [\gamma', M'] : \Gamma' \to [\Gamma, x : \sigma]}$$

Fig. 9. General rules

$$\frac{\Gamma_s\ \mathit{ctxt}}{s(\vec{x})\ \mathit{type}\ [\Gamma_s]}\ (s \text{ has introductory axiom } s(\vec{x})\ \mathit{type}\ [\Gamma_s])$$

$$\frac{\sigma_F\ \mathit{type}\ [\Gamma_F]}{F(\vec{x}) : \sigma_F}\ (F \text{ has introductory axiom } F(\vec{x}) : \sigma_F\ [\Gamma_F])$$

$$\frac{M : \sigma\ [\Gamma] \quad M' : \sigma\ [\Gamma]}{M = M' : \sigma\ [\Gamma]}\ (M = M' : \sigma\ [\Gamma] \text{ is a term-equality axiom})$$

$$\frac{\sigma\ \mathit{type}\ [\Gamma] \quad \sigma'\ \mathit{type}\ [\Gamma]}{\sigma = \sigma'\ [\Gamma]}\ (\sigma = \sigma'\ [\Gamma] \text{ is a type-equality axiom}).$$

Fig. 10. Rules for axioms

in the type σ. In reading the rules, it should not be forgotten that we only consider contexts which are well-formed, in the sense of satisfying the conditions on variables mentioned in Definition 6.1.1. Thus for example, in the rule for deriving $[\Gamma, x : \sigma]$, it is necessarily the case that x is distinct from the variables in Γ.

The rules in Fig. 9 are general rules for dependently typed equational logic, whereas those in Fig. 10 relate to the particular axioms of *Th*. Note that these rules for using the axioms of *Th* do have hypotheses: if the hypotheses of such a rule are never satisfied, then we can never make use of the axiom.

Definition 6.1.3. We will say that a dependently typed algebraic theory *Th* is *well-formed* if all of the hypotheses of the rules for introducing axioms are provable from the rules in Figs 9 and 10. (In this case each of the axioms is indeed a theorem of *Th*).

Remark 6.1.4. The form of judgement '$\gamma : \Gamma \to \Gamma'$' is not usually made explicit in presentations of theories of dependent types. We have chosen to make it a 'first-class' judgement form because it permits a conceptually clear and concise formulation of the substitution rules, and because it will play a key role in the definition of the classifying category of a dependently-typed algebraic theory in the next section. However, it is easy to see from the rules in Fig. 9 that *this form of judgement, along with the forms '$\gamma = \gamma' : \Gamma \to \Gamma'$', '$\Gamma\ \mathit{ctxt}$' and '$\Gamma = \Gamma'$', are all equivalent to finite conjunctions of instances of the forms 'σ type $[\Gamma]$', '$M : \sigma\ [\Gamma]$', '$\sigma = \sigma'\ [\Gamma]$', and '$M = M' : \sigma\ [\Gamma]$'.*

Example 6.1.5. The theory of categories provides an example of a well-formed, dependently-typed algebraic theory. The underlying signature is:

$$\begin{aligned}
\mathit{obj} &: \textsc{Types} \\
\mathit{hom} &: \textsc{Terms}^2 \to \textsc{Types} \\
\mathit{Id} &: \textsc{Terms} \to \textsc{Terms} \\
\mathit{Comp} &: \textsc{Terms}^5 \to \textsc{Terms}.
\end{aligned}$$

The axioms are as follows

$$\mathit{obj}\ \mathit{type}\,[\,]$$

$$\mathit{hom}(x,y)\ \mathit{type}\,[x:\mathit{obj}, y:\mathit{obj}]$$

$$\mathit{Id}(x):\mathit{hom}(x,x)\ [x:\mathit{obj}]$$

$$\mathit{Comp}(x,y,z,f,g):\mathit{hom}(x,z)\ [\Gamma]$$

$$\mathit{Comp}(x,x,y,\mathit{Id}(x),f) = f : \mathit{hom}(x,y)\ [x:\mathit{obj}, y:\mathit{obj}, f:\mathit{hom}(x,y)]$$

$$\mathit{Comp}(x,y,y,f,\mathit{Id}(y)) = f : \mathit{hom}(x,y)\ [x:\mathit{obj}, y:\mathit{obj}, f:\mathit{hom}(x,y)]$$

$$\begin{aligned}
&\mathit{Comp}(x,y,w,f,\mathit{Comp}(y,z,w,g,h)) = \\
&\qquad \mathit{Comp}(x,z,w,\mathit{Comp}(x,y,z,f,g),h) : \mathit{hom}(x,w)\ [\Delta]
\end{aligned}$$

where

$$\begin{aligned}
\Gamma &\stackrel{\text{def}}{=} [x:\mathit{obj}, y:\mathit{obj}, z:\mathit{obj}, f:\mathit{hom}(x,y), g:\mathit{hom}(y,z)] \\
\Delta &\stackrel{\text{def}}{=} [x:\mathit{obj}, y:\mathit{obj}, z:\mathit{obj}, w:\mathit{obj}, \\
&\qquad f:\mathit{hom}(x,y), g:\mathit{hom}(y,z), h:\mathit{hom}(z,w)]\ .
\end{aligned}$$

It is evident from this example that the formal requirement in the introductory axiom of a function symbol (such as that for *Comp*) that all variables in the context occur explicitly as arguments of the function is at variance with informal practice. See [Cartmell, 1986, Section 10] for a discussion of this issue.

Remark 6.1.6 (Substitution and Weakening.). A general rule for substituting along a context morphism is derivable from the rules in Figs 9 and 10, viz:

$$\frac{\gamma : \Gamma' \to \Gamma \quad J\,[\Gamma]}{J[\gamma/\vec{x}]\,[\Gamma']}\ . \tag{6.1}$$

where J is one of the four forms '$e\,\mathit{type}$', '$e = e'$', '$e : e'$', or '$e = e' : e''$', and $\vec{y}$ is the list of variables in Γ'. A special case of this is a general derived rule for weakening contexts:

$$\frac{\sigma\ \mathit{type}\ [\Gamma] \quad J\ [\Gamma, \Gamma']}{J\ [\Gamma, x : \sigma, \Gamma']} \ . \tag{6.2}$$

The rules for substitution in Fig. 9 also have as special cases forms which correspond more closely to the substitution rule (2.9) for simply typed equational logic, viz:

$$\frac{M = M' : \sigma\ [\Gamma] \quad N = N' : \tau\ [\Gamma, x : \sigma, \Gamma']}{N[M/x] = N'[M'/x] : \tau[M/x]\ [\Gamma, \Gamma'[M/x]]} \tag{6.3}$$

$$\frac{M = M' : \sigma\ [\Gamma] \quad \tau = \tau'\ [\Gamma, x : \sigma, \Gamma']}{\tau[M/x] = \tau'[M'/x]\ [\Gamma, \Gamma'[M/x]]} \ . \tag{6.4}$$

6.2 Classifying category of a theory

We are now going to construct a category, $\mathcal{C}\ell(\mathit{Th})$, out of the syntax of a dependently typed algebraic theory *Th*. In fact the construction is essentially just as in Section 4.2. More accurately, the construction in that section is the special case of the one given here when the theory has only 0-ary type-valued function symbols.

We will use the following terminology with respect to the judgements which are theorems of *Th*. Say that Γ is a *Th-context* if the judgement $[\Gamma]\ \mathit{ctxt}$ is a theorem of *Th*; say that *Th*-contexts Γ and Γ' are *Th-provably equal* if the judgement $\Gamma = \Gamma'$ is a theorem of *Th*; say that $\gamma : \Gamma \to \Gamma'$ is a *morphism of Th-contexts* if the judgement $\gamma : \Gamma \to \Gamma'$ is a theorem of *Th*; and finally, say that two *Th*-context morphisms are *Th-provably equal* if $\gamma = \gamma' : \Gamma \to \Gamma'$ is a theorem of *Th*.

Objects of $\mathcal{C}\ell(\mathit{Th})$. The collection of objects of the classifying category is the quotient of the set of *Th*-contexts by the equivalence relation of being *Th*-provably equal. This definition makes sense because the following rules are derivable from those in Figs 9 and 10.

$$\frac{\Gamma\ \mathit{ctxt}}{\Gamma = \Gamma} \qquad \frac{\Gamma = \Gamma'}{\Gamma' = \Gamma} \qquad \frac{\Gamma = \Gamma' \quad \Gamma' = \Gamma''}{\Gamma = \Gamma''} \ .$$

We will tend not to distinguish notationally between a *Th*-context and the object of $\mathcal{C}\ell(\mathit{Th})$ that it determines.

Morphisms of $\mathcal{C}\ell(\mathit{Th})$. Given two objects in $\mathcal{C}\ell(\mathit{Th})$, represented by *Th*-contexts Γ and Γ' say, the collection of morphisms in the classifying category from the first object to the second is the quotient of the set of *Th*-context morphisms $\Gamma \to \Gamma'$ by the equivalence relation of being *Th*-provably equal. This definition makes sense because the following rules are derivable from those in Figs 9 and 10.

$$\frac{\Gamma = \Gamma' \quad \gamma : \Gamma' \to \Gamma'' \quad \Gamma'' = \Gamma'''}{\gamma : \Gamma \to \Gamma'''}$$

$$\frac{\gamma : \Gamma \to \Gamma'}{\gamma = \gamma : \Gamma \to \Gamma'} \qquad \frac{\gamma = \gamma' : \Gamma \to \Gamma'}{\gamma' = \gamma : \Gamma \to \Gamma'}$$

$$\frac{\gamma = \gamma' : \Gamma \to \Gamma' \quad \gamma' = \gamma'' : \Gamma \to \Gamma'}{\gamma = \gamma'' : \Gamma \to \Gamma'} \ .$$

We will tend not to distinguish notationally between a *Th*-context morphism and the morphism of $\mathcal{C}\ell(Th)$ which it determines.

Composition in $\mathcal{C}\ell(Th)$. Given context morphisms $\gamma : \Gamma \to \Gamma'$ and $\gamma' : \Gamma' \to \Gamma''$, with $\gamma' = [N_1, \ldots, N_m]$ and $var(\Gamma') = \vec{x}'$ say, define the *composition* $\gamma' \circ \gamma : \Gamma \to \Gamma''$ just as in Section 4.2, viz:

$$\gamma' \circ \gamma \overset{\text{def}}{=} [N_1[\gamma/\vec{x}'], \ldots, N_m[\gamma/\vec{x}']] \ .$$

The derived rules for substitution mentioned in Remark 6.1.6 can be used to show that the following derived rules for composition are valid.

$$\frac{\gamma : \Gamma \to \Gamma' \quad \gamma' : \Gamma' \to \Gamma''}{\gamma' \circ \gamma : \Gamma \to \Gamma''}$$

$$\frac{\gamma_1 = \gamma_2 : \Gamma \to \Gamma' \quad \gamma_1' = \gamma_2' : \Gamma' \to \Gamma''}{\gamma_1' \circ \gamma_1 = \gamma_2' \circ \gamma_2 : \Gamma \to \Gamma''}$$

$$\frac{\gamma : \Gamma \to \Gamma' \quad \gamma' : \Gamma' \to \Gamma'' \quad \gamma'' : \Gamma'' \to \Gamma'''}{\gamma'' \circ (\gamma' \circ \gamma) = (\gamma'' \circ \gamma') \circ \gamma : \Gamma \to \Gamma'''} \ .$$

From these rules it follows that a well-defined and associative operation of composition is induced on the morphisms of the classifying category. The identity morphisms for this composition are induced by the *identity context morphisms* $id_\Gamma : \Gamma \to \Gamma$, which are defined just as in Section 4.2:

$$id_\Gamma \overset{\text{def}}{=} [x_1, \ldots, x_n]$$

where $x_1, \ldots, x_n$ are the variables listed in Γ. These context morphisms do induce morphisms in $\mathcal{C}\ell(Th)$ which are units for composition because the following rules are derivable.

$$\frac{\Gamma \ ctxt}{id_\Gamma : \Gamma \to \Gamma} \qquad \frac{\Gamma = \Gamma'}{id_\Gamma = id_{\Gamma'} : \Gamma \to \Gamma}$$

$$\frac{\gamma : \Gamma \to \Gamma'}{\gamma \circ id_\Gamma = \gamma : \Gamma \to \Gamma'} \qquad \frac{\gamma : \Gamma \to \Gamma'}{id'_\Gamma \circ \gamma = \gamma : \Gamma \to \Gamma'} \ .$$

This completes the definition of $\mathcal{C}\ell(Th)$ as a category. We now move on to examine what extra categorical structure it possesses.

6.3 Type-categories

For a simply typed algebraic theory, we saw in Section 4 that the relevant categorical structure on the corresponding classifying category was finite products. Here it will turn out to be the more general[3] property of possessing a terminal object and *some* pullbacks. To explain which pullbacks, we identify a special class of morphisms in the classifying category of *Th*.

Definition 6.3.1. If Γ is a *Th*-context, the collection of Γ-*indexed types* in $\mathcal{C}\ell(Th)$ is defined to be the quotient of the set of types σ such that σ *type* $[\Gamma]$ is a theorem of *Th*, with respect to the equivalence relation identifying σ and σ' just in case $\sigma = \sigma'$ $[\Gamma]$ is a theorem of *Th*. As usual, we will not make a notational distinction between σ and the Γ-indexed type it determines. Each such Γ-indexed type has associated with it a *projection morphism*, represented by the context morphism

$$\pi_\sigma \stackrel{\text{def}}{=} [x_1, \ldots, x_n] : [\Gamma, x : \sigma] \to \Gamma.$$

Here $x_1, \ldots, x_n$ are the variables listed in Γ, and x is any other variable. Note that the object in $\mathcal{C}\ell(Th)$ represented by $[\Gamma, x : \sigma]$ and the morphism represented by π_σ are independent of which particular variable x is chosen.

Lemma 6.3.2. *Given a morphism* $\gamma : \Gamma' \longrightarrow \Gamma$ *in* $\mathcal{C}\ell(Th)$ *and a* Γ*-indexed type represented by* σ*, then*

$$\begin{tikzcd}
{[\Gamma', y : \sigma[\gamma/\vec{x}]]} \arrow[r, "{[\gamma, y]}"] \arrow[d, "\pi_{\sigma[\gamma/\vec{x}]}"'] & {[\Gamma, x : \sigma]} \arrow[d, "\pi_\sigma"] \\
\Gamma' \arrow[r, "\gamma"'] & \Gamma
\end{tikzcd}$$

is a pullback square in $\mathcal{C}\ell(Th)$.

Proof. Suppose given $\gamma' : \Gamma'' \to \Gamma'$ and $\gamma'' : \Gamma'' \to [\Gamma, x : \sigma]$ in $\mathcal{C}\ell(Th)$ satisfying $\gamma \circ \gamma' = \pi_\sigma \circ \gamma'' : \Gamma'' \to \Gamma$. We have to prove that there is a unique morphism $\delta : \Gamma'' \to [\Gamma', x' : \sigma[\gamma/\vec{x}]]$ satisfying $\pi_{\sigma[\gamma/\vec{x}]} \circ \delta = \gamma' : \Gamma'' \to \Gamma'$ and $[\gamma, x'] \circ \delta = \gamma'' : \Gamma'' \to [\Gamma, x : \sigma]$.

[3]Since finite products can be constructed from pullbacks and a terminal object.

Now since $\gamma \circ \gamma' = \pi_\sigma \circ \gamma'' : \Gamma'' \to \Gamma$, we must have that the list of terms γ'' is of the form $[\gamma \circ \gamma', N]$ for some term N for which $N : \sigma[\gamma \circ \gamma'/\vec{x}]\ [\Gamma'']$ is a theorem of *Th*. Now since $\sigma[\gamma \circ \gamma'/\vec{x}] = \sigma[\gamma/\vec{x}][\gamma'/\vec{x}']$ (where $\vec{x}'$ is the list of variables in Γ'), we get a morphism

$$\delta \overset{\text{def}}{=} [\gamma', N] : \Gamma'' \to [\Gamma', x' : \sigma[\gamma/\vec{x}]]$$

satisfying

$$\begin{aligned} [\gamma, x'] \circ \delta &= [\gamma, x'] \circ [\gamma', N] \\ &= [\gamma \circ \gamma', N] \\ &= \gamma'' \end{aligned}$$

and

$$\begin{aligned} \pi_{\sigma[\gamma/\vec{x}]} \circ \delta &= [\vec{x}'\,] \circ [\gamma', N] \\ &= \gamma' \end{aligned}$$

as required. If $\delta' : \Gamma'' \to [\Gamma', x' : \sigma[\gamma/\vec{x}]]$ were any other such morphism, then from the requirement $\pi_{\sigma[\gamma/\vec{x}]} \circ \delta' = \gamma'$ we conclude that the list δ' is of the form $[\gamma', N']$; and then from the requirement $[\gamma, x'] \circ \delta' = \gamma''$ we conclude further that

$$N' = N : \sigma[\gamma/\vec{x}][\gamma'/\vec{x}']\ [\Gamma'']$$

is a theorem of *Th*, so that $\delta' = \delta : \Gamma'' \to [\Gamma', x' : \sigma[\gamma/\vec{x}]]$. ∎

The above Lemma motivates the following definition, which is essentially Cartmell's (unpublished) notion of 'category with attributes'.

Definition 6.3.3. A *type-category*, $\mathcal{C}$, is a category possessing a terminal object, *1*, and equipped with the following structure:

- For each object X in $\mathcal{C}$, a collection $\mathit{Type}_{\mathcal{C}}(X)$, whose elements will be called *X-indexed types* in $\mathcal{C}$.
- For each object X in $\mathcal{C}$, operations assigning to each X-indexed type A an object $X \ltimes A$, called the *total object* of A, together with a morphism

  $$\pi_A : X \ltimes A \longrightarrow X$$

  called the *projection morphism* of A.
- For each morphism $f : Y \longrightarrow X$ in $\mathcal{C}$, an operation assigning to each X-indexed type A, a Y indexed type f^*A, called the *pullback of A along X*, together with a morphism $f \ltimes A : Y \ltimes f^*A \longrightarrow X \ltimes A$ making the following a pullback square in the category $\mathcal{C}$.

$$\begin{tikzcd}
Y \ltimes f^*A \arrow[r, "f \ltimes A"] \arrow[d, "\pi_{f^*A}"'] & X \ltimes A \arrow[d, "\pi_A"] \\
Y \arrow[r, "f"'] & X
\end{tikzcd} \tag{6.5}$$

In addition, the following *strictness conditions* will be imposed on these operations:

$$id_X^* A = A \quad \text{and} \quad id_X \ltimes A = id_{X \ltimes A}$$

$$g^*(f^*A) = (f \circ g)^*A \quad \text{and} \quad (f \ltimes A) \circ (g \ltimes f^*A) = (f \circ g) \ltimes A \ .$$

Notation 6.3.4. If $\mathcal{C}$ is a type-category and X is a $\mathcal{C}$-object, then as usual C/X will denote the *slice category* whose objects are the $\mathcal{C}$-morphisms with codomain X, $f : dom(f) \longrightarrow X$, and whose morphisms $f \longrightarrow f'$ are $\mathcal{C}$-morphisms $g : dom(f) \longrightarrow dom(f')$ satisfying $f' \circ g = f$. Identity morphisms and composition in $\mathcal{C}/X$ are inherited from those in $\mathcal{C}$. Note that for each $A \in Type_{\mathcal{C}}(X)$, the projection morphism π_A is an object in the slice category $\mathcal{C}/X$. So we can make $Type_{\mathcal{C}}(X)$ into a category (equivalent to a full subcategory of $\mathcal{C}/X$) by taking morphisms $A \longrightarrow A'$ to be $\mathcal{C}/X$-morphisms $\pi_A \longrightarrow \pi_{A'}$ (with identities and composition inherited from $\mathcal{C}/X$).

Proposition 6.3.5. *The classifying category, $\mathcal{C}\ell(Th)$, of a dependently typed algebraic theory Th is a type-category.*

Proof. We identified notions of indexed type and associated projection morphisms for $\mathcal{C}\ell(Th)$ in Definition 6.3.1 and verified in Lemma 6.3.2 the existence of pullback squares of the required form (6.5). The 'strictness' conditions in Definition 6.3.3 are met because of the usual properties of substitution of terms for variables. Finally, note that $\mathcal{C}\ell(Th)$ does have a terminal object, namely the (equivalence class of the) empty context $[\,]$. ∎

Here are some examples of 'naturally-occurring' type-categories.

Example 6.3.6 (Sets). The category $\mathcal{S}et$ of sets and functions supports the structure of a type-category: for each set X, we can take the X-indexed types to be set-valued functions with domain X. Given such a function $A : X \longrightarrow \mathcal{S}et$, the associated total set is the disjoint union

$$X \ltimes A \overset{\text{def}}{=} \textstyle\coprod_{x \in X} A(x) = \{(x, a) \mid x \in X \ \& \ a \in A(x)\}$$

with projection function π_A given by first projection $(x, a) \mapsto x$.

The pullback of A along a function $f : Y \longrightarrow X$ is just the composition

$$f^*A \overset{\text{def}}{=} A \circ f : Y \longrightarrow \mathcal{S}et$$

which indeed results in a pullback square in $\mathcal{S}et$ of the required form (6.5) and satisfying the strictness conditions, with $f \ltimes A : Y \ltimes f^*A \longrightarrow X \ltimes A$ equal to the function $(y, a) \mapsto (f(y), a)$.

Note the following property of this example, which is not typical of type-categories in general: *up to bijection over X, any function with codomain X, $f : Y \longrightarrow X$, can be expressed as the projection function associated with an indexed type*:

$$f = \left(Y \overset{i}{\cong} X \ltimes A \xrightarrow{\pi_A} X \right)$$

where $A(x) = \{y \in Y \mid f(y) = x\}$ and $i(y) = (f(y), y)$.

Example 6.3.7 (Constant families). Every category $\mathcal{C}$ with finite products can be endowed with the structure of a type-category, in which the indexed types are 'constant families'. For each object X in $\mathcal{C}$, one defines $\mathit{Type}_{\mathcal{C}}(X)$ to be $\mathit{Obj}_{\mathcal{C}}$, the class of all objects of $\mathcal{C}$. The associated operations are

$$\begin{array}{rcll} X \ltimes A & \overset{\text{def}}{=} & X \times A & \text{(categorical product)} \\ \pi_A & \overset{\text{def}}{=} & \pi_1 : X \times A \longrightarrow X & \text{(first projection).} \end{array}$$

Given $f : Y \longrightarrow X$ in $\mathcal{C}$ and $A \in \mathit{Type}_{\mathcal{C}}(X) = \mathit{Obj}_{\mathcal{C}}$, the pullback Y-indexed type f^*A is just the object A itself and the morphism $f \ltimes A : Y \ltimes f^*A \longrightarrow X \ltimes A$ is $f \times id_A$. This does yield a pullback square in $\mathcal{C}$, viz.:

$$\begin{tikzcd} Y \times A \arrow[r, "f \times id_A"] \arrow[d, "\pi_1"'] & X \times A \arrow[d, "\pi_1"] \\ Y \arrow[r, "f"'] & X \end{tikzcd}$$

and these pullbacks do satisfy the strictness conditions in Definition 6.3.3. When a category with finite products is regarded as a type-category in this way, the categorical semantics of simply typed algebraic theories given in Section 2 becomes essentially a special case of the semantics of dependently typed theories to be given below.

Example 6.3.8 (Toposes). The following example is given for readers familiar with the notion of *topos* (see the references in Section 7). One can make each topos $\mathcal{E}$ into a type-category as follows. First note that $\mathcal{E}$ is indeed a category with a terminal object. For each object $X \in \mathit{Obj}_{\mathcal{E}}$, define $\mathit{Type}_{\mathcal{E}}(X)$ to consist of all pairs (A, a) where $A \in \mathit{Obj}_{\mathcal{E}}$ and $a : X \times A \longrightarrow \Omega$. Here Ω denotes the codomain of the object classifier of $\mathcal{E}$, $\top : \mathit{1} \longrightarrow \Omega$. Given such an X-indexed type, the total object $X \ltimes (A, a)$ is given by forming a pullback square in $\mathcal{E}$:

$$\begin{tikzcd} X \ltimes (A,a) \arrow[r] \arrow[d] & X \times A \arrow[d, "a"] \\ \mathit{1} \arrow[r, "\top"'] & \Omega \end{tikzcd} \tag{6.6}$$

The projection morphism associated to (A, a) is defined by the composition

$$\pi_{(A,a)} : X \ltimes (A, a) \longrightarrow X \times A \xrightarrow{\pi_1} X \ .$$

For any morphism $f : Y \longrightarrow X$, $f^*(A, a)$ is defined to be the Y-indexed type $(A, a \circ (f \times id_A))$. The pullback square (6.5) is obtained as the factorization through (6.6) of the corresponding pullback for $a \circ (f \times id_A)$. Elementary properties of pullback squares ensure that the strictness conditions of Definition 6.3.3 hold, *without* the need for any coherence assumptions on the chosen pullback operations for $\mathcal{E}$. Traditionally, dependently typed theories have been interpreted in toposes using the *locally cartesian closed structure* that they possess, whereby the category $\mathit{Type}_{\mathcal{E}}(X)$ of X-indexed types (cf. 6.3.4) is just the slice category $\mathcal{E}/X$: see [Seely, 1984]. Such a choice for indexed types yields a 'non-strict' type-category structure, in that the strictness conditions of Definition 6.3.3 have to be weakened by replacing the equalities by coherent isomorphisms. This can lead to complications in the interpretation of type theory: see [Curien, 1990]. The method of regarding a topos as a type-category given here achieves the strictness conditions and side-steps issues of coherence. Given that toposes correspond to theories in intuitionistic higher order logic (cf. [Bell, 1988], [Mac Lane and Moerdijk, 1992]), this example can form the basis for interpreting dependently typed theories in higher order predicate logic: see [Jacobs and Melham, 1993]. However, Hofmann [1995] has recently pointed out that using a particular construction of Bénabou (for producing a split fibration equivalent to a given fibration), an *arbitrary* locally cartesian closed category, $\mathcal{E}$, can be endowed with a more involved notion of X-indexed type which makes $\mathcal{E}$ a type-category (supporting the interpretation of product, sum and identity types) and with $\mathit{Type}_{\mathcal{E}}(X)$ equivalent to $\mathcal{E}/X$ (pseudo-naturally in X).

Example 6.3.9 (Split fibrations). Let $\mathcal{C}at$ denote the category of small categories and functors. We can turn it into a type-category by decreeing that for each small category, $\mathbf{C}$, the $\mathbf{C}$-indexed types are functors $\mathbf{A} : \mathbf{C}^{\mathrm{op}} \longrightarrow \mathcal{C}at$. The associated total category $\mathbf{C} \ltimes \mathbf{A}$ is given by a construction due to Grothendieck:

- An object of $\mathbf{C} \ltimes \mathbf{A}$ is a pair (X, A), with X an object of $\mathbf{C}$ and A an object of $\mathbf{A}(X)$.
- A morphism in $\mathbf{C} \ltimes \mathbf{A}$ from (X, A) to (X', A') is a pair (x, a), where $x : X \longrightarrow X'$ in $\mathbf{C}$ and $a : A \longrightarrow \mathbf{A}(x)(A')$ in $\mathbf{A}(X)$. The composition of two such morphisms $(x, a) : (X, A) \longrightarrow (X', A')$ and $(x', a') : (X', A') \longrightarrow (X'', A'')$ is given by the pair $(x' \circ x, \mathbf{A}(x)(a') \circ a)$. The identity morphism for the object (X, A) is given by (id_X, id_A).

The projection functor $\pi_{\mathbf{A}} : \mathbf{C} \ltimes \mathbf{A} \longrightarrow \mathbf{C}$ is of course given on both objects and morphisms by projection onto the first co-ordinate.

The pullback of the indexed type $\mathbf{A} : \mathbf{C}^{\mathrm{op}} \longrightarrow \mathcal{C}at$ along a functor $F : \mathbf{D} \longrightarrow \mathbf{C}$ is just given by composing $\mathbf{A}$ with F, regarded as a functor $\mathbf{D}^{\mathrm{op}} \longrightarrow \mathbf{C}^{\mathrm{op}}$. Applying the Grothendieck construction to $\mathbf{A}$ and to $\mathbf{A} \circ F$, one obtains a pullback square in the category $\mathcal{C}at$ of the required form, with $F \ltimes \mathbf{A} : \mathbf{D} \ltimes (\mathbf{A} \circ F) \longrightarrow \mathbf{C} \ltimes \mathbf{A}$ the functor which acts on objects and morphisms by applying F in the first co-ordinate.

Regarding a set as a discrete category, there is an inclusion between the type-category of Example 6.3.6 and the present one. Unlike the previous example, not every morphism in $\mathcal{C}at$ arises as the first projection of an indexed type. Those that do were characterized by Grothendieck and are known as 'split Grothendieck fibrations'. As the name suggests, these are a special instances of the more general notion of 'Grothendieck fibration'—which would give a more general way of making $\mathcal{C}at$ into a type-category, except that the 'strictness' conditions in Definition 6.3.3 are not satisfied. See [Jacobs, 1991] for more information on the use of Grothendieck fibrations in the semantics of type theory.

6.4 Categorical semantics

We will now give the definition of the semantics of the types, terms, contexts and context morphisms of Th in a type-category $\mathcal{C}$. In general contexts are interpreted as $\mathcal{C}$-objects, context morphisms as $\mathcal{C}$-morphisms, and types as indexed types in $\mathcal{C}$. Terms as interpreted using the following notion of 'global section' of an indexed type.

Definition 6.4.1. Given an object X in a type-category $\mathcal{C}$, the *global sections* of an X-indexed type $A \in Type_{\mathcal{C}}(X)$ are the morphisms $a : X \longrightarrow X \ltimes A$ in $\mathcal{C}$ satisfying $\pi_A \circ a = id_X$. We will write

$$a \in_X A$$

to indicate that a is a global section of the X-indexed type A. For any morphism $f : Y \longrightarrow X$, using the universal property of the pullback square (6.5) we get

$$f^* a \in_Y f^* A$$

where $f^* a : Y \longrightarrow Y \ltimes f^* A$ is the unique morphism with $\pi_{f^*A} \circ f^* a = id_Y$ and $(f \ltimes A) \circ f^* a = a \circ f$.

Definition 6.4.2. Let Sg be a signature as in Section 6.1. A *structure* for Sg in a type-category $\mathcal{C}$ is specified by:

- For each type-valued function symbol s, a $\mathcal{C}$-object X_s and an indexed type $A_s \in \mathit{Type}_{\mathcal{C}}(X_s)$.
- For each term-valued function symbol F, a $\mathcal{C}$-object X_F, an indexed type $A_F \in \mathit{Type}_{\mathcal{C}}(X_F)$, and a global section $a_F \in_{X_F} A_F$.

Semantics of expressions. Given such a structure, we will define four relations:

$$\begin{aligned} [\![\Gamma\ ctxt]\!] &\asymp X \\ [\![\sigma\ type\ [\Gamma]]\!] &\asymp A\,[X] \\ [\![M : \sigma\ [\Gamma]]\!] &\asymp a : A\,[X] \\ [\![\gamma : \Gamma' {\rightarrow} \Gamma]\!] &\asymp f : X' {\rightarrow} X \end{aligned}$$

where the judgements enclosed by $[\![\quad]\!]$ are all well-formed and where X is a $\mathcal{C}$-object, $A \in \mathit{Type}_{\mathcal{C}}(X)$, $a \in_X A$, and f is a morphism from X' to X in $\mathcal{C}$. These relations are defined inductively by the rules in Fig. 11. In the first rule concerning context morphisms, $\langle\,\rangle : X {\rightarrow} \mathit{1}$ denotes the unique morphism from X to the terminal object. The two rules concerning the semantics of variables use the following pairing notation.

Notation 6.4.3. Referring to the pullback square (6.5), if $g : Z \longrightarrow Y$ and $h : Z \longrightarrow X \ltimes A$ are morphisms satisfying $f \circ g = \pi_A \circ h$, then

$$\langle g, h \rangle : Z \longrightarrow Y \ltimes f^* A$$

will denote the unique morphism satisfying $\pi_{f^*A} \circ \langle g, h \rangle = g$ and $(f \ltimes A) \circ \langle g, h \rangle = h$ whose existence is guaranteed by the universal property of the pullback square. Thus for example, in this notation the global section $f^* a \in_Y f^* A$ referred to in Definition 6.4.1 is given by the morphism $\langle id_Y, a \circ f \rangle : Y \longrightarrow Y \ltimes f^* A$.

It is clear from the form of the rules in Fig. 11 that the relations they define are *monogenic*, in the sense that for a judgement J and categorical structures E and E', if $[\![J]\!] \asymp E$ and $[\![J]\!] \asymp E'$, then $E = E'$ in $\mathcal{C}$. Thus given a structure for Sg in $\mathcal{C}$, the assignments

Contexts

$$\frac{}{[\![[\,]\ \mathit{ctxt}]\!] \asymp \mathit{1}}$$

$$\frac{[\![\Gamma\ \mathit{ctxt}]\!] \asymp X \quad [\![\sigma\ \mathit{type}\ [\Gamma]]\!] \asymp A\ [X]}{[\![[\Gamma, x : \sigma]\ \mathit{ctxt}]\!] \asymp X \ltimes A}$$

Types

$$\frac{[\![\gamma : \Gamma \rightarrow \Gamma_s]\!] \asymp f : X \rightarrow X_s}{[\![s(\gamma)\ \mathit{type}\ [\Gamma]]\!] \asymp f^* A_s\ [X]}$$

Terms

$$\frac{[\![\Gamma\ \mathit{ctxt}]\!] \asymp X \quad [\![\sigma\ \mathit{type}\ [\Gamma]]\!] \asymp A\ [X] \qquad [\![\tau\ \mathit{type}\ [\Gamma, x : \sigma]]\!] \asymp \pi_A^* A\ [X \ltimes A]}{[\![x : \tau\ [\Gamma, x : \sigma]]\!] \asymp \langle id_{X \ltimes A}, id_{X \ltimes A} \rangle : \pi_A^* A\ [X \ltimes A]}$$

$$\frac{[\![\Gamma\ \mathit{ctxt}]\!] \asymp X \quad [\![\sigma\ \mathit{type}\ [\Gamma]]\!] \asymp A\ [X] \quad [\![\sigma'\ \mathit{type}\ [\Gamma, x : \sigma]]\!] \asymp A'\ [X \ltimes A] \qquad [\![\tau\ \mathit{type}\ [\Gamma, x : \sigma, x' : \sigma']]\!] \asymp (\pi_{A'} \circ \pi_A)^* A\ [X \ltimes A \ltimes A']}{[\![x : \tau\ [\Gamma, x : \sigma, x' : \sigma']]\!] \asymp \langle id_{X \ltimes A \ltimes A'}, \pi_{A'} \rangle : (\pi_{A'} \circ \pi_A)^* A\ [X \ltimes A \ltimes A']}$$

$$\frac{[\![\gamma : \Gamma \rightarrow \Gamma_F]\!] \asymp f : X \rightarrow X_F \quad [\![\sigma\ \mathit{type}\ [\Gamma]]\!] \asymp f^* A_F\ [X]}{[\![F(\gamma) : \sigma\ [\Gamma]]\!] \asymp f^* a_F : f^* A_F\ [X]}$$

Context morphisms

$$\frac{[\![\Gamma\ \mathit{ctxt}]\!] \asymp X}{[\![[\,] : \Gamma \rightarrow [\,]]\!] \asymp \langle\,\rangle : X \rightarrow \mathit{1}}$$

$$\frac{[\![\gamma : \Gamma' \rightarrow \Gamma]\!] \asymp f : X' \rightarrow X \quad [\![\sigma\ \mathit{type}\ [\Gamma]]\!] \asymp A\ [X] \qquad [\![M : \sigma[\gamma/\vec{x}]\ [\Gamma']]\!] \asymp a : f^* A\ [X']}{[\![[\gamma, M] : \Gamma' \rightarrow [\Gamma, x : \sigma]]\!] \asymp (f \ltimes A') \circ a : X' \rightarrow X \ltimes A}$$

Fig. 11. Categorical semantics

$$\begin{array}{rcl} \Gamma & \mapsto & [\![\Gamma\ ctxt]\!] \\ \sigma, \Gamma & \mapsto & [\![\sigma\ type\ [\Gamma]]\!] \\ M, \sigma, \Gamma & \mapsto & [\![M : \sigma\ [\Gamma]]\!] \\ \gamma, \Gamma', \Gamma & \mapsto & [\![\gamma : \Gamma' \to \Gamma]\!] \end{array}$$

determine partial functions. We will write $[\![J]\!]\Downarrow$ to indicate that these partial functions are defined at a particular argument J.

Remark 6.4.4. The case when J is $x : \tau\ [\Delta]$ deserves some comment, since this is the most complicated case in the definition of $[\![J]\!]$. The well-formedness of $x : \tau\ [\Delta]$ guarantees that x is assigned a type in Δ, i.e. that Δ is of the form $[\Gamma, x : \sigma, \Gamma']$. However, the presence of type equality judgements in the logic means that σ is not necessarily syntactically identical to τ. This accounts for the presence of the hypotheses concerning τ in the two rules for this case in Fig. 11. The first of these two rules deals with the case when $\Gamma' = [\,]$, and then the second rule must be applied repeatedly to deal with general Γ'.

Satisfaction of judgements Next we define what it means for the various forms of judgement to be *satisfied* by a structure for *Sg* in a type-category, $\mathcal{C}$.

- $\Gamma\ ctxt$ is satisfied if and only if $[\![\Gamma\ ctxt]\!]\Downarrow$.
- $\sigma\ type\ [\Gamma]$ is satisfied if and only if $[\![\sigma\ type\ [\Gamma]]\!]\Downarrow$.
- $M : \sigma\ [\Gamma]$ is satisfied if and only if $[\![M : \sigma\ [\Gamma]]\!]\Downarrow$.
- $\gamma : \Gamma \to \Gamma'$ is satisfied if and only if $[\![\gamma : \Gamma \to \Gamma']\!]\Downarrow$.
- $\Gamma = \Gamma'$ is satisfied if and only if for some (necessarily unique) object X
$$[\![\Gamma\ ctxt]\!] \asymp X \quad \text{and} \quad [\![\Gamma'\ ctxt]\!] \asymp X\ .$$
- $\sigma = \sigma'\ [\Gamma]$ is satisfied if and only if for some (necessarily unique) object X and X-indexed type $A \in \mathit{Type}_{\mathcal{C}}(X)$
$$[\![\sigma\ type\ [\Gamma]]\!] \asymp A\ [X] \quad \text{and} \quad [\![\sigma'\ type\ [\Gamma]]\!] \asymp A\ [X]\ .$$
- $M = M' : \sigma\ [\Gamma]$ is satisfied if and only if for some (necessarily unique) object X, indexed type $A \in \mathit{Type}_{\mathcal{C}}(X)$, and global section $a \in_X A$
$$[\![M : \sigma\ [\Gamma]]\!] \asymp a : A\ [X] \quad \text{and} \quad [\![M' : \sigma\ [\Gamma]]\!] \asymp a : A\ [X]\ .$$
- $\gamma = \gamma' : \Gamma \to \Gamma'$ is satisfied if and only if for some (necessarily unique) morphism $f : X' \longrightarrow X$ in C
$$[\![\gamma : \Gamma' \to \Gamma]\!] \asymp f : X' \to X \quad \text{and} \quad [\![\gamma' : \Gamma' \to \Gamma]\!] \asymp f : X' \to X\ .$$

Suppose *Th* is a dependently typed algebraic theory. A *model* of *Th* in a type-category, $\mathcal{C}$, is a structure in $\mathcal{C}$ for the underlying signature of

Th with the property that the collection of judgements satisfied by the structure is closed under the rules for the axioms of *Th* given in Fig. 10.

Theorem 6.4.5 (Soundness). *The collection of judgements that are satisfied by a structure in a type-category, $\mathcal{C}$, is closed under the general rules of dependently typed equational logic given in Fig. 9. Consequently, a model in $\mathcal{C}$ of a dependently typed algebraic theory, Th, satisfies all the judgements which are theorems of Th.*

Much as with the corresponding result for simply typed equational logic, the key tool used in proving the above theorem is a lemma giving the behaviour of substitution in the categorical semantics:

Lemma 6.4.6 (Semantics of substitution). *Suppose that* $[\![\gamma : \Gamma' \to \Gamma]\!] \asymp f : X' \to X$. *Then*

$$\begin{array}{rcl} [\![\sigma\ \mathit{type}\ [\Gamma]]\!] \asymp A\ [X] & \Rightarrow & [\![\sigma[\gamma/\vec{x}]\ \mathit{type}\ [\Gamma']]\!] \asymp f^*A\ [X'] \\ [\![M : \sigma\ [\Gamma]]\!] \asymp a : A\ [X] & \Rightarrow & [\![M[\gamma/\vec{x}] : \sigma[\gamma/\vec{x}]\ [\Gamma']]\!] \asymp f^*a : f^*A\ [X'] \\ [\![\gamma' : \Gamma'' \to \Gamma']\!] \asymp f' : X'' \to X' & \Rightarrow & [\![\gamma \circ \gamma' : \Gamma'' \to \Gamma]\!] \asymp f \circ f' : X'' \to X\ . \end{array}$$

The generic model of a well-formed theory. Suppose that *Th* is a dependently typed algebraic theory that is well-formed, in the sense of Definition 6.1.3. Then the classifying category $\mathcal{C}\ell(\mathit{Th})$ contains a structure, G, for the underlying signature of *Th*. G is defined as follows, where we use a notation for the components of the structure as in Definition 6.4.2.

The well-formedness of *Th* implies that for each type-valued function symbol s, with introductory axiom $s(\vec{x})\ \mathit{type}\ [\Gamma_s]$ say, Γ_s is a *Th*-context and hence determines an object X_s of $\mathcal{C}\ell(\mathit{Th})$. Then define the X_s-indexed type $A_s \in \mathit{Type}_{\mathcal{C}\ell(\mathit{Th})}(X_s)$ to be that represented by $s(\vec{x})$ (which is a Γ_s-indexed type because $s(\vec{x})\ \mathit{type}\ [\Gamma_s]$ is a theorem of *Th*).

For each term-valued function symbol F with introductory axiom $F(\vec{x}) : \sigma_F\ [\Gamma_F]$, the well-formedness of *Th* implies that $\sigma_F\ \mathit{type}\ [\Gamma_F]$ is a theorem of *Th*. Hence Γ_F is a *Th*-context and hence determines an object $X_F \overset{\mathrm{def}}{=} \Gamma_F$ of $\mathcal{C}\ell(\mathit{Th})$. Then define the X_F-indexed type $A_F \in \mathit{Type}_{\mathcal{C}\ell(\mathit{Th})}(X_F)$ to be that represented by the Γ_F-indexed type σ_F, and define the global section $a_F \in_{X_F} A_F$ to be the morphism $X_F \longrightarrow X_F \ltimes A_F$ in $\mathcal{C}\ell(\mathit{Th})$ represented by the *Th*-context morphism $[\mathit{id}_{\Gamma_F}, F(\vec{x})] : \Gamma_F \longrightarrow [\Gamma_F, x : \sigma_F]$.

The structure G has the following properties:

- For each judgement $\Gamma\ \mathit{ctxt}$ and $\mathcal{C}\ell(\mathit{Th})$-object X, the relation $[\![\Gamma]\!] \asymp X$ holds if and only if $\Gamma\ \mathit{ctxt}$ is a theorem of *Th* and X is the object of $\mathcal{C}\ell(\mathit{Th})$ represented by Γ.
- For each judgement $\sigma\ \mathit{type}\ [\Gamma]$, each object X and each X-indexed type $A \in \mathit{Type}_{\mathcal{C}\ell(\mathit{Th})}(X)$, the relation $[\![\sigma\ \mathit{type}\ [\Gamma]]\!] \asymp A\ [X]$ holds if and only if $\sigma\ \mathit{type}\ [\Gamma]$ is a theorem of *Th*, X is the object of $\mathcal{C}\ell(\mathit{Th})$ represented by Γ, and A is the X-indexed type represented by σ.

- For each judgement $M : \sigma\ [\Gamma]$, each object X and each X-indexed type $A \in \mathit{Type}_{\mathcal{C}\ell(\mathit{Th})}(X)$, and each global section $a \in_X A$, the relation $[\![M : \sigma\ [\Gamma]]\!] \asymp a \in A\ [X]$ holds if and only if X is the object of $\mathcal{C}\ell(\mathit{Th})$ represented by Γ, A is the X-indexed type represented by σ, and a is the global section represented by the Th-context morphism $[id_\Gamma, M] : \Gamma \longrightarrow [\Gamma, x : \sigma]$.
- An equality judgement is satisfied by G if and only if it is a theorem of Th.

Thus G is a model of Th and a judgement is satisfied by this model if and only if it is a theorem of Th.

Remark 6.4.7. The classifying type-category $\mathcal{C}\ell(\mathit{Th})$ and the generic model G of a well-formed, dependently typed algebraic theory Th have a universal property analogous to that given for algebraic theories in Theorem 4.2.5, with respect to a suitable notion of morphism of type-categories. To go further and develop a correspondence between dependently typed algebraic theories and type-categories along the lines of Section 4.3, one has to restrict to the subclass of 'reachable' type-categories. For note that an object X in a type-category $\mathcal{C}$ can appear as the interpretation of a context only if for some $n \geq 0$ there is a sequence of indexed types

$$A_1 \in \mathit{Type}_{\mathcal{C}}(1), A_2 \in \mathit{Type}_{\mathcal{C}}(1 \ltimes A_1), \ldots, A_n \in \mathit{Type}_{\mathcal{C}}(1 \ltimes A_1 \ltimes \cdots \ltimes A_{n-1})$$

with $X = 1 \ltimes A_1 \ltimes \cdots \ltimes A_n$. Call $\mathcal{C}$ *reachable* if there is such a sequence for every $\mathcal{C}$-object. Every classifying category has this property. Conversely if $\mathcal{C}$ is reachable, then it is equivalent to $\mathcal{C}\ell(\mathit{Th})$ for a theory in a suitable 'internal language' for $\mathcal{C}$ (cf. Section 2.3).

6.5 Dependent products

The categorical framework corresponding to dependently typed equational logic that we have given can be used to infer the categorical structure needed for various dependently typed datatype constructors—much as we did for simple types in Section 3. We will just consider one example here, namely the structure in a type-category corresponding to dependent products. The formation, introduction, elimination, and equality rules for dependent product types are given in Fig. 12. Equality rules for both β- and η-conversion are given. There are also congruence rules for the constructors Π, λ, and ap, which we omit (cf. Remark 3.0.3).

If $\mathcal{C}$ is a type-category, the existence of the pullback squares (6.5) allows us to define a functor between slice categories

$$\pi_A^* : \mathcal{C}/X \longrightarrow \mathcal{C}/(X \ltimes A) \tag{6.7}$$

given by pullback along $\pi_A : X \ltimes A \longrightarrow X$:

$$\frac{\sigma \ \mathit{type}\ [\Gamma] \quad \sigma'(x)\ \mathit{type}\ [\Gamma, x : \sigma]}{\Pi(\sigma, \sigma')\ \mathit{type}\ [\Gamma]}$$

$$\frac{M'(x) : \sigma'(x)\ [\Gamma, x : \sigma]}{\lambda(M') : \Pi(\sigma, \sigma')\ [\Gamma]}$$

$$\frac{N : \Pi(\sigma, \sigma')\ [\Gamma] \quad M : \sigma\ [\Gamma]}{\mathsf{ap}(N, M) : \sigma'(M)\ [\Gamma]}$$

$$\frac{M'(x) : \sigma'(x)\ [\Gamma, x : \sigma] \quad M : \sigma\ [\Gamma]}{\mathsf{ap}(\lambda(M'), M) = M'(M) : \sigma'(M)\ [\Gamma]}$$

$$\frac{N : \Pi(\sigma, \sigma')\ [\Gamma]}{N = \lambda((x)\mathsf{ap}(N, x)) : \Pi(\sigma, \sigma')\ [\Gamma]}$$

Fig. 12. Rules for dependent product types

$$\pi_A^* \left(f \xrightarrow{g} f'\right) \ \stackrel{\text{def}}{=}\ \left(f \ltimes A \xrightarrow{\langle g \circ \pi_{f^* A}, f \ltimes A\rangle} f' \ltimes A\right) \quad .$$

(This definition makes use of the pairing notation introduced in 6.4.3.) Symmetrically, there is a functor

$$f^* : \mathit{Type}_{\mathcal{C}}(X) \longrightarrow \mathit{Type}_{\mathcal{C}}(Y) \tag{6.8}$$

induced by pullback along $f : Y \longrightarrow X$:

$$f^* \left(A \xrightarrow{g} A'\right) \ \stackrel{\text{def}}{=}\ \left(f^* A \xrightarrow{\langle \pi_{f^* A}, g \circ (f \ltimes A)\rangle} f^* A'\right) \quad .$$

Definition 6.5.1. A type-category $\mathcal{C}$ has *dependent products* if for each $X \in \mathit{Obj}_{\mathcal{C}}$, $A \in \mathit{Type}_{\mathcal{C}}(X)$, and $A' \in \mathit{Type}_{\mathcal{C}}(X \ltimes A)$, there is an indexed type

$$\Pi(A, A') \in \mathit{Type}_{\mathcal{C}}(X) \tag{6.9}$$

and a morphism in $\mathit{Type}_{\mathcal{C}}(X \ltimes A)$

$$\mathit{ap}_{A,A'} : \pi_A^* \Pi(A, A') \longrightarrow A' \tag{6.10}$$

satisfying:

- *Adjointness Property.* $\pi_{\Pi(A,A')} : X \ltimes \Pi(A,A') \longrightarrow X$ is the value of the right adjoint to the pullback functor (6.7) at $\pi_{A'} : X \ltimes A \ltimes A' \longrightarrow X \ltimes A$, with counit $ap_{A,A'}$. In other words, for any $f : Y \longrightarrow X$ in $\mathcal{C}$ and $g : \pi_A^*(f) \longrightarrow \pi_{A'}$ in $\mathcal{C}/(X \ltimes A)$, there is a unique morphism in $\mathcal{C}/X$
$$cur(g) : f \longrightarrow \pi_{\Pi(A,A')}$$
satisfying $ap_{A,A'} \circ \pi_A^*(cur(g)) = g$.
- *Strictness Property.* For any morphism $f : Y \longrightarrow X$ in $\mathcal{C}$:
$$\begin{aligned} f^*\Pi(A,A') &= \Pi(f^*A, (f \ltimes A)^*A') \\ (f \ltimes A)^*(ap_{A,A'}) &= ap_{f^*A,(f\ltimes A)^*A'} \end{aligned}$$
Here f^* and $(f \ltimes A)^*$ are instances of the pullback functors (6.8).

To specify the categorical semantics of dependent product types in such a type-category, we give rules extending the inductively defined relation of Section 6.4.

Formation

$$\frac{[\![\sigma\ type\ [\Gamma]]\!] \asymp A\ [X] \quad [\![\sigma'(x)\ type\ [\Gamma, x : \sigma]]\!] \asymp A'\ [X \ltimes A]}{[\![\Pi(\sigma,\sigma')\ type\ [\Gamma]]\!] \asymp \Pi(A,A')\ [X]} \tag{6.11}$$

Introduction

$$\frac{\begin{array}{c}[\![\sigma\ type\ [\Gamma]]\!] \asymp A\ [X] \quad [\![\sigma'(x)\ type\ [\Gamma, x : \sigma]]\!] \asymp A'\ [X \ltimes A] \\ [\![M'(x) : \sigma'(x)\ [\Gamma, x : \sigma]]\!] \asymp a' : A'\ [X \ltimes A]\end{array}}{[\![\lambda(M') : \Pi(\sigma,\sigma')\ [\Gamma]]\!] \asymp cur(a') : \Pi(A,A')\ [X]} \tag{6.12}$$

Note that the conclusion of this rule is well-formed if the hypotheses are. For if $a' \in_{X\ltimes A} A'$, then $a' : \pi_A^*(id_X) = id_{X\ltimes A} \longrightarrow \pi_{A'}$ in $\mathcal{C}/X \ltimes A$; so we can apply the *Adjointness Property* from Definition 6.5.1 to form $cur(a') : id_X \longrightarrow \pi_{\Pi(A,A')}$, which is in particular a global section of $\Pi(A,A')$.

Elimination

$$\frac{\begin{array}{c}[\![\sigma\ type\ [\Gamma]]\!] \asymp A\ [X] \quad [\![\sigma'(x)\ type\ [\Gamma, x : \sigma]]\!] \asymp A'\ [X \ltimes A] \\ [\![N : \Pi(\sigma,\sigma')\ [\Gamma]]\!] \asymp b : \Pi(A,A')\ [X] \quad [\![M : \sigma\ [\Gamma]]\!] \asymp a : A\ [X]\end{array}}{[\![\mathsf{ap}(N,M) : \sigma'(M)\ [\Gamma]]\!] \asymp \langle id_X\ ,\ ap_{A,A'} \circ \langle a,b\rangle\rangle : a^*A'\ [X]} \tag{6.13}$$

If the hypotheses of this rule are well-formed, then $a \in_X A$ and $b \in_X \Pi(A,A')$, i.e. $a : id_X \longrightarrow \pi_A$ and $b : id_X \longrightarrow \pi_{\Pi(A,A')}$ in $\mathcal{C}/X$. Since

$\pi_A \circ a = id_X = \pi_{\Pi(A,A')} \circ b$, we can use the pairing operation of 6.4.3 to form

$$\langle a, b\rangle : X \longrightarrow X \ltimes A \ltimes \pi_A^* \Pi(A, A') \ .$$

Since by definition $\pi_{\pi_A^* \Pi(A,A')} \circ \langle a, b\rangle = a$, we get a morphism $\langle a, b\rangle : a \longrightarrow \pi_{\pi_A^* \Pi(A,A')}$ in $\mathcal{C}/(X \ltimes A)$ and hence by composition a morphism $ap_{A,A'} \circ \langle a, b\rangle : a \longrightarrow \pi_{A'}$. Since $A \circ id_X = a = \pi_{A'} \circ ap_{A,A'} \circ \langle a, b\rangle$, the pairing operation yields a morphism

$$\langle id_X \ , \ ap_{A,A'} \circ \langle a, b\rangle\rangle : X \longrightarrow X \ltimes a^* A$$

whose composition with $\pi_{a^* A'}$ is id_X and hence which is a global section of $a^* A'$. Thus the conclusion of the above rule is well-formed when its hypotheses are.

With these rules we can extend Theorem 6.4.5 to include dependent product types.

Theorem 6.5.2 (Soundness). *Defining satisfaction of judgements as in Section 6.4, it is the case that the collection of judgements satisfied by a structure in a type-category with dependent products is closed under the rules in Fig. 12 (as well as the general rules of Fig. 9).*

Thus the structure of Definition 6.5.1 is sufficient for soundly interpreting dependent products. On the other hand it is necessary, in the sense that Proposition 6.3.5 can be extended as follows.

Proposition 6.5.3. *If Th is a theory in dependently typed equational logic with dependent product types, then the classifying category $\mathcal{C}\ell(Th)$ constructed in Section 6.2 is a type-category with dependent products.*

Proof. Referring to Definition 6.5.1, to define (6.9) find judgements

$$\Gamma \ ctxt \qquad \sigma \ type \ [\Gamma] \qquad \sigma'(x) \ type \ [\Gamma, x : \sigma]$$

that are theorems of *Th* and such that X, A, and A' are the equivalence classes of Γ, σ, and $\sigma'(x)$ respectively. Then take $\Pi(A, A')$ to be the Γ-indexed type determined by $\Pi(\sigma, \sigma')$. This definition is independent of the choice of representatives. The morphism (6.10) is that represented by the *Th*-context morphism

$$[id_\Gamma, x, \mathsf{ap}(z, x)] : [\Gamma, x : \sigma, z : \Pi(\sigma, \sigma')] \longrightarrow [\Gamma, x : \sigma, x' : \sigma(x)] \ .$$

Since $\pi_{\sigma'(x)} \circ [id_\Gamma, x, \mathsf{ap}(z, x)] = id_{[\Gamma, x:\sigma]}$ this does indeed induce a morphism of the required kind; and once again the definition of $ap_{A,A'}$ is independent of the choice of representatives.

The *Adjointness Property* required for $\Pi(A, A')$ can be deduced from the equality rules in Fig. 12, and the *Stability Property* follows from standard properties of substitution. ∎

7 Further Reading

This section lists some important topics in categorical logic which have not been covered in this chapter and gives pointers to the literature on them.

Higher order logic The study of *toposes* (a categorical abstraction of key properties of categories of set-valued sheaves) and their relationship to set theory and higher order logic has been one of the greatest stimuli of the development of a categorical approach to logic. [Mac Lane and Moerdijk, 1992] provides a very good introduction to topos theory from a mathematical perspective; [Lambek and Scott, 1986, Part II] and [Bell, 1988] give accounts emphasizing the connections with logic. The hyperdoctrine approach to first order logic outlined in Section 5 can be extended to higher order logic and such higher order hyperdoctrines can be used to generate toposes: see [Hyland *et al.*, 1980], [Pitts, 1981], [Hyland and Ong, 1993].

Polymorphic lambda calculus Hyperdoctrines have also been used successfully to model type theories involving type variables and quantification over types. [Crole, 1993, Chps 5 and 6] provides an introduction to the categorical semantics of such type theories very much in the spirit of this chapter.

Categories of relations Much of the development of categorical logic has been stimulated by a process of abstraction from the properties of categories of sets and *functions*. However, categorical properties of sets and *binary relations* (under the usual operation of composition of relations) have also been influential. The book by Freyd and Scedrov [1990] provides a wealth of material on categorical logic from this perspective.

Categorical proof theory Both Lawvere [1969] and Lambek [1968] put forward the idea that proofs of logical entailment between propositions, $\phi \vdash \psi$, may be modelled by morphisms, $[\![\phi]\!] \longrightarrow [\![\psi]\!]$, in categories. If one is only interested in the existence of such proofs, then one might as well only consider categories with at most one morphism between any pair of objects, i.e. only consider pre-ordered sets. This is the point of view taken in Section 5. However, to study the structure of proofs one must consider categories rather than just pre-orders. Lambek in particular has studied the connection between this categorical view of proofs and the two main styles of proof introduced by Gentzen (Natural Deduction and Sequent Calculus), introducing the notion of *multicategory* to model sequents in which more than one proposition occurs on either side of the turnstile, $\vdash$: see [Lambek, 1989]. This has resulted in applications of proof theory to category theory, for example in the use of cut elimination theorems to prove *coherence* results: see [Minc, 1977], [Mac Lane, 1982]. In the reverse direction of applications of category theory to proof theory, general categorical machinery, particularly enriched category theory, has provided useful guidelines for what constitutes a model of proofs in Girard's Linear Logic (see [Seely, 1989], [Barr, 1991], [Mackie *et al.*, 1993], [Bierman, 1994]); for example in [Benton *et al.*, 1993], such considerations facilitated the discovery of a well-behaved Natural Deduction formulation of intuitionistic linear logic.

Categorical combinators The essentially algebraic nature of the category theory corresponding to various kinds of logic and type theory gives rise to var¡iable-free, combinatory presentations of such systems. These have been used as the basis of abstract machines for expression evaluation and type checking: see [Curien, 1993], [Ritter, 1992].

References

[Backhouse *et al.*, 1989] R. Backhouse, P. Chisholm, G. Malcolm, and E. Saaman. Do-it-yourself type theory. *Formal Aspects of Computing*, 1:19–84, 1989.

[Barr and Wells, 1990] M. Barr and C. Wells. *Category Theory for Computing Science.* Prentice Hall, 1990.

[Barr, 1991] M. Barr. $*$-autonomous categories and linear logic. *Math. Structures in Computer Science*, 1:159–178, 1991.

[Bell, 1988] J. L. Bell. *Toposes and Local Set Theories. An Introduction*, volume 14 of *Oxford Logic Guides.* Oxford University Press, 1988.

[Benton *et al.*, 1993] P. N. Benton, G. M. Bierman, V. C. V. de Paiva, and J. M. E. Hyland. A term calculus for intuitionistic linear logic. In Bezem and Groote [1993], pages 75–90.

[Bezem and Groote, 1993] M. Bezem and J. F. Groote, editors. *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1993.

[Bierman, 1994] G. M. Bierman. *On Intuitionistic Linear Logic.* PhD thesis, Cambridge Univ., 1994.

[Bloom and Ésik, 1993] S. L. Bloom and Z. Ésik. *Iteration Theories.* EATCS Monographs on Theoretical Computer Science. Springer-Verlag, New York, 1993.

[Cartmell, 1986] J. Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.

[Chang and Keisler, 1973] C. C. Chang and H. J. Keisler. *Model Theory.* North-Holland, Amsterdam, 1973.

[Coste, 1979] M. Coste. Localisation, spectra and sheaf representation. In M. P. Fourman, C. J. Mulvey, and D. S. Scott, editors, *Applications of Sheaves*, volume 753 of *Lecture Notes in Mathematics*, pages 212–238. Springer-Verlag, Berlin, 1979.

[Crole and Pitts, 1992] R. L. Crole and A. M. Pitts. New foundations for fixpoint computations: Fix-hyperdoctrines and the fix-logic. *Information and Computation*, 98:171–210, 1992.

[Crole, 1993] R. L. Crole. *Categories for Types.* Cambridge Univ. Press, 1993.

[Curien, 1989] P.-L. Curien. Alpha-conversion, conditions on variables and categorical logic. *Studia Logica*, 48:319–360, 1989.

[Curien, 1990] P.-L. Curien. Substitution up to isomorphism. Technical Report LIENS-90-9, Laboratoire d'Informatique, Ecole Normale Supérieure, Paris, 1990.

[Curien, 1993] P.-L. Curien. *Categorical Combinators, Sequential Algorithms, and Functional Programming.* Birkhäuser, Boston, 1993. Second Edition.

[Dummett, 1977] M. Dummett. *Elements of Intuitionism.* Oxford University Press, 1977.

[Ehrhard, 1988] Th. Ehrhard. A categorical semantics of Constructions. In *3rd Annual Symposium on Logic in Computer Science*, pages 264–273. IEEE Computer Society Press, Washington, 1988.

[Freyd and Scedrov, 1990] P. J. Freyd and A. Scedrov. *Categories, Allegories.* North-Holland, Amsterdam, 1990.

[Freyd, 1972] P. J. Freyd. Aspects of topoi. *Bull. Austral. Math. Soc.*, 7:1–76 and 467–80, 1972.

[Girard, 1986] J.-Y. Girard. The system f of variable types, fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[Girard, 1987] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Girard, 1989] J.-Y. Girard. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1989. Translated and with appendices by Paul Taylor and Yves Lafont.

[Goguen and Meseguer, 1985] J. A. Goguen and J. Meseguer. Initiality, induction and computability. In Nivat and Reynolds [1985], pages 459–541.

[Goguen *et al.*, 1977] J. A. Goguen, J. W. Thatcher, E. G. Wagner, and J. B. Wright. Initial algebra semantics and continuous algebras. *JACM*, 24:68–95, 1977.

[Gray and Scedrov, 1989] J. W. Gray and A. Scedrov, editors. *Categories in Computer Science and Logic*, volume 92 of *Contemporary Mathematics.* Amer. Math. Soc, Providence RI, 1989.

[Hofmann, 1995] M. Hofmann. On the interpretation of type theory in locally cartesian closed cateories. In J. Tiuryn, editor, *Computer Science Logic, Kazimierz, Poland, 1994*, Lecture Notes in Computer Science. Springer-Verlag, Berlin, 1995.

[Hyland and Ong, 1993] J. M. E. Hyland and C.-H. L. Ong. Modified realizability toposes and strong normalization proofs. In Bezem and Groote [1993], pages 179–194.

[Hyland and Pitts, 1989] J. M. E. Hyland and A. M. Pitts. The theory of constructions: categorical semantics and topos-theoretic models. In

Gray and Scedrov [1989], pages 137–199.

[Hyland *et al.*, 1980] J. M. E. Hyland, P. T. Johnstone, and A. M. Pitts. Tripos theory. *Math. Proc. Cambridge Philos. Soc.*, 88:205–232, 1980.

[Hyland, 1988] J. M. E. Hyland. A small complete category. *Annals of Pure and Applied Logic*, 40:135–165, 1988.

[Jacobs and Melham, 1993] B. Jacobs and T. Melham. Translating dependent type theory into higher order logic. In Bezem and Groote [1993], pages 209–229.

[Jacobs, 1991] B. Jacobs. *Categorical Type Theory.* PhD thesis, Univ. Nijmegen, 1991.

[Lambek and Scott, 1986] J. Lambek and P. J. Scott. *Introduction to Higher Order Categorical Logic*, volume 7 of *Cambridge studies in advanced mathematics.* Cambridge Univ. Press, 1986.

[Lambek, 1968] J. Lambek. Deductive systems and categories i. *Math. Systems Theory*, 2:287–318, 1968.

[Lambek, 1988] J. Lambek. On the unity of algebra and logic. In F. Borceux, editor, *Categorical Algebra and its Applications*, volume 1348 of *Lecture Notes in Mathematics.* Springer-Verlag, Berlin, 1988.

[Lambek, 1989] J. Lambek. Multicategories revisited. In Gray and Scedrov [1989], pages 217–239.

[Lawvere, 1964] F. W. Lawvere. An elementary theory of the category of sets. *Proc. Nat. Acad. Sci. USA*, 52:1506–1511, 1964.

[Lawvere, 1966] F. W. Lawvere. The category of categories as a foundation for mathematics. In S. Eilenberg *et al*, editor, *Proc. Conference on Categorical Algebra, La Jolla*, pages 1–21, 1966.

[Lawvere, 1969] F. W. Lawvere. Adjointness in foundations. *Dialectica*, 23:281–296, 1969.

[Lawvere, 1970] F. W. Lawvere. Equality in hyperdoctrines and the comprehension schema as an adjoint functor. In A. Heller, editor, *Applications of Categorical Algebra*, pages 1–14. American Mathematical Society, Providence RI, 1970.

[Mac Lane and Moerdijk, 1992] S. Mac Lane and I. Moerdijk. *Sheaves in geometry and logic. A first introduction to topos theory.* Universitext. Springer-Verlag, Berlin, 1992.

[Mac Lane, 1971] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics.* Springer-Verlag, Berlin, 1971.

[Mac Lane, 1982] S. Mac Lane. Why commutative diagrams coincide with equivalent proofs. *Contemporary Mathematics*, 13:387–401, 1982.

[Mackie *et al.*, 1993] I. Mackie, L. Román, and S. Abramsky. An internal language for autonomous categories. *Applied Categorical Structures*, 1:311–343, 1993.

[Makkai and Reyes, 1977] M. Makkai and G. E. Reyes. *First Order Categorical Logic*, volume 611 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1977.

[McLarty, 1992] C. McLarty. *Elementary Categories, Elementary Toposes*, volume 21 of *Oxford Logic Guides*. Oxford University Press, 1992.

[Minc, 1977] G. E. Minc. Closed categories and the theory of proofs. *Zap. Nauch. Sem. Leningrad. Otdel. Mat. Inst. im. V. A. Steklova (LOMI)*, 96:83–114,145, 1977. Russian, with English summary.

[Mitchell and Scott, 1989] J. C. Mitchell and P. J. Scott. Typed lambda models and cartesian closed categories (preliminary version). In Gray and Scedrov [1989], pages 301–316.

[Moggi, 1991] E. Moggi. Notions of computations and monads. *Information and Computation*, 93:55–92, 1991.

[Nivat and Reynolds, 1985] M. Nivat and J. C. Reynolds, editors. *Algebraic Methods in Semantics*. Cambridge University Press, 1985.

[Nordström *et al.*, 1990] B. Nordström, K. Petersson, and J. M. Smith. *Programming in Martin-Löf's Type Theory*, volume 7 of *International Series of Monographs on Computer Science*. Oxford University Press, 1990.

[Obtułowicz, 1989] A. Obtułowicz. Categorical and algebraic aspects of martin-löf type theory. *Studia Logica*, 48:299–318, 1989.

[Oles, 1985] F. J. Oles. Types algebras, functor categories and block structure. In Nivat and Reynolds [1985], chapter 15, pages 543–574.

[Pierce, 1991] B. C. Pierce. *Basic Category Theory for Computer Scientists*. MIT Press, 1991.

[Pitts, 1981] A. M. Pitts. *The Theory of Triposes*. PhD thesis, Cambridge Univ., 1981.

[Pitts, 1987] A. M. Pitts. Polymorphism is set theoretic, constructively. In D. H. Pitt, A. Poigné, and D. E. Rydeheard, editors, *Category Theory and Computer Science, Proc. Edinburgh 1987*, volume 283 of *Lecture Notes in Computer Science*, pages 12–39. Springer-Verlag, Berlin, 1987.

[Pitts, 1989] A. M. Pitts. Non-trivial power types can't be subtypes of polymorphic types. In *4th Annual Symposium on Logic in Computer Science*, pages 6–13. IEEE Computer Society Press, Washington, 1989.

[Reynolds and Plotkin, 1993] J. C. Reynolds and G. D. Plotkin. On functors expressible in the polymorphic typed lambda calculus. *Information and Computation*, 105:1–29, 1993.

[Reynolds, 1983] J. C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83*, pages 513–523. Elsevier B.V. (North-Holland), Amsterdam, 1983.

[Ritter, 1992] E. Ritter. *Categorical Abstract Machines for Higher-Order Typed Lambda Calculi*. PhD thesis, Cambridge Univ., 1992.

[Román, 1989] L. Román. Cartesian categories with natural numbers object. *Journal of Pure and Applied Algebra*, 58:267–278, 1989.

[Scott, 1980] D. S. Scott. Relating theories of the lambda calculus. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 403–450. Academic Press, London, 1980.

[Seely, 1984] R. A. G. Seely. Locally cartesian closed categories and type theories. *Math. Proc. Cambridge Philos. Soc.*, 95:33–48, 1984.

[Seely, 1987] R. A. G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *Journal of Symbolic Logic*, 52:969–989, 1987.

[Seely, 1989] R. A. G. Seely. Linear logic, $*$-autonomous categories and cofree algebras. In Gray and Scedrov [1989], pages 371–382.

[Streicher, 1989] Th. Streicher. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions.* PhD thesis, Univ. Passau, 1989. Tech. Report MIP-8913.

[Streicher, 1991] Th. Streicher. *Semantics of Type Theory.* Birkhäuser, 1991.

[Taylor, 1986] P. Taylor. *Recursive domains, indexed category theory and polymorphism.* PhD thesis, Cambridge Univ., 1986.

Subject Index

Notation Index