Universal Structure and a Categorical Framework for Type Theory

Makoto Takeyama

Doctor of Philosophy

The University of Edinburgh

February 1995

(Graduation date: May 1995)

Abstract

This thesis investigates the possibility of a computer checked language for categories with extra structure; the language is to describe objects and morphisms of those categories and to reason about them. We do so first by developing an abstract analysis of representability. This is followed by the investigation of a categorical framework for studying type theory. Our computer checked language therefore allows us to reason about the semantics of programming languages and models of logics.

In order to provide our computer checked language, we need to classify categories with extra structure. Traditionally, that has been done in terms of equational structure, or more generally, essentially algebraic structure. That has proved to be somewhat awkward, both conceptually and computationally, so we give an alternative development of categories with extra structure in terms of universal structure, universality being the most important and most central concept of category theory. This unifies many of the concepts of the category theory, in particular many of those of greatest interest to computer scientists, such as cartesian closed structure, fibrations with extra structure, limits, colimits, and natural numbers objects. We use our definition of universal structure to develop a computer checked language in the proof system LEGO. We further give an abstract development of universal structure by showing how the concept of fibrations with extra structure may be seen in terms of universal structure.

We continue by an investigation of the use of category theory within computer science. One of the principle uses, perhaps the deepest use, is to provide a semantics for type theory. We give a unified treatment of those categories with extra structure that are needed for the semantics of type theory, thus allowing us to use our computer checked language to reason about the semantics of programs, via use of the underlying type theory of a programming language. It also allow us a semantic study of logics, via use of their corresponding type theories as given by the Curry-Howard isomorphism. This provides what we call a categorical framework for type theory. It extends the relationship between typed lambda calculus and cartesian closed categories, allowing us, for instance, to account for dependent types, as they appear in some programming languages and proof systems such as LEGO, with its underlying type system the Extended Calculus of Constructions. Finally we illustrate our categorical framework with its universal structures by study of the fibration of "deliverables".

Acknowledgement

I would like to thank my supervisor, Rod Burstall, for his support and interest in both academic and non-academic aspects of my studies in Edinburgh.

I also thank the members of LEGO club; particularly James McKinna for his comment on the draft of thesis and his guidance as a senior member of the club, Randy Pollack for the LEGO system, and Martin Hofmann for numerous discussions on fibrations and type theory.

I am also grateful to Claudio Hermida and Bart Jacobs for their expertise on fibrations and encouragement.

My utmost acknowledgement is to John Power, who was my second supervisor in the last six months. Without his commitment as my supervisor, and his encouragement throughout the course of my study, this work would have never been composed as a thesis.

This work was supported by Edinburgh University Postgraduate Studentship and Overseas Research Studentship for the first three years of my study, and by Laboratory for the Foundations of Computer Science for the last six months.

Paul Taylor's `diagrams` TeX macro package was used to produce diagrams in the text.

Declaration

I declare that this thesis was composed by myself, and the work contained in it is my own except where otherwise stated.

Makoto Takeyama

Table of Contents

Chapter 1

Introduction

This thesis has two objectives. The first one is to provide category theorists with a computer checked language to describe morphisms of categories of their choice. The categories will in general have extra structure; we base our language on such structure. This supports the proofs of coherence theorems in particular, and any category theoretic proofs of a syntactic nature in general. The second goal, of more direct interest to computer science, is to provide what we might call a "categorical framework" for type theory, together with a formalisation of that framework on a computer; with our computer checked language of the first goal providing the formalisation.

Earlier work related to the first goal of computer checked language is that of Rydeheard and Burstall's book "Computational Category theory" [71], which formalises several category theoretic constructions in ML. There, several individual universal constructions are described. Here, we give a uniform treatment, giving a general recipe for how to formalise universal constructions. This is of interest not only to category theorists but also to some computer scientists as they attempt to build programming languages based upon categorical constructions. Notable examples are Curien *et. al.*'s Categorical Abstract Machine [18], Hagino's Categorical Programming Language [30], and Cockett and Spencer's CHARITY [17, 75].

There are a number of ways to systematically formalise category theory on a computer. One common approach is to classify structure on a category as equational structure or more generally as essentially algebraic structure. There are many accounts of this in the category theoretic literature, but for an account directed toward computer scientists, see the paper "Why Tricategories?" [65,66] by Power. This paper also clarifies the importance of such systematic classification; we shall come back to this point in explaining our second goal. That particular classification, however, is sometimes a little awkward as we shall explain in more detail below. So, we prefer to classify structure on categories in terms of universal structure, universality being the single most important concept in category theory.

Universality embodies the thinking of how an object behaves with respect to the other objects of the category. In general, universal structure on categories is given by the existence of objects/morphisms specified by a universal property; e.g., categories with products, with limits, with exponentials, and with a subobject classifier. Our objective is to provide uniform ways to construct / manipulate / reason about objects/morphisms in those categories.

There are several ways to express universal structure. One way is to provide operations and equations that express the universality; *i.e.,* universal structure may be treated as an instance of algebraic structure. These operations arbitrarily choose objects among all possible isomorphic copies. Although this algebraic approach is straightforward to incorporate in type theory as sets of available lemmas, it is not subtle enough in the following senses.

First, it does not reflect well the way we reason about universal structure. Elegant reasoning about universal structure is usually derived from direct consideration of the universal property involved, and not from equational reasoning by a chain of calculations. For instance, it is possible to specify a terminal object by operations and equations [21]. To do so, first one gives a nullary operation to provide the terminal object T. Then, one gives a unary operation which, to each object X of the category, provides an arrow from X to T. But then, to express the unicity of the arrow in terms of operations and equations, the usual approach [21] is to provide an operation that to each arrow $f\colon X \longrightarrow Y$ gives a commutative triangle, subject to operations forcing the commutative triangle to be given by f and the two arrows to T. Then, one must add an equation to make the arrow from T to itself given by the above unary operation be the identity. Obviously, only very rarely one does reason about terminal objects in this way; usually, one just uses the universal property.

Second, equational reasoning on arbitrarily complex entities (= objects and morphisms encoded in a type theory), rather than simple types and functions, is a specifically weak and awkward point in most computer proof assistant. These become particularly acute since in our context, we have to take explicit and precise care of coherent isomorphisms, which naturally arise from universal structure.

The following example clarifies both points. In a category with binary products, one speaks often of $A \times B \times C$. But formally, no such entity exists. One has $(A \times B) \times C$ and $A \times (B \times C)$: they are isomorphic, and satisfy the universal property one wants of $A \times B \times C$, but there is no operation explicitly defining the latter. When binary products are given by operations and equations, one may show the isomorphism between $(A \times B) \times C$ and $A \times (B \times C)$ by calculation

$$\begin{aligned}
&\langle\langle\pi_0, \pi_0\pi_1\rangle, \pi_1\pi_1\rangle\langle\pi_0\pi_0, \langle\pi_1\pi_0, \pi_1\rangle\rangle \\
&\qquad = \langle\langle\pi_0, \pi_0\pi_1\rangle\langle\pi_0\pi_0, \langle\pi_1\pi_0, \pi_1\rangle\rangle, \pi_1\pi_1\langle\pi_0\pi_0, \langle\pi_1\pi_0, \pi_1\rangle\rangle\rangle \\
&\qquad = \langle\langle\pi_0\langle\pi_0\pi_0, \langle\pi_1\pi_0, \pi_1\rangle\rangle, \pi_0\pi_1\langle\pi_0\pi_0, \langle\pi_1\pi_0, \pi_1\rangle\rangle\rangle, \pi_1\langle\pi_1\pi_0, \pi_1\rangle\rangle \\
&\qquad = \langle\langle\pi_0\pi_0, \pi_0\langle\pi_1\pi_0, \pi_1\rangle\rangle, \pi_1\rangle \\
&\qquad = \langle\langle\pi_0\pi_0, \pi_1\pi_0\rangle, \pi_1\rangle \\
&\qquad = \langle\langle\pi_0\pi_0, \pi_1\pi_0\rangle, \pi_1\rangle \\
&\qquad = \langle\langle\pi_0, \pi_1\rangle\pi_0, \pi_1\rangle \\
&\qquad = \langle\pi_0, \pi_1\rangle \\
&\qquad = \mathbf{id},
\end{aligned}$$

and similarly in the other way. But the natural reasoning is "they both classify triples of morphisms into A, B, and C, and hence are canonically isomorphic." More precisely, one finds that $(A \times B) \times C$ classifies a pair $((f, g), h)$ that consists of a pair (f, g) of morphisms and h, and that $(A \times (B \times C))$ classifies a pair $(f, (g, h))$ of f and (g, h). The equivalence of the two is immediate. One may argue that, if products of sets should be treated formally, this requires the same calculation as above showing that the two sets $\{((f, g), h) | f \colon X \longrightarrow A, g \colon X \longrightarrow B, h \colon X \longrightarrow C\}$ and $\{(f, (g, h)) | \cdots\}$ are isomorphic. However, our point is that we feel much more natural about, and are more capable of, such calculation with sets than with objects of a given category. Note that the situation is exactly the same when one tries formalisation in a type theory with computer proof assistant. The assistant is designed for reasoning about types and functions, not objects and morphisms in a given category.

Therefore we seek an alternative approach. Among the other categorical ways of expressing universal structure, we regard representability of certain functors into **Set** as most suitable. Operations that give required universal objects/morphisms correspond to chosen representations of those functors. We aim to exploit this fact without breaking it down to sets of equations on operations. In this way, universal structure retains a direct connection with its universal property, accommodating our natural thinking. Yet the uniform nature of representable functors allows a uniform treatment across different kinds of universality. Note that this is a little more general than classifying the concept of universal structure in terms of adjoints.

For an instance of how this approach works, the above example of binary products may be treated as an instance of a category $\mathcal{C}$ for which the functor $\mathcal{C}(-,A) \times \mathcal{C}(-,B) \colon \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ $(A, B \in \mathcal{C})$ is representable. A machine then may be told that $A \times B$ is a representing object for this functor, and that fact can readily be used to interchange a morphism into $A \times B$ with a pair of morphisms into A and B. The particular encoding of objects and that of the construction $A \times B$ is irrelevant and hidden from the user. This further leads to the use of an internal language for cartesian categories.

This example is typical; many more examples are discussed in Chapter 2, culminating in an abstract definition of "universal structure", which incorporates all examples we have in mind. In Chapter 3, we proceed to give a formalisation of universal structure in the proof system LEGO. We have chosen LEGO because we specifically seek a proof system to facilitate category theoretic proofs. So, LEGO is more appropriate than a general programming language such as ML. Of course, there are many possible proof systems we could have chosen; our specific choice of LEGO is based primarily upon convenience and familiarity. In Chapter 4, we develop our examples of universal structure to illustrate how sophisticated category theoretic concepts are expressible in those terms. In particular, we concentrate upon fibrations with extra structure as these are of particular interest to computer science. They are also more broadly used in categorical logic, algebraic topology, and elsewhere. That completes our study in this thesis of a computer checked language for category theory. We hope to further develop that work; see Chapter 8 for details.

We now turn to our second goal, that of providing a categorical framework for type theory together with a formalisation of that in a computer.

First, we put our use of the word "framework" into context and explain the relationship with the first goal. A canonical example of "framework" in studies of type theory and logics is the the Logical Framework [32,23]. Historically, we had diverse and *ad hoc* descriptions for various logics. To overcome this diversity, the Logical Framework provided a common basis for representing various logics in which it is claimed that one can perform unified study of them. Questions addressed by such unified study are the role of variable-binding, discharge of assumptions, context sensitive side conditions *etc.*, which are all notions central to various logics but difficult to capture without a common basis [23]. The Logical Framework has immediate application in attempting to develop computer-assisted tools for reasoning about various logics [23].

Therefore, there is an analogy between the first goal and the Logical Framework. Our definition of universal structure provides a common basis with which various extra structures on categories are studied in a unified manner. Computer-assisted treatment of those structure is our motivation. Also note that the three questions listed above are, semantically speaking, about introduction / elimination rules of logics, which in turn have a close connection with universality. In this sense, our approach to the first goal with universal structure aims to provide an abstract "categorical framework" for categories with extra structure. The term "categorical framework for type theory" signifies our goal of a unified study of type theory within such an abstract categorical framework, with a view to computer-checked formalisation.

A significant difference, however, between the Logical Framework and our framework is about what aspects of logic/type theory are studied in a unified manner. The Logical Framework is predominantly syntax-oriented. This is dictated by how it provides a common basis, *i.e.,* encoding the syntax of various logics in a common type theory. In contrast, our framework is for semantics of various type theories, as we shall explain now (also see Section 5.1).

As we mentioned earlier, category theory has occasionally been used directly in computer science to build programming languages. Much more common, however, is for category theoretic structure to arise as the semantics for type theory.

Type theory has been widely used as a foundation for computer science. There are two distinct ways in which type theory is used. On the one hand, generic and strong systems are used as formalised and more manageable versions of set theory, acting as an ambient environment. The Calculus of constructions and its variants are such systems. Our "objects of study" *e.g.*, programs, processes, or mathematical objects (in the case of formal proof development), are encoded or implemented in terms of type theory so that the implementation agrees with our informal understanding of objects. On the other hand, more specific systems tailored for each object of study are used to formally denote and manipulate them, reflecting our informal understanding of them. Various logics for concurrent processes are such examples.

The Logical Framework relates these two uses. Objects of study are specific object-level type systems. When one represents an object-level type system in an ambient type theory, the latter is used to implement the specification of the former.

One objection to these approaches is their heavily syntactic nature. When one studies an encoded object as a syntactic entity in type theory, its sheer complexity makes it difficult to concentrate on the essential properties of the original (informal) object without their being obscured by the incidental details that the syntax of type theory forces. In the Logical Framework approach, details of syntactic formulations of object logics may have inadvertent effects; *e.g.,* one of two logics that are essentially the same can be represented while the other cannot (*cf.* Section 5.1.12 of Gardner's thesis [23]).

This realization has led to the use of category theory and there have been many studies that exhibit categories with structure as the semantics for type theories. A canonical example is the correspondence between the simply typed lambda calculus seen as a type theory and cartesian closed categories [44]. More recently there have been much more sophisticated analyses, in particular, involving fibrations with extra structure. A readable reference is Bart Jacobs' thesis [37]. Others are Dusko Pavlovic's thesis [61], Seely's work on hyperdoctrines [72], and Hermida's work connecting logical relations with fibrations [34]. Of particular interest to us is Burstall and McKinna's work on deliverables [11,52]. For a brief general account of the relationship between type theory and categories with structure one may again see "Why Tricategories?" [65,66], which also explains the need for a categorical framework, although the term "categorical framework" is not explicitly used there. Here, we add considerably more detail to that outline, and the work in our previous chapters provides a computer checked formalism for the framework.

In all that work, there has been little unified detailed analysis of the relationship between type theory and categories with structure in general. The above cited papers either address specific instances or give only a sketch of the relationship. Detailed analysis is needed in view of providing a basis for formalisation. So, in Chapters 5 and 6, we give a detailed account of a general class of type theories and their relationship with categories with universal structure. This allows us to use the results of our previous chapters to reason about programming languages and their semantics. These two chapters are a modified version of our previous work [84]. In Chapter 7, for concreteness, we illustrate the fibration of deliverables and the universal structures associated with it. Deliverables form an important proof technique for checking of correctness of programs. They are studied in detail in McKinna's thesis [52] with accounts appearing in [11].

Finally, in Chapter 8, we outline our plans for further research.

In more detail, the thesis is organised as follows. Chapter 2 is directed towards Definition 2.6.3, that of universal structure; Section 2.2 defines the bicategory of modules. We capture the universality of structure by the representability of modules defined in Section 2.3. Section 2.4 lists common examples. The preservation of such structure by functors is examined in Section 2.5. This leads to Section 2.6, where we define universal structure. The remaining Sections 2.7 and 2.8 investigate change of variance and composition of universal operations.

Chapter 3 gives the current implementation in LEGO of our computer checked language. A comparison with the work of Rydeheard and Burstall is given in Section 3.2, followed by an introduction to the LEGO syntax and commands in Section 3.3. Section 3.4 address general implementation issues. The code itself is shown in Section 3.5, divided into the code common to all structures, for binary products, for terminal objects, for equalizers, for pullbacks, and finally for an example, the construction of equalizers from pullbacks and terminal objects.

Chapter 4 examines fibrations with extra structure as an example of sophisticated structure defined in terms of universality widely used in computer science. This also gives a purely categorical description of a particular structure used in the next two chapters. The chapter is put after the LEGO code since, although it is not difficult to apply our implementation to fibrations, we prefer to provide a basis for a systematic generalisation of our approach from ordinary categories to fibred categories. Section 4.1 explains this and the current limit of study. Sections 4.2 – 4.5 give the basic definitions and overall structure of the 2-category **Fib** of fibrations, including the correspondence with indexed categories. We describe the local smallness structure of fibrations in detail in Section 4.6. Using these, Sections 4.7 and 4.8 investigate fibred limits and products and coproducts indexed by base objects.

Chapter 5 starts the second part of our study. After clarifying the goal in Section 5.1, Section 5.2 gives a class of traditional type systems with dependent types; $\mathcal{G}A$ with no type-formation, λP1 with dependent products, $w\mathcal{M}L$ with dependent products and weak dependent coproducts, $\mathcal{M}L$ with dependent products and strong dependent coproducts. Section 5.3 defines a new type system $\mathcal{T}F$ that corresponds to an equational presentation of locally small fibrations with the appropriate products and coproducts indexed by base objects. This is needed to give the precise relationship between the traditional type systems and locally small fibrations.

Chapter 6 studies that relationship in terms of syntactic translations between the traditional type systems and $\mathcal{T}F$. Section 6.2 explains how local smallness captures the context comprehension of dependent type systems by defining translations from $\mathcal{G}A$ theories to $\mathcal{T}F$ theories. Section 6.3 shows that $\mathcal{T}F$ is a conservative extension of $\mathcal{G}A$. Dependent products are dealt with in Section 6.4. Section 6.5 examines the difference between functions in λP1 and morphisms in TF, defining a variant $\mathcal{T}F\iota$ of $\mathcal{T}F$ where these two notions coincide. Then Section 6.6 establishes the equivalence between wML and $\mathcal{T}F\iota$. This is extended in Section 6.7 to the equivalence between $\mathcal{M}L$ and $\mathcal{T}F\iota\sigma$, another variant of $\mathcal{T}F$.

Chapter 7 examines the fibrations $\mathrm{p}_{\mathbf{Del}_1}$ and $\mathrm{p}_{\mathbf{Del}_2}$ of first-order and second-order deliverables and their structure. This is a concrete example of "objects of study" in an ambient type theory, and we exemplify how one can gain clarity by moving away from syntax and concentrating on a semantic property. The basic syntactic definitions are given in Section 7.2. The rich categorical structure of $\mathrm{p}_{\mathbf{Del}_1}$ and $\mathrm{p}_{\mathbf{Del}_2}$ are given in Section 7.3, showing they support the type systems studied in Chapter 6 as object-level type systems within the ambient type theory. This analysis leads to the definition of abstract first-order and second-order deliverables in Section 7.4. By moving from syntactically defined deliverables to abstract ones, we can clarify the relationship between first- and second-order deliverables as a Kleisli construction in Section 7.5. Finally, Section 7.6 applies this result to give a categorical analysis of the syntactic "family construction" given by McKinna [52].

Chapter 8 describes possible directions of further work. One particular issue is a generalisation of our analysis for ordinary categories in Chapter 2 to that for fibred categories. One needs this to put the second part of our work more in line with the first part, truly achieving a "categorical framework for type theory". We indicate possible ways to solve this problem.

Chapter 2

Analysis of representability

2.1 Introduction

This chapter is devoted to Definition 2.6.3 of universal structures. We capture the universality of structure by the notion of representability of a module by a functor.

A central feature of this analysis is the distinction between the *existence* of universal structure and an *operation* that assigns universal structure; for instance the difference between a category having binary products and an operation that specifies a particular choice of binary products. Our definition of universal structure allows us to say precisely what should be asked to exist in a uniform way without choosing operations. So we concentrate on that difference throughout the chapter, and we will see it vividly in writing our LEGO code in Chapter 3.

For readers unfamiliar with the concept of representability, we give an informal explanation of our use of modules and their representability. We work with a category $\mathcal{C}$, diagrams of shape $\mathcal{I}$ in $\mathcal{C}$, and their limits. Let Diagram be the set $\mathrm{Ob}\,[\mathcal{I}, \mathcal{C}]$ of diagrams of shape $\mathcal{I}$ in $\mathcal{C}$ and fix $h \in \mathrm{Diagram}$.

Consider a functor $\mathrm{Cones}(-, h) \colon \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ that sends

- an object $C \in \mathcal{C}$ to the set $\mathrm{Cones}(C, h)$ of cones over h with vertex C,
- a morphism $f \colon C \longrightarrow B$ to the function $(-) \circ f \colon \mathrm{Cones}(B, h) \longrightarrow \mathrm{Cones}(C, h)$ given by composing f with each component of a cone.

The limit Lim h of the diagram h is a representing object of Cones$(-,h)$, *i.e.*, there is a natural bijection $\mathcal{C}(-,\mathrm{Lim}\ h) \cong \mathrm{Cones}(-,h)$. Naturality is needed to determine Lim h up to unique isomorphism.

The notion *module* becomes involved when we vary $h \in$ Diagram. Then, we have a family $(\mathrm{Cones}(-,h))_{h\in\mathrm{Diagram}}$ of functors from $\mathcal{C}^{\mathrm{op}}$ to **Set** and a family of representations $(\mathcal{C}(-,\mathrm{Lim}\ h) \cong \mathrm{Cones}(-,h))_{h\in\mathrm{Diagram}}$. This gives a function Lim$(-)$ from Diagram to $\mathrm{Ob}\,\mathcal{C}$. But we further want to express the uniformity with which Lim h varies as we vary h. So, we replace the set Diagram by the category $[\mathcal{I},\mathcal{C}]$ and extend the family $(\mathrm{Cones}(-,h))_{h\in\mathrm{Diagram}}$ to a functor $\mathrm{Cones}\colon \mathcal{C}^{\mathrm{op}} \times [\mathcal{I},\mathcal{C}] \longrightarrow \mathbf{Set}$. For each $C \in \mathcal{C}$ and $s\colon h \longrightarrow k$, the function $\mathrm{Cones}(C,s)\colon \mathrm{Cones}(C,h) \longrightarrow \mathrm{Cones}(C,k)$ is given by componentwise composition. A *module* is such a functor that takes one contravariant argument and one covariant argument, and we say that Cones is a module from $[\mathcal{I},\mathcal{C}]$ to $\mathcal{C}$. This determines Lim up to unique isomorphism as a functor from $[\mathcal{I},\mathcal{C}]$ to $\mathcal{C}$ with a natural isomorphism $\mathcal{C}(-,\mathrm{Lim}\ ?) \cong \mathrm{Cones}(-,?)$; in other words, "the functor Lim represents the module Cones". So, modules arise in our analysis as functorially parameterised families of functors into **Set**.

Our precise definition of universal structure requires delicate analysis, so we approach it gradually. First, we must define the notion of module from a category $\mathcal{A}$ to $\mathcal{C}$. We do this in Section 2.2. Modules do not quite form a category; what they do form is a mild generalisation of a category, called a *bicategory*. The reason they do not form a category is because their composition is only associative up to coherent isomorphism, not up to equality. The definition of bicategory accounts for that. So, we give a definition of bicategory and show that modules form a bicategory.

The reason we introduce modules is because they capture the notion of structure that may asked to be universal. More precisely, it is expressed by the notion of representability of a module by a functor, which is defined in Section 2.3. A list of common examples that appear widely in category theory used in computer science appears in Section 2.4.

Then in Section 2.5, having investigated the presence in a category of universal structures we investigate the preservation of such structure by functors, continuing the analysis of our leading examples. The analysis is technical but important, since we must often pass between categories with structure and we need that structure preserved. Then, in Section 2.6, we define the notion *universal structure*.

In Section 2.7 and 2.8 we investigate that part of the theory of modules that we need for our applications of universal structure. Section 2.7 deals with the problem of contravariance; this is needed to accommodate exponentials. Section 2.8 explain how composition of operations that represent modules can be expressed in terms of the original modules. These are important to fully develop our computer checked treatment of universal structure, though they are yet to be incorporated to our LEGO implementation in Chapter 3.

Finally, note a difference, from our view point, between the status of structure in a given category $\mathcal{C}$ and that in **Set**. Since we partly formalise our analysis in the next chapter by replacing set theory by type theory, we need to clarify the relationship between these two structures. In doing so, we work in an *external* way, *i.e.*, we think of $\mathcal{C}$ as an object formed within set theory. Our analysis externalises structure in $\mathcal{C}$, giving a systematic way to reduce it to structure in **Set**. Structure in **Set** is assumed to be given in *set theoretic* terms; similarly, in the next chapter, we use primitive type constructors for operations on types. It is not our goal here to treat structure in a given $\mathcal{C}$ and that in **Set** in the same way. For this purpose, an *internal* way of working, *i.e.*, regarding $\mathcal{C}$ or **Set** as a mathematical object satisfying certain (first-order) axioms, is more suited; this naturally leads to study of extra structure in terms of operations and equations, where the universal structure of some *metatheory* is all that is assumed, rather than that of **Set**. For related discussions, see Section V.5 "Internal Versus External" of *Sheaves in Geometry and Logic* [50] and Section I.6 "Foundations" of *Categories for the Working Mathematician* [49].

2.2 The bicategory Mod of modules

As in the Introduction, a universal structure on a category involves the representability of certain functors into **Set**. So, the following definition is central to us; for other basic category theory, however, we only refer to Mac Lane's text [49].

Definition 2.2.1. Given a category $\mathcal{C}$ and a functor $P\colon \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$, P is *representable* if there exists an object R in $\mathcal{C}$ together with a natural isomorphism $\rho\colon \mathcal{C}(-,R) \cong P$. Such a pair (R,ρ) is called a *representation* of P with *representing object* R. The element $\rho_R(\mathrm{id}_R)$ of set PR is called the *counit* of (R,ρ).

In our applications, these functors usually form a parameterised family and the parameters form a category. Consider, for example, binary products *i.e.* Example 2.4.1. The representing objects for this parametrised family are parametrised functorially if and only if the family is parametrised functorially, as we shall now explain.

Proposition 2.2.2. Given categories $\mathcal{A}$ and $\mathcal{C}$, and let $(P(-,A)\colon \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set})_{A\in\mathcal{A}}$ be a family of functors such that each component is representable, *i.e.*, for each A, there exists $R(A) \in \mathcal{C}$ together with a natural isomorphism $\phi_{-,A}\colon \mathcal{C}(-,R(A)) \cong P(-,A)$. If the family $(P(-,A))_{A\in\mathcal{A}}$ can be extended to a functor $P\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathbf{Set}$, then $(R(A))_{A\in\mathcal{A}}$ uniquely extends to a functor $R\colon \mathcal{A} \longrightarrow \mathcal{C}$ such that $(\phi_{-,A})_{A\in\mathcal{A}}$ becomes a natural transformation $\mathcal{C}(-,R?) \longrightarrow P(-,?)\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathbf{Set}$. Conversely, if $(R(A))_{A\in\mathcal{A}}$ extends to a functor, then the family $(P(-,A))_{A\in\mathcal{A}}$ extends to a functor $P\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathbf{Set}$. These processes are mutually inverse.

Proof In the forward direction, $R(A \xrightarrow{f} B)$ is obtained as follows.

$$\begin{array}{cl}
RA = RA & \\
\hline
1 \longrightarrow P(RA,A) & \cong \text{ ; by the Yoneda lemma} \\
\hline
1 \longrightarrow P(RA,B) & \downarrow P(RA,f)\circ - \\
\hline
RA \xrightarrow[Rf]{} RB & \cong \text{ ; Yoneda}
\end{array}$$

In the reverse direction, given $f\colon A \longrightarrow A'$ in $\mathcal{A}$ and $g\colon C' \longrightarrow C$ in $\mathcal{C}$, $P(g,f)\colon P(C,A) \longrightarrow P(C',A')$ is given by

$$P(C,A) \overset{\phi^{-1}_{C,A}}{\cong} \mathcal{C}(C,RA) \xrightarrow{\mathcal{C}(g,Rf)} \mathcal{C}(C',RA') \overset{\phi_{C',A'}}{\cong} P(C',A').$$

∎

We henceforth consider only functorial families of representations, and functoriality is built in to our terminology.

Definition 2.2.3. Given categories $\mathcal{A}$ and $\mathcal{C}$, a *module* P from $\mathcal{A}$ to $\mathcal{C}$, written $P\colon \mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$, is a functor $\mathcal{C}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathbf{Set}$. A morphism between two modules with the same domain and codomain is a natural transformation between them.

In the literature, modules are also called attributes, profunctors, distributors, and two-sided fibrations [4,5,45,80,77,79].

Note that a module P from, say $\mathcal{A} \times \mathcal{B}$ to $\mathcal{C}$, that is, a functor $P\colon \mathcal{C}^{\mathrm{op}} \times (\mathcal{A} \times \mathcal{B}) \longrightarrow \mathbf{Set}$ is different from a module $P'\colon \mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; (\mathcal{B}^{\mathrm{op}} \times \mathcal{C})$, that is, a functor $P'\colon (\mathcal{C} \times \mathcal{B}^{\mathrm{op}})^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathbf{Set}$, even if P and P' are the same under the usual identification of $\mathcal{C}^{\mathrm{op}} \times (\mathcal{A} \times \mathcal{B})$ with $(\mathcal{C} \times \mathcal{B}^{\mathrm{op}})^{\mathrm{op}} \times \mathcal{A}$. However, we often write modules in the form of functors into $\mathbf{Set}$ when confusion is unlikely. In particular, we often identify $P\colon \mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ with $\mathcal{C}^{\mathrm{op}} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{A}^{\mathrm{op}}$; the latter may be written P^{op} if necessary. This is similar to omitting $(\ )^{\mathrm{op}}$ from F^{op} in $F^{\mathrm{op}}\colon \mathcal{A}^{\mathrm{op}} \longrightarrow \mathcal{C}^{\mathrm{op}}$ for $F\colon \mathcal{A} \longrightarrow \mathcal{C}$.

As indicated by the notation $\mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$, modules can be regarded as morphisms and one can define composition of them. However, this composition satisfies associativity and unit laws only up to coherent isomorphism. A *bicategory* is a structure that accounts for this kind of composition, generalising the notion of category. After giving the definition of bicategory, we show that modules form a bicategory.

Definition 2.2.4. A bicategory $\mathcal{W}$ is determined by the following data satisfying the axioms listed below.

(i) a set $\mathrm{Ob}\,\mathcal{W}$ of *objects* of $\mathcal{W}$.

(ii) for each pair (A, B) of objects, a category $\mathcal{W}(A, B)$. An object f of $\mathcal{W}(A, B)$ is called a *1-cell* from A to B and written $f\colon A \longrightarrow B$. A morphism $s \in \mathcal{W}(A, B)(f, g)$ is called a *2-cell* and written $s\colon f \Longrightarrow g$ or $A \underset{g}{\overset{f}{\underset{s\Downarrow}{\rightrightarrows}}} B$.

(iii) for each triple (A, B, C) of objects, a *composition* functor $\mathrm{c}_{A,B,C}\colon \mathcal{W}(B, C) \times \mathcal{W}(A, B) \longrightarrow \mathcal{W}(A, C)$. For 1-cells $f\colon B \longrightarrow C$ and $g\colon A \longrightarrow B$, the composite $\mathrm{c}_{A,B,C}(f, g)$ is written fg or $f \circ g$. For 2-cells $A \underset{t\Downarrow}{\rightrightarrows} B \underset{s\Downarrow}{\rightrightarrows} C$, $\mathrm{c}_{A,B,C}(s, t)$ is written st or $s \circ t$.

(iv) for each object A, an object $\mathrm{I}_A \in \mathcal{W}(A, A)$, called the *identity* 1-cell of A.

(v) for each quadruple (A, B, C, D) of objects and 1-cells $f\colon C \longrightarrow D$, $g\colon B \longrightarrow C$, and $h\colon A \longrightarrow B$, an invertible 2-cell

$$(\alpha_{A,B,C,D})_{f,g,h}\colon f(gh) \Longrightarrow (fg)h,$$

natural in f, g, and h, called the *associativity isomorphism*.

(vi) for each pair (A, B) of objects and 1-cell $f\colon A \longrightarrow B$, invertible 2-cells $(l_{A,B})_f\colon \mathrm{I}_B f \Longrightarrow f$ and $(\mathrm{r}_{A,B})_f\colon f\mathrm{I}_A \Longrightarrow f$, called the *left identity* and *right* identity, respectively.

We omit the indices A, B in $l_{A,B}$ when they are specified by the domain and the codomain; *e.g.*, for 1-cell $f\colon A \longrightarrow B$, we write l_f for $(l_{A,B})_f\colon \mathrm{I}_B f \Longrightarrow f$. Similarly we omit the indices in $\alpha_{A,B,C,D}$, $\mathrm{r}_{A,B}$, *etc.*. The axioms for these data are

(i) for 1-cells $x\colon D \longrightarrow E$, $y\colon C \longrightarrow D$, $z\colon B \longrightarrow C$, and $w\colon A \longrightarrow B$,

$$\begin{tikzcd}
((x \circ y) \circ z) \circ w \arrow[rr, "\alpha_{x,y,z} \circ \mathrm{id}_w"] \arrow[d, "\alpha_{x\circ y,z,w}"'] & & (x \circ (y \circ z)) \circ w \arrow[d, "\alpha_{x,y\circ z,w}"] \\
(x \circ y) \circ (z \circ w) \arrow[dr, "\alpha_{x,y,z\circ w}"'] & & x \circ ((y \circ z) \circ w) \arrow[dl, "\mathrm{id}_x \circ \alpha_{y,z,w}"] \\
& x \circ (y \circ z) &
\end{tikzcd}$$

commutes in $\mathcal{W}(A, D)$.

(ii) for 1-cells $x\colon B \longrightarrow C$ and $y\colon A \longrightarrow B$,

$$\begin{tikzcd}
(x \circ \mathrm{I}_B) \circ y \arrow[r, "\alpha_{x,\mathrm{I},y}"] \arrow[dr, "\mathrm{r}_x \circ \mathrm{id}_y"'] & x \circ (\mathrm{I}_B \circ y) \arrow[d, "\mathrm{id}_x \circ l_y"] \\
& x \circ y
\end{tikzcd}$$

commutes in $\mathcal{W}(A, C)$.

To put this definition in context, a one object bicategory is exactly a monoidal category: the 1-cells of the one object category correspond to the objects of the monoidal category; and the composition of 1-cells of the bicategory corresponds to the tensor product.

A leading example of a bicategory is given by **Span**: an object of which is a set X, a 1-cell from X to Y is a diagram of the form $X \longleftarrow Z \longrightarrow Y$, and composition is given by using pullbacks. The composition is not strictly associative because pullbacks are only determined up to isomorphism.

Composition of modules is defined in terms of a special kind of colimit called *coend* [49,40]

Definition 2.2.5.
- For a functor $G\colon \mathcal{A}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathcal{B}$ and an object $X \in \mathcal{B}$, a *cowedge* $t\colon G \longrightarrow X$ with *vertex* X is a family $(t_A)_{A\in\mathcal{A}}$ of $\mathcal{B}$-morphisms $t_A\colon GAA \longrightarrow X$ such that, for every $f\colon C \longrightarrow B$ in $\mathcal{A}$,

$$\begin{tikzcd}
GBC \arrow[r, "GfC"] \arrow[d, "GBf"'] & GCC \arrow[d, "t_C"] \\
GBB \arrow[r, "t_B"'] & X
\end{tikzcd}$$

  commutes.
- The functor $\mathrm{Cowdg}(G, -)\colon \mathcal{B} \longrightarrow \mathbf{Set}$ sends X in $\mathcal{B}$ to the set $\mathrm{Cowdg}(G, X)$ of cowedges from G to X, and $f\colon X \longrightarrow Y$ in $\mathcal{B}$ to the map that sends $t \in \mathrm{Cowdg}(G, X)$ to $(ft_A)_{A\in\mathcal{A}} \in \mathrm{Cowdg}(G, Y)$.
- A coend $\int^{A\in\mathcal{A}} GAA$ of G in $\mathcal{B}$ is a representing object of $\mathrm{Cowdg}(G, -)$. The unit of the representation is written $\mu \in \mathrm{Cowdg}(G, \int_{A\in\mathcal{A}} GAA)$

More concretely, $\int^{A\in\mathcal{A}} GAA$ is the vertex of a cowedge $\mu\colon G \longrightarrow \int^{A\in\mathcal{A}} GAA$ such that, for any cowedge $t\colon G \longrightarrow X$, there is a unique morphism $f\colon \int^{A\in\mathcal{A}} GAA \longrightarrow X$ with $t_A = f\mu_A \quad (A \in \mathcal{A})$.

Let $\mu\colon G \longrightarrow \int^{A\in\mathcal{A}} GAA$ and $\nu\colon H \longrightarrow \int^{A\in\mathcal{A}} HAA$ be universal cowedges for $G, H\colon \mathcal{A}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathcal{B}$, respectively. For a natural transformation $t\colon G \longrightarrow H$, the universality of μ determines the unique morphism

$$\int^{A\in\mathcal{A}} t_{AA}\colon \int^{A\in\mathcal{A}} GAA \longrightarrow \int^{A\in\mathcal{A}} HAA$$

given by the cowedge $(\nu_A t_{AA})_{A\in\mathcal{A}}\colon G \longrightarrow \int^{A\in\mathcal{A}} HAA$.

Proposition 2.2.6. For a cocomplete category $\mathcal{B}$, every coend $\int^{A\in\mathcal{A}} GAA$ with small $\mathcal{A}$ exists.

Proof Given $G\colon \mathcal{A}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathcal{B}$ with small $\mathcal{A}$, define $\sigma\colon \coprod_{A,B\in\mathcal{A}} \coprod_{f\in\mathcal{A}(B,A)} GAB \longrightarrow \coprod_{A\in\mathcal{A}} GAA$ by

$$\begin{tikzcd}
\coprod\limits_{A,B\in\mathcal{A}} \coprod\limits_{f\in\mathcal{A}(B,A)} GAB \arrow[r, "\sigma"] & \coprod\limits_{A\in\mathcal{A}} GAA \\
GAB \arrow[u, "i_{A,B,f}"] \arrow[r, "GAf"'] & GAA \arrow[u, "i_A"']
\end{tikzcd}$$

where $i_{A,B,f}$ and i_A are injections. Similarly, define $\rho\colon \coprod_{A,B\in\mathcal{A}} \coprod_{f\in\mathcal{A}(B,A)} GAB \longrightarrow \coprod_{A\in\mathcal{A}} GAA$ by $\rho i_{A,B,f} = i_B GfB$. Then, it is routine to check that the coend $\int^{A\in\mathcal{A}} GAA$ is the coequalizer of σ and ρ:

$$\coprod_{A,B\in\mathcal{A}} \coprod_{f\in\mathcal{A}(B,A)} GAB \underset{\rho}{\overset{\sigma}{\rightrightarrows}} \coprod_{A\in\mathcal{A}} GAA \xrightarrow{\mu} \int^{A\in\mathcal{A}} GAA.$$

∎

Corollary 2.2.7. **Set** has all small coends. ∎

The dual concept to coend is end; given $G\colon \mathcal{A}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathcal{B}$, an end of G, written $\int_{A\in\mathcal{A}} GAA$, is the vertex of a universal wedge $\lambda_B\colon \int_{A\in\mathcal{A}} GAA \longrightarrow GBB$ with, for any $f\colon A \longrightarrow B$, $GAf \circ \lambda_A = GfB \circ \lambda_B$. As a special case, given $F, G\colon \mathcal{A} \longrightarrow \mathcal{B}$, the set of natural transformation $[\mathcal{A}, \mathcal{C}](F, G)$ is an end $\int_{A\in\mathcal{A}} \mathcal{B}(FA, GA)$ with evaluation maps as the universal wedge; see Mac Lane's text [49]. With this connection in mind, one can see that the following lemma is equivalent to the Yoneda lemma.

Lemma 2.2.8. (Yoneda lemma)

Given a functor $G\colon \mathcal{A} \longrightarrow \mathbf{Set}$, there is an isomorphism

$$\int^{A\in\mathcal{A}} \mathcal{A}(A,B) \times GA \cong GB$$

natural in $B \in \mathcal{A}$.

Proof The cowedge $\mu_A\colon \mathcal{A}(A,B) \times GA \longrightarrow GB$ obtained by transposing G_{AB}: $\mathcal{A}(A,B) \longrightarrow [GA, GB]$ exhibits GB as the coend $\int^{A\in\mathcal{A}} \mathcal{A}(A,B) \times GA$. ∎

Having defined coends, now we can define composition of modules.

Definition 2.2.9. For modules $P\colon \mathcal{B} \looparrowright \mathcal{C}$ and $Q\colon \mathcal{A} \looparrowright \mathcal{B}$, the *composite* module $P \circ Q\colon \mathcal{A} \looparrowright \mathcal{C}$ sends (C, A) in $\mathcal{C}^{\mathrm{op}} \times \mathcal{A}$ to $\int^{B\in\mathcal{B}} P(C,B) \times Q(B,A)$ and $(f,g)\colon (C,A) \longrightarrow (C',A')$ in $\mathcal{C}^{\mathrm{op}} \times \mathcal{A}$ to $\int^{B\in\mathcal{B}} P(f,B) \times Q(B,g)$.

Observe here that this composition is only defined up to isomorphism. Henceforth, we assume that we have a particular choice, given set theoretically, of coend and universal cowedges for each $P \circ Q$.

Proposition 2.2.10. The following data give the bicategory **Mod** of modules.

(i) an object $\mathcal{A}$ of **Mod** is a small category.

(ii) for categories $\mathcal{A}$ and $\mathcal{B}$, $\mathbf{Mod}(\mathcal{A},\mathcal{B})$ is the category of modules $\mathcal{A} \looparrowright \mathcal{B}$ and module morphisms.

(iii) for categories $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, the composition functor

$$\mathrm{c}_{\mathcal{A},\mathcal{B},\mathcal{C}}\colon \mathbf{Mod}(\mathcal{B},\mathcal{C}) \times \mathbf{Mod}(\mathcal{A},\mathcal{B}) \longrightarrow \mathbf{Mod}(\mathcal{A},\mathcal{C})$$

sends (P,Q) to $P \circ Q$, and (s,t) to $(\int^{B\in\mathcal{B}} s_{CB} \times t_{BA})_{C\in\mathcal{C},A\in\mathcal{A}}$.

(iv) for a category $\mathcal{A}$, the identity 1-cell $\mathrm{I}_{\mathcal{A}}\colon \mathcal{C} \looparrowright \mathcal{C}$ is the Hom-functor $\mathcal{C}(-,?)$: $\mathcal{C}^{\mathrm{op}} \times \mathcal{C} \longrightarrow \mathbf{Set}$.

(v) for modules $P: \mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{D}$, $Q: \mathcal{B} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$, and $R: \mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{B}$, the associativity isomorphism $\alpha_{P,Q,R}: (P \circ Q) \circ R \longrightarrow P \circ (Q \circ R)$ is given by the universal property of coends.

(vi) for a module $P: \mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{B}$, the identity isomorphisms l_P and r_P are given by Lemma 2.2.8; *e.g.*, $l_P: \int^{B \in \mathcal{B}} \mathcal{B}(-, B) \times P(B, ?) \cong P(-, ?)$.

∎

Checking the axioms of bicategories for **Mod** is a routine calculation using the Yoneda lemma; an example of such calculation is shown in the proof of the following proposition.

There are two canonical ways to embed **Cat** into **Mod**, regarding a module as a generalised functor (see [45]).

Proposition 2.2.11. Given $F: \mathcal{A} \longrightarrow \mathcal{C}$, let $F_*: \mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ be the module defined by $F_*(C, A) = \mathcal{C}(C, FA)$. Then, for $F, G: \mathcal{A} \longrightarrow \mathcal{C}$, one has a natural isomorphism $[\mathcal{A}, \mathcal{C}](F, G) \cong \mathbf{Mod}(\mathcal{A}, \mathcal{C})(F_*, G_*)$. Similarly, for $F^*: \mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{A}$ with $F^*(A, C) = \mathcal{C}(FA, C)$, one has $[\mathcal{A}, \mathcal{C}](F, G) \cong \mathbf{Mod}(\mathcal{C}, \mathcal{A})(F^*, G^*)$.

Proof

$$
\begin{aligned}
\mathbf{Mod}(\mathcal{A}, \mathcal{C})(F_*, G_*) &\equiv [\mathcal{C}^{\mathrm{op}} \times \mathcal{A}, \mathbf{Set}](\mathcal{C}(-, F?), \mathcal{C}(-, G?)) \\
&\cong [\mathcal{A}, [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]](\mathcal{C}(-, F?), \mathcal{C}(-, G?)) \\
&\cong \int_A [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](\mathcal{C}(-, FA), \mathcal{C}(-, GA)) \\
&\cong \int_A \mathcal{C}(FA, GA) \quad \text{; by the Yoneda lemma} \\
&\cong [\mathcal{A}, \mathcal{C}](F, G)
\end{aligned}
$$

Similarly for $[\mathcal{A}, \mathcal{C}](F, G) \cong \mathbf{Mod}(\mathcal{C}, \mathcal{A})(F^*, G^*)$. ∎

2.3 Representability of modules

The reason we introduced modules is that they capture our notion of universal structure, that is, existence of a parameterised family of representing objects for functors into **Set** (*cf.* Proposition 2.2.2). The following definition makes precise what we mean by the assertion that each component of a module seen as a family of functors is representable.

Definition 2.3.1. Given a module $P\colon \mathcal{A} \longrightarrow\!\!\!\!\!\!\mapsto \mathcal{C}$, a *right representation* of P is a pair (R, ρ) consisting a functor $R\colon \mathcal{A} \longrightarrow \mathcal{C}$ and a natural isomorphism $\rho\colon R_* \cong P$. Dually, a *left representation* of P is a right representation of P^{op}: $\mathcal{C}^{\mathrm{op}} \longrightarrow\!\!\!\!\!\!\mapsto \mathcal{A}^{\mathrm{op}}$, *i.e.,* a pair $(L\colon \mathcal{C} \longrightarrow \mathcal{A}, \lambda\colon L^* \cong P)$.

We often omit ρ, saying "a right representation $R\colon \mathcal{A} \longrightarrow \mathcal{C}$". Note that a right representation of a module $P\colon \mathcal{A} \longrightarrow\!\!\!\!\!\!\mapsto \mathcal{C}$ is different from a representation of a functor $P\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathbf{Set}$. In the literature, P is also said to *converge to* R [79].

Next, we define the notion of left Kan extension and left lifting and relate them to the above definition.

Definition 2.3.2. Given categories $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, a natural transformation ϕ

$$\begin{tikzcd} & \mathcal{C} \arrow[dr, "T"] \arrow[d, phantom, "\Uparrow \phi"] & \\ \mathcal{A} \arrow[ur, "K"] \arrow[rr, "G"'] & {} & \mathcal{B} \end{tikzcd}$$

is said to *exhibit* T *as a pointwise left Kan extension of* G *along* K if ϕ induces a natural isomorphism $\bar{\phi}$ as follows.

$$\cfrac{\cfrac{\phi_A\colon GA \longrightarrow TKA}{\mathcal{C}(KA, C) \longrightarrow \mathcal{B}(GA, TC)} \cong\text{(Yoneda)}}{\bar{\phi}_{C,B}\colon \mathcal{B}(TC, B) \cong [\mathcal{A}^{\mathrm{op}}, \mathbf{Set}](\mathcal{C}(K-, C), \mathcal{B}(G-, B))} \cong\text{(Yoneda)}$$

The natural isomorphism $\bar{\phi}$ shows that (T, ϕ) is uniquely determined by G and K up to isomorphism. We write $\mathrm{Lan}_K G$ for such a T.

Left Kan extensions play a central role in category theory. Their main application is as follows. If $\mathcal{A}$ and $\mathcal{B}$ are small categories and $H\colon \mathcal{A} \longrightarrow \mathcal{B}$ is a functor, then $[H, \mathbf{Set}]\colon [\mathcal{B}, \mathbf{Set}] \longrightarrow [\mathcal{A}, \mathbf{Set}]$ has a left adjoint and it is given by left Kan extension along H. More generally, we have:

Proposition 2.3.3. Given functors $G\colon \mathcal{A} \longrightarrow \mathcal{B}$, $K\colon \mathcal{A} \longrightarrow \mathcal{C}$, and $H\colon \mathcal{C} \longrightarrow \mathcal{B}$ with $\mathcal{A}, \mathcal{C}$ small and $\mathcal{B}$ cocomplete, pasting with the unit ϕ of $\mathrm{Lan}_K G$ induces a natural isomorphism

$$[\mathcal{C}, \mathcal{B}](\mathrm{Lan}_K G, H) \cong [\mathcal{A}, \mathcal{B}](G, HK), \qquad t \mapsto tK \cdot \phi\ .$$

Proof One can check the following isomorphism sends t to $tK \cdot \phi$.

$$\begin{aligned}
[\mathcal{C}, \mathcal{B}](\mathrm{Lan}_K G, H) &\cong \int_C \mathcal{B}(\mathrm{Lan}_K GC, HC) \\
&\cong \int_C [\mathcal{A}^{\mathrm{op}}, \mathbf{Set}](\mathcal{C}(K-, C), \mathcal{B}(G-, HC)) \quad \text{; by } \bar{\phi}_{C,HC} \\
&\cong \int_A \int_C [\mathcal{C}(KA, C), \mathcal{B}(GA, HC)] \\
&\cong \int_A \mathcal{B}(GA, HKA) \quad \text{; by Yoneda} \\
&\cong [\mathcal{A}, \mathcal{C}](G, HK)
\end{aligned}$$

∎

One dual (reversing the direction of functors) to left Kan extensions are left liftings, defined below. Right representations can be reformulated in terms of left liftings, which in turn connect them to left Kan extensions (Corollary 2.3.7).

Definition 2.3.4. Given $K\colon \mathcal{C} \longrightarrow \mathcal{B}$, $G\colon \mathcal{A} \longrightarrow \mathcal{B}$, and $T\colon \mathcal{A} \longrightarrow \mathcal{C}$, a natural transformation η

$$\begin{tikzcd}
& \mathcal{C} \arrow[dr, "K"] \arrow[d, phantom, "\Uparrow \eta"] & \\
\mathcal{A} \arrow[ur, "T"] \arrow[rr, "G"'] & {} & \mathcal{B}
\end{tikzcd}$$

is said to *exhibit* T as an *absolute left lifting* of G *through* K if η induces a natural isomorphism $\bar{\eta}$ by Yoneda:

$$\frac{\eta_A\colon GA \longrightarrow TKA}{\bar{\eta}_{A,C}\colon \mathcal{C}(TA, C) \cong \mathcal{B}(GA, KC).} \cong\text{(Yoneda)}$$

The functor T is determined by K and G up to isomorphism. We write $\mathrm{Lit}_K G$ for such T.

A special case of left liftings is an adjoint pair of functors $\mathcal{A} \underset{G}{\overset{F}{\underset{\perp}{\rightleftarrows}}} \mathcal{B}$ where the unit η of $F \dashv G$ exhibits F as $\mathrm{Lit}_{\mathrm{id}_{\mathcal{A}}} G$.

The dual to Proposition 2.3.3 is:

Proposition 2.3.5. Given functors $G \colon \mathcal{A} \longrightarrow \mathcal{B}$, $K \colon \mathcal{C} \longrightarrow \mathcal{B}$, and $H \colon \mathcal{A} \longrightarrow \mathcal{C}$, pasting with the unit η of $\mathrm{Lit}_K G$ induces a natural isomorphism

$$[\mathcal{A}, \mathcal{C}](\mathrm{Lit}_K G, H) \cong [\mathcal{A}, \mathcal{B}](G, KH), \qquad t \mapsto Kt \cdot \eta.$$

∎

Functors with the universal properties of Propositions 2.3.3 and 2.3.5 are called (not necessarily pointwise) left Kan extension of G along K and (not necessarily absolute) left lifting of G through K, respectively. These properties are weaker than Definitions 2.3.2 and 2.3.4 [40]. However, since we do not use these weaker notions, we omit the word ‘pointwise’ and ‘absolute’ in the following.

Proposition 2.3.6. Given a diagram

$$\begin{tikzcd} & \mathcal{B}^{\mathrm{op}} \times \mathcal{A} \arrow[dr, "P"] \arrow[d, phantom, "\Uparrow \phi"] & \\ \mathcal{A}^{\mathrm{op}} \times \mathcal{A} \arrow[ur, "K \times \mathrm{id}_{\mathcal{A}}"] \arrow[rr, "\mathcal{A}(-,?)"'] & & \mathbf{Set}, \end{tikzcd}$$

ϕ exhibits P as $\mathrm{Lan}_{K \times \mathrm{id}} \mathcal{A}(-,?)$ if and only if ϕ exhibits $K \times \mathrm{id}$ as $\mathrm{Lit}_P \mathcal{A}(-,?)$.

Proof Both statements amount to requiring that ϕ induces a natural isomorphism $\mathcal{B}(B, KA) \cong PBA$. ∎

Corollary 2.3.7. Given a module $P\colon \mathcal{A} \nrightarrow \mathcal{C}$, P has a right representation (R, ρ) if and only if the natural transformation $\eta_{A,B}\colon \mathcal{A}(A, B) \longrightarrow P(RA, B)$ defined by

$$\dfrac{\dfrac{\mathcal{C}(A, RB) \xrightarrow{\rho_{A,B}} PAB}{1 \longrightarrow P(RB, B)} \cong(\text{Yoneda})}{\mathcal{A}(A, B) \xrightarrow{\eta_{A,B}} P(RA, B)} \cong(\text{Yoneda})$$

exhibits P as a left Kan extension of $\mathcal{A}(-, ?)$ along $R \times \mathrm{id}_{\mathcal{A}}$:

$$\begin{tikzcd}
& \mathcal{C}^{\mathrm{op}} \times \mathcal{A} \arrow[dr, "P"] \arrow[d, phantom, "\Uparrow \eta"] & \\
\mathcal{A}^{\mathrm{op}} \times \mathcal{A} \arrow[ur, "R \times \mathrm{id}"] \arrow[rr, "\mathcal{A}(-,?)"'] & & \mathbf{Set}.
\end{tikzcd}$$

Proof By the definition of Lit and Propostion 2.3.6. ▮

2.4 Examples of representable modules

In this section, we introduce our leading examples of modules. The right or left representability of these modules defines categorical structures widely used in computer science.

Example 2.4.1. (Binary products)

Consider the right representation corresponding to a binary product functor on a category $\mathcal{C}$. Define the module $\mathrm{Pairs}\colon \mathcal{C} \times \mathcal{C} \nrightarrow \mathcal{C}$ by

$$\begin{array}{lccc}
\mathrm{Pairs} & : \mathcal{C}^{\mathrm{op}} \times (\mathcal{C} \times \mathcal{C}) & \longrightarrow & \mathbf{Set} \\
& (X; A, B) & \mapsto & \mathcal{C}(X, A) \times \mathcal{C}(X, B), \\
& (h; f, g) & \mapsto & \mathcal{C}(h, f) \times \mathcal{C}(h, g).
\end{array}$$

If Pairs admits a right representation

$$(-\times ?\colon \mathcal{C} \times \mathcal{C} \longrightarrow \mathcal{C},\ \widetilde{\times}_{X,(A,B)}\colon \mathcal{C}(X, A \times B) \cong \mathcal{C}(X, A) \times \mathcal{C}(X, B)),$$

then $A \times B$ is a binary product of A and B. The morphism part of $\times$ agrees with the usual definition of the product functor based on the universal property of $A \times B$. Conversely, if $\mathcal{C}$ has binary products, Pairs admits a right representation.

The two projections $\pi_0 \colon A \times B \longrightarrow A$ and $\pi_1 \colon A \times B \longrightarrow B$ are given by the counit of $\widetilde{\times}_{-,(A,B)}$, *i.e.*, the two components of $\widetilde{\times}_{A\times B,(A,B)}(\mathrm{id}_{A\times B})$.

Note that $\times$ in $\mathcal{C}$ is defined in terms of $\times$ in **Set** (as in $\mathcal{C}(X, A) \times \mathcal{C}(X, B)$). Here, the latter $\times$ is naively given in set theoretic terms.

Example 2.4.2. (Equalizer)

Another example of a limit construction is an equalizer. To give the category of parameters for equalizers, let **Para** be the category indicated by $\{\bullet \rightrightarrows \bullet\}$. The functor category $[\mathbf{Para}, \mathcal{C}]$ has as objects parallel pairs $A \overset{f}{\underset{g}{\rightrightarrows}} B$ of morphisms of $\mathcal{C}$. A morphism of $[\mathbf{Para}, \mathcal{C}]$ from $A \overset{f}{\underset{g}{\rightrightarrows}} B$ to $A' \overset{f'}{\underset{g'}{\rightrightarrows}} B'$ is a pair of morphisms of $\mathcal{C}$ $(A \xrightarrow{l} A', B \xrightarrow{m} B')$ that makes the following diagrams commute.

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "l"'] & B \arrow[d, "m"] \\ A' \arrow[r, "f'"'] & B' \end{tikzcd} \qquad \begin{tikzcd} A \arrow[r, "g"] \arrow[d, "l"'] & B \arrow[d, "m"] \\ A' \arrow[r, "g'"'] & B' \end{tikzcd}$$

The module $\mathrm{Ker} \colon [\mathbf{Para}, \mathcal{C}] \longrightarrow\!\!\!\!\!+\; \mathcal{C}^{\mathrm{op}}$ given by

$$\begin{array}{lll} \mathrm{Ker} & : \mathcal{C}^{\mathrm{op}} \times [\mathbf{Para}, \mathcal{C}] \longrightarrow \mathbf{Set} & \\ & \begin{tikzcd} X & A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] \arrow[d, "l"'] & B \arrow[d, "m"] \\ X' \arrow[u, "k"] & A' \arrow[r, shift left, "f'"] \arrow[r, shift right, "g'"'] & B' \end{tikzcd} \mapsto & \begin{array}{c} \{h \colon X \longrightarrow A \mid fh = gh\} \\ \downarrow h \mapsto lhk \\ \{h' \colon X' \longrightarrow A' \mid f'h' = g'h'\} \end{array} \end{array}$$

admits a right representation

$$\begin{aligned} &(\mathrm{Eq}(-,-) \colon [\mathbf{Para}, \mathcal{C}] \longrightarrow \mathcal{C}, \\ &\widetilde{\mathrm{Eq}}_{X;f,g} \colon \mathcal{C}(X, \mathrm{Eq}(f, g)) \cong \mathrm{Ker}(X; f, g)), \end{aligned}$$

if and only if $\mathcal{C}$ has equalizers of all parallel pairs, and then they are given by $\mathrm{Eq}(f, g)$. The universal morphism $\mathrm{Eq}(f, g) \longrightarrow A$ is given by the counit $\widetilde{\mathrm{Eq}}_{\mathrm{Eq}(f,g);f,g}(\mathrm{id}_{\mathrm{Eq}(f,g)})$, which is necessarily a mono since $\mathrm{Ker}(-; f, g)$ is a subfunctor of a representable.

The morphism part of Eq sends (l, m) to the unique k in the diagram

$$\begin{tikzcd}
\mathrm{Eq}(f,g) \arrow[r, tail] \arrow[d, dashed, "k"'] & A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] \arrow[d, "l"'] & B \arrow[d, "m"] \\
\mathrm{Eq}(f',g') \arrow[r, tail] & A' \arrow[r, shift left, "f'"] \arrow[r, shift right, "g'"'] & B'.
\end{tikzcd}$$

Note that

$$\mathrm{Ker}(X; f, g) \hookrightarrow \mathcal{C}(X, A) \underset{\mathcal{C}(X,g)}{\overset{\mathcal{C}(X,f)}{\rightrightarrows}} \mathcal{C}(X, B)$$

is the canonical equalizer diagram in **Set**; in other words, equalizers in $\mathcal{C}$ are defined by those in **Set**, similarly to the binary product case. This is a simple consequence of the fact that Yoneda functor $\mathcal{C}(?, -): \mathcal{C} \longrightarrow [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$ preserves all limits. This illustrates the more general situation that a concept in a category $\mathcal{C}$ internalises the corresponding, external concept in **Set**.

Example 2.4.3. (Subobject Classifier)

This is a non-limit example of a right representation. This differs from previous ones in that to talk about $\mathcal{C}$ having a classifier, one needs pullbacks of all monos along arbitrary maps in $\mathcal{C}$. In this example, $\mathcal{C}$ is assumed to have all such pullbacks.

A subobject of $A \in \mathcal{C}$ is typically defined to be an isomorphism class (in $\mathcal{C}/A$) of monomorphisms into A. This leads to an imprecise practice of treating subobjects as monomorphisms; operations on subobjects, for example, are defined on monomorphisms and checking well-definedness is often omitted.

So here, we modify the definition of subobjects by defining a subobject of A to be a representable sieve on A, *i.e.*, a representable subfunctor of $\mathcal{C}(-, A)$. This is, conceptually, to regard subobjects as an internalisation of (representable) predicates on all morphisms into A. It is routine to verify that this is equivalent to the usual definition. For the forward direction, choose any representation $\varphi: \mathcal{C}(-, B) \cong S$ of a representable subfunctor S of $\mathcal{C}(-, A)$. The counit $\varphi_B(\mathrm{id}_B) \in SB \subset \mathcal{C}(B, A)$ is necessarily a monomorphism, thus defining an isomorphism class. Conversely, choose any monomorphism $m: B \longrightarrow A$ in an isomorphism class and define a subfunctor S by

$$SC = \{f\colon C \longrightarrow A | \exists h\colon C \longrightarrow B. f = mh\}.$$

Then S is representable by $\mathcal{C}(-,B) \cong S-,\ h \mapsto mh$. In both ways, the constructions do not depend on the choice and they are mutually inverse.

The pullback $S[f]$ of a subobject S of A along a morphism $f\colon B \longrightarrow A$ is defined by the canonical pullback diagram of inclusions in $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$:

$$\begin{tikzcd} S[f] \arrow[r] \arrow[d, hook] \arrow[dr, phantom, "\lrcorner", very near start] & S \arrow[d, hook] \\ \mathcal{C}(-,B) \arrow[r, "\mathcal{C}(-,f)"] & \mathcal{C}(-,A). \end{tikzcd}$$

More concretely, $S[f](-) = \{h\colon - \longrightarrow B | fh \in S-\}$. For $\mathcal{C}$ to have pullbacks of monomorphisms along f is equivalent to saying that $S[f]$ is representable whenever S is.

With this definition, the module $\mathrm{Sub}\colon \mathbf{1} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ given by the functor

$$\begin{array}{rcccl} \mathrm{Sub} & : & \mathcal{C}^{\mathrm{op}} & \longrightarrow & \mathbf{Set} \\ & & A & & \{S | S \hookrightarrow \mathcal{C}(-,A) \text{ and representable.}\} \\ & & f \Big\uparrow & \mapsto & \Big\downarrow S \mapsto S[f] \\ & & B & & \{T | T \hookrightarrow \mathcal{C}(-,B) \text{ and representable.}\} \end{array}$$

admits a right representation

$$(\Omega \in \mathcal{C}, \widetilde{\Omega}\colon \mathcal{C}(-,\Omega) \cong \mathrm{Sub}(-)),$$

if and only if $\mathcal{C}$ has a subobject classifier Ω. The generic subobject of Ω as a representable subfunctor is given by the counit

$$\widetilde{\Omega}_\Omega(\mathrm{id}_\Omega)\colon T \hookrightarrow \mathcal{C}(-,\Omega) \in \mathrm{Sub}(\Omega),$$

for some representable T. One can show that, for each $X \in \mathcal{C}$, TX necessarily consists of one morphism $X \xrightarrow{\top_X} C$ for each $X \in \mathcal{C}$. Therefore, a representing object 1 for T is always a terminal object of $\mathcal{C}$ and the counit $1 \xrightarrow{\top_1} \Omega$ of this representation $\mathcal{C}(-,1) \cong T-$ gives a generic monomorphism.

Example 2.4.4. (Tensor products)

Some operations are representations of modules whose definitions require not only two categories $\mathcal{A}$ and $\mathcal{C}$, but some added data. For example, we consider tensor products. Suppose $\mathcal{C}$ is a concrete category, *i.e.*, equipped with a faithful functor $U\colon \mathcal{C} \longrightarrow \mathbf{Set}$. For A, B, and X in $\mathcal{C}$, define $\mathrm{Bilin}(A, B; X)$ by the following pullback diagram in $\mathbf{Set}$.

$$\begin{tikzcd}
\mathrm{Bilin}(A,B;X) \arrow[r, tail] \arrow[dd, tail] \arrow[dr, phantom, "\lrcorner", very near start] & {[UB,\mathcal{C}(A,X)]} \arrow[d, tail, "{[UB,U_{AX}]}"] \\
& {[UB,[UA,UX]]} \arrow[d, "\cong"] \\
{[UA,\mathcal{C}(B,X)]} \arrow[r, tail, "{[UA,U_{BX}]}"'] & {[UA,[UB,UX]]\cong[UA\times UB,UX]}
\end{tikzcd}$$

Because U is faithful, $\mathrm{Bilin}(A, B; X)$ consists of bilinear maps $UA \times UB \longrightarrow UX$, *i.e.*, those maps which, in each argument, are underlying functions of morphisms in $\mathcal{C}$. By the functoriality of the other three nodes, Bilin extends to a functor from $(\mathcal{C} \times \mathcal{C})^{\mathrm{op}} \times \mathcal{C}$ to $\mathbf{Set}$, *i.e.*, to a module $\mathrm{Bilin}\colon \mathcal{C} \longrightarrow\!\!\!\!\!\!\mapstochar\;\; \mathcal{C} \times \mathcal{C}$.

Let $\mathbf{SL}$ be the category of $\wedge$-semilattices with U the underlying functor. Then, the explicit construction of the tensor $A \otimes B$ gives a left representation $\otimes$ of Bilin

$$\otimes = (-\otimes ?\colon \mathbf{SL} \times \mathbf{SL} \longrightarrow \mathbf{SL},\ \tilde{\otimes}_{AB-}\colon \mathbf{SL}(A \otimes B, -) \cong \mathrm{Bilin}(A, B; -))$$

with $\tilde{\otimes}$ provided by the characterisation theorem. The usual way to treat $\otimes$ more abstractly is to regard it as a monoidal structure, *i.e.*, a bifunctor equipped with a few natural transformation subject to a few equations. One point of this work is to show that it is also a useful abstraction to regard it as a left representation with two arguments of a module that satisfies a condition.

2.5 Preservation of structure

Section 2.4 gave examples of universal structures on a category expressed as right/left representability of modules. However, a single module describes only an *instance* of a structure on a given category. The notion of structure corresponds to a uniform class of such instances. For example, the module $\mathrm{Pairs}(\mathcal{A}): \mathcal{A}\times\mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{A}$ whose representability gives $\mathcal{A}$ binary products structure is exactly the same as $\mathrm{Pairs}(\mathcal{B}): \mathcal{B}\times\mathcal{B} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{B}$ for $\mathcal{B}$ except, of course, that $\mathcal{A}$ is replaced by $\mathcal{B}$.

In this section, we show that this uniformity is implicitly used in the notion of structure preserving functors, and should be given as part of the data that define structure. We first recall the usual definitions of structure preserving functors, which are expressed in terms of particular representations.

Example 2.5.1. (Binary products)

For a category $\mathcal{A}$ with chosen binary products, a functor $G: \mathcal{A} \longrightarrow \mathcal{B}$ preserves binary products if G sends the chosen, and hence every, product cone for A and B in $\mathcal{A}$ to a product cone for GA and GB in $\mathcal{B}$ [49].

$$\begin{tikzcd}[column sep=small] & A\times B \arrow[dl, "\pi_0"'] \arrow[dr, "\pi_1"] & & & & G(A\times B) \arrow[dl, "G\pi_0"'] \arrow[dr, "G\pi_1"] & \\ A & & B & \overset{G}{\mapsto} & GA & & GB \end{tikzcd}$$

Let $\mathrm{Pairs}(\mathcal{C})(-;?_0,?_1): \mathcal{C}\times\mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ be the module $\mathcal{C}(-,?_0)\times\mathcal{C}(-,?_1)$ for any category $\mathcal{C}$. Suppose $\mathrm{Pairs}(\mathcal{A})$ has a right representation

$$\begin{aligned} \times^{\mathcal{A}} = (&?_0\times^{\mathcal{A}}?_1: \mathcal{A}\times\mathcal{A} \longrightarrow \mathcal{A}, \\ &\widetilde{\times^{\mathcal{A}}}_{X;A,B}: \mathcal{A}(X, A\times^{\mathcal{A}} B) \cong \mathrm{Pairs}(\mathcal{A})(X;A,B)). \end{aligned}$$

The above definition of preservation amounts to requiring that the module

$$\mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1): \mathcal{A}\times\mathcal{A} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{B}$$

is right representable, *but with a specific right representation.* To see what should be the right representation, let $\mathrm{Pairs}(G)$ be the morphism of modules $G_{-?_0}\times G_{-?_1}$

from Pairs($\mathcal{C}$) to Pairs($\mathcal{B}$)($G-; G?_0, G?_1$), *i.e.*,

$$\begin{aligned}\mathrm{Pairs}(G)_{X;A,B}\colon \mathrm{Pairs}(\mathcal{A})(X;A,B) &\equiv \mathcal{A}(X,A)\times\mathcal{A}(X,B)\\ &\xrightarrow{G_{XA}\times G_{XB}} \mathcal{B}(GX,GA)\times\mathcal{B}(GX,GA)\\ &\equiv \mathrm{Pairs}(\mathcal{B})(GX;GA,GB).\end{aligned}$$

One can show that G preserves products if and only if Pairs($\mathcal{B}$)($-; G?_0, G?_1$) is right representable with the right representation

$$\begin{aligned}G\times^{\mathcal{A}} = (\ & G(?_0\times^{\mathcal{A}}?_1)\colon \mathcal{A}\times\mathcal{A}\longrightarrow\mathcal{B},\\ & \widetilde{G\times^{\mathcal{A}}}_{X;A,B}\colon \mathcal{B}(X, G(A\times^{\mathcal{A}} B)) \cong \mathrm{Pairs}(\mathcal{B})(X;GA,GB)),\end{aligned}$$

where $\widetilde{G\times^{\mathcal{A}}}$ is defined by

$$\cfrac{\cfrac{\mathcal{A}(X, A\times^{\mathcal{A}} B) \overset{\widetilde{\times^{\mathcal{A}}}}{\cong} \mathrm{Pairs}(\mathcal{A})(X;A,B) \xrightarrow{\mathrm{Pairs}(G)} \mathrm{Pairs}(\mathcal{B})(GX;GA,GB)}{1\longrightarrow \mathrm{Pairs}(\mathcal{B})(G(A\times^{\mathcal{A}} B); GA, GB)}\cong}{\mathcal{B}(X, G(A\times^{\mathcal{A}} B)) \xrightarrow{\widetilde{G\times^{\mathcal{A}}}} \mathrm{Pairs}(\mathcal{B})(X;GA,GB).}\cong$$

Note that Pairs($\mathcal{B}$) itself need not be right representable, i.e., $\mathcal{B}$ does not necessarily have all binary products.

Example 2.5.2. (Limits)

More generally, for a category $\mathcal{I}$ and $\mathcal{A}$ with limits of shape $\mathcal{I}$, a functor $G\colon\mathcal{A}\longrightarrow\mathcal{B}$ is said to preserve those limits whenever it sends a chosen, and hence every, limiting cone for $F\colon\mathcal{I}\longrightarrow\mathcal{A}$ in $\mathcal{A}$ to one for GF in $\mathcal{B}$ [49].

Let Cones($\mathcal{A}$)($-;?$): $[\mathcal{I},\mathcal{A}] \longrightarrow\!\!\!\!\!\!+\ \mathcal{A}$ be the module $[\mathcal{I},\mathcal{A}](\Delta-, ?)$, where Δ is the diagonal functor. A category $\mathcal{A}$ is equipped with chosen limits $\mathrm{Lim}^{\mathcal{A}}F$ of shape $\mathcal{I}$ when Cones($\mathcal{A}$) has a right representation

$$\mathrm{Lim}^{\mathcal{A}} = (\mathrm{Lim}^{\mathcal{A}}\colon [\mathcal{I},\mathcal{A}]\longrightarrow\mathcal{A},\ \widetilde{\mathrm{Lim}}^{\mathcal{A}}\colon \mathcal{A}(-,\mathrm{Lim}?)\cong \mathrm{Cones}(-;?)).$$

The functor G sends any limiting *object* of every $F\colon\mathcal{I}\longrightarrow\mathcal{A}$ to that of GF in $\mathcal{B}$ if Cones($\mathcal{B}$)($-; G\mathrm{Lim}^{\mathcal{A}}?$): $[\mathcal{I},\mathcal{A}] \longrightarrow\!\!\!\!\!\!+\ \mathcal{B}$ is right representable; but for G to send any limiting *cone* of F to that of GF, the representation must a specific one. Let

$\mathrm{Cones}(G)$ be the morphism of modules given by

$$\begin{aligned}\mathrm{Cones}(G)_{I;F}\colon \mathrm{Cones}(\mathcal{A})(I;F) &\equiv [\mathcal{I},\mathcal{A}](\Delta I, F)\\ &\xrightarrow{[\mathcal{I},G]_{\Delta I F}} [\mathcal{I},\mathcal{B}](G\Delta I, GF)\\ &= [\mathcal{I},\mathcal{B}](\Delta GI, GF) \equiv \mathrm{Cones}(\mathcal{B})(GI;GF).\end{aligned}$$

One can show that G preserves $\mathcal{I}$-shaped limits if and only if $\mathrm{Cones}(\mathcal{B})(-;G?)$ has the right representation

$$\begin{aligned}G\mathrm{Lim}^{\mathcal{A}} = (&G\mathrm{Lim}^{\mathcal{A}}?\colon [\mathcal{I},\mathcal{A}] \longrightarrow \mathcal{B},\\ &\widetilde{G\mathrm{Lim}}^{\mathcal{A}}\colon \mathcal{B}(-, G\mathrm{Lim}^{\mathcal{A}}?) \cong \mathrm{Cones}(\mathcal{B})(-; G\mathrm{Lim}^{\mathcal{A}}?)),\end{aligned}$$

where $\widetilde{G\mathrm{Lim}}^{\mathcal{A}}$ is defined by

$$\cfrac{\cfrac{\mathcal{A}(-,\mathrm{Lim}^{\mathcal{A}}?) \overset{\widetilde{\mathrm{Lim}}^{\mathcal{A}}}{\cong} \mathrm{Cones}(\mathcal{A})(-;?) \xrightarrow{\mathrm{Cones}(G)} \mathrm{Cones}(G-;G?)}{1 \longrightarrow \mathrm{Cones}(\mathcal{B})(G\mathrm{Lim}^{\mathcal{A}}?;G?)}\cong}{\mathcal{B}(-,G\mathrm{Lim}^{\mathcal{A}}?) \xrightarrow[\widetilde{G\mathrm{Lim}}^{\mathcal{A}}]{} \mathrm{Cones}(\mathcal{B})(-;G?).}\cong$$

For the preservation of $\mathrm{Lim}F$ for a particular $F\colon \mathcal{I} \longrightarrow \mathcal{A}$, one fixes ? in the above to be F and replace the domain $[\mathcal{I},\mathcal{A}]$ by $\mathbf{1}$; the rest of the calculation is the same.

Example 2.5.3. (Subobject classifier)

We adopt the usual definition of subobject as isomorphic classes of monomorphisms for the moment. The functor $G\colon \mathcal{A} \longrightarrow \mathcal{B}$ preserves a subobject classifier of $\mathcal{A}$ if it sends a generic subobject $1 \overset{\top^{\mathcal{A}}}{\rightarrowtail} \Omega$ of $\mathcal{A}$ to that of $\mathcal{B}$. Further, G must preserve pullbacks of monomorphisms along arbitrary morphisms, since one expects G to preserve monomorphisms and send the characteristic morphism of a monomorphism m in $\mathcal{A}$ to the characteristic morphism of Gm in $\mathcal{B}$. Usually this is implicit because often $\mathcal{A}$ and $\mathcal{B}$ are toposes and G is to preserve all finite limits.

With the definition of subobject as representable sieves, let $\mathrm{Sub}(\mathcal{C})\colon \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ be the module $\mathbf{1} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ defined in Section 2.4 for a category $\mathcal{C}$ with pullbacks of monomorphisms. For a functor $G\colon \mathcal{A} \longrightarrow \mathcal{B}$ with $\mathcal{A}$ having pullbacks of monomorphisms, and for a subfunctor $S \hookrightarrow \mathcal{A}(-,A)$, define a subfunctor GS of

$\mathcal{B}(-, GA)$ by that generated by the image of S by G:

$$GS- = \{h\colon - \longrightarrow GA | \exists B \in \mathcal{A},\ k\colon B \longrightarrow A \in SB,\ l\colon - \longrightarrow GB.\ h = Gk \circ l\}.$$

A functor G preserves monomorphisms and pullbacks of them if and only if GS is representable whenever S is, and $(GS)[Gf] = G(S[f])$ (hence the former is representable) for any $S \hookrightarrow \mathcal{A}(-, A)$ and $f\colon A' \longrightarrow A$. Any representation of S with a counit $\overset{m}{\rightarrowtail} A$ induces one of GS with the counit Gm, i.e., this action on subobjects corresponds to that of G on (mono)morphisms.

Suppose $\mathrm{Sub}(\mathcal{A})$ has a representation $\Omega^{\mathcal{A}} \in \mathcal{A}$, $\widetilde{\Omega^{\mathcal{A}}}\colon \mathcal{A}(-, \Omega) \cong \mathrm{Sub}(\mathcal{A})(-)$ and $G\colon \mathcal{A} \longrightarrow \mathcal{B}$ preserves pullbacks of monomorphisms. The above definition of subobject classifier preservation translates to the requirement that $\mathrm{Sub}(\mathcal{B})(-)$ $: \mathcal{B}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ has the representation

$$\begin{aligned}&(G\Omega \in \mathcal{B},\\ &\widetilde{G\Omega}\colon \mathcal{B}(-, G\Omega) \cong \mathrm{Sub}^{\mathcal{B}}(-))\end{aligned}$$

where $\widetilde{G\Omega}$ is defined by

$$\cfrac{\cfrac{\mathcal{A}(-,\Omega) \overset{\widetilde{\Omega}}{\cong} \mathrm{Sub}(\mathcal{A})(-) \xrightarrow{\mathrm{Sub}(G)} \mathrm{Sub}(\mathcal{B})(G-)}{1 \longrightarrow \mathrm{Sub}^{\mathcal{B}}(G\Omega)} \cong}{\mathcal{B}(-, G\Omega) \underset{\widetilde{G\Omega}}{\longrightarrow} \mathrm{Sub}^{\mathcal{B}}(-)} \cong$$

with

$$\begin{array}{llcl}\mathrm{Sub}(G) & : \mathrm{Sub}(\mathcal{A})(-) & \longrightarrow & \mathrm{Sub}(\mathcal{B})(G-)\\ & \quad S & \mapsto & GS.\end{array}$$

2.6 Universal structure

The above examples show a uniform relationship between right representations in domain categories and those in target categories induced by structure preserving functors. This leads to our definition of universal structure. First, observe the following.

Proposition 2.6.1. Given a functor $G\colon \mathcal{A} \longrightarrow \mathcal{B}$, the morphism part of G exhibits $G_*\colon \mathcal{B}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathbf{Set}$ as $\mathrm{Lan}_{G\times \mathrm{id}_{\mathcal{A}}}\mathcal{A}(-,?)$:

$$\begin{tikzcd}
 & \mathcal{B}^{\mathrm{op}} \times \mathcal{A} \arrow[dr, "G_*"] \arrow[d, phantom, "\Uparrow G_{-?}"] & \\
\mathcal{A}^{\mathrm{op}} \times \mathcal{A} \arrow[ur, "G \times \mathrm{id}_{\mathcal{A}}"] \arrow[rr, "\mathcal{A}(-,?)"'] & {} & \mathbf{Set}.
\end{tikzcd}$$

Proof The isomorphism $\overline{G_{-?}}$ required in Definition 2.3.2 is equal to the inverse of the following isomorphism given by successive application of the Yoneda lemma.

$$\begin{aligned}
&\overline{G_{(B,A),X}}^{-1}\colon [(\mathcal{A}^{\mathrm{op}} \times \mathcal{A})^{\mathrm{op}}, \mathbf{Set}]((\mathcal{B}^{\mathrm{op}} \times \mathcal{A})((G \times \mathrm{id})(-,?),(B,A)),[\mathcal{A}(-,?),X])\\
&\qquad\cong \int_{C\in\mathcal{A}}\int_{D\in\mathcal{A}^{\mathrm{op}}}[\mathcal{B}(B,GC),[\mathcal{A}(D,A),[\mathcal{A}(C,D),X]]]\\
&\qquad\cong \int_{C\in\mathcal{A}}[\mathcal{B}(B,GC),[\mathcal{A}(C,A),X]]\\
&\qquad\cong \int_{C\in\mathcal{A}}[\mathcal{A}(C,A),[\mathcal{B}(B,GC),X]]\\
&\qquad\cong [\mathcal{B}(B,GA),X] = [G_*(B,A),X]
\end{aligned}$$

∎

Corollary 2.6.2. Given functors $G\colon \mathcal{A} \longrightarrow \mathcal{B}$ and $R\colon \mathcal{C} \longrightarrow A$, the natural transformation $(G_{A,RC})_{A\in\mathcal{A},C\in\mathcal{C}}$ exhibits $(GR)_*$ as a left Kan extension of $\mathcal{A}(-,R?)$ along $G \times \mathrm{id}_{\mathcal{C}}$. ∎

Then, let $\mathcal{A}$ be a category with binary products with a chosen representation $(\times^{\mathcal{A}}, \widetilde{\times^{\mathcal{A}}})$ for $\mathrm{Pairs}(\mathcal{A})$. Corollary 2.6.2 means that, for any $G\colon \mathcal{A} \longrightarrow \mathcal{B}$, $(G_{C;A\times B})_{C,A,B\in\mathcal{A}}$ exhibits the functor $\mathcal{B}(-,G(?_0\times^{\mathcal{A}}?_1))\colon \mathcal{B}^{\mathrm{op}} \times (\mathcal{A}\times\mathcal{A}) \longrightarrow \mathbf{Set}$ as a left Kan extension of $\mathcal{A}(-,?_0\times ?_1)$ along $G \times \mathrm{id}_{\mathcal{A}\times\mathcal{A}}$. So, by taking H in Proposition 2.3.3 to be $\mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1)$, we have a natural isomorphism

$$\begin{aligned}
&[\mathcal{B}^{\mathrm{op}} \times (\mathcal{A}\times\mathcal{A}),\mathbf{Set}](\mathcal{B}(-,G(?_0\times^{\mathcal{A}}?_1)),\mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1))\\
&\quad\cong [\mathcal{A}^{\mathrm{op}} \times (\mathcal{A}\times\mathcal{A}),\mathbf{Set}](\mathcal{A}(-,?_0\times^{\mathcal{A}}?_1),\mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1)\circ(G\times \mathrm{id}_{\mathcal{A}\times\mathcal{A}})).
\end{aligned}$$

By routine calculation, one can check that the natural transformation

$$\widetilde{G\times^{\mathcal{A}}}\colon \mathcal{B}(-,G(?_0\times^{\mathcal{A}}?_1)) \longrightarrow \mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1))$$

in Example 2.5.1 is the one determined by

$$\mathrm{Pairs}(G)\widetilde{\times^{\mathcal{A}}} \colon \mathcal{A}(-,?_0\times^{\mathcal{A}}?_1) \longrightarrow \mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1)\circ(G\times\mathrm{id}_{\mathcal{A}\times\mathcal{A}}).$$

Diagrammatically, it can be shown as

$$\begin{tikzcd}
& \mathcal{B}^{\mathrm{op}}\times(\mathcal{A}\times\mathcal{A}) \arrow[dr, bend left, "\mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1)"] \arrow[dr, "\mathcal{B}(-,G(?_0\times^{\mathcal{A}}?_1))"'] \arrow[dr, phantom, shift left=4, "\Uparrow\widetilde{G\times^{\mathcal{A}}}"] & \\
\mathcal{A}^{\mathrm{op}}\times(\mathcal{A}\times\mathcal{A}) \arrow[ur, "G\times\mathrm{id}"] \arrow[rr, "\mathcal{A}(-,?_0\times^{\mathcal{A}}?_1)"'] \arrow[rr, phantom, shift left=8, "\Uparrow G_{-,?_0\times^{\mathcal{A}}?_1}"] & & \mathbf{Set}
\end{tikzcd}$$

$$= \quad \begin{tikzcd}
& \mathcal{B}^{\mathrm{op}}\times(\mathcal{A}\times\mathcal{A}) \arrow[dr, "\mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1)"] & \\
\mathcal{A}^{\mathrm{op}}\times(\mathcal{A}\times\mathcal{A}) \arrow[ur, "G\times\mathrm{id}"] \arrow[rr, shift left, "\mathrm{Pairs}(\mathcal{A})(-;?_0,?_1)"] \arrow[rr, shift right, "\mathcal{A}(-,?_0\times^{\mathcal{A}}?_1)"'] \arrow[rr, phantom, "\cong\widetilde{\times^{\mathcal{A}}}"] \arrow[rr, phantom, shift left=8, "\Uparrow\mathrm{Pairs}(G)"] & & \mathbf{Set}.
\end{tikzcd}$$

So, requiring $\widetilde{G\times^{\mathcal{A}}}$ to be an isomorphism is equivalent to requiring that $\mathrm{Pairs}(G)$ exhibits $\mathrm{Pairs}(\mathcal{B})(-;G?_0,G?_1)$ as a left Kan extension of $\mathrm{Pairs}(\mathcal{A})(-;?_0,?_1)$ along $G\times\mathrm{id}_{\mathcal{A}\times\mathcal{A}}$, thus eliminating the reference to the particular representation $\times^{\mathcal{A}}$. The choice for representation can be avoided for the cases of limits and subobject classifiers in a similar way, using $\mathrm{Cones}(G)$ and $\mathrm{Sub}(G)$ instead of $\mathrm{Pairs}(G)$.

To make this precise, we define a universal structure as a family of modules whose components are related with each other in a uniform way.

Definition 2.6.3. A *universal structure* $\mathcal{S}$ consists of the following data.

- a subcategory $\mathrm{Dom}(\mathcal{S})$, the *domain* of $\mathcal{S}$, of $\mathbf{Cat}$.

- a functor $\mathrm{Param}(\mathcal{S})\colon \mathrm{Dom}(\mathcal{S}) \longrightarrow \mathbf{Cat}$, the *parameter functor* of $\mathcal{S}$.
- a module $\mathrm{Module}(\mathcal{S})(\mathcal{C})(-;?)\colon \mathrm{Param}(\mathcal{S})(\mathcal{C}) \longrightarrow\!\!\!\!\!\!\!+\; \mathcal{C}$ for each $\mathcal{C}$ in $\mathrm{Dom}(\mathcal{S})$, the *defining module of $\mathcal{S}$ for $\mathcal{C}$*.
- a morphism of modules

$$\begin{aligned}&\mathrm{Module}(\mathcal{S})(G)\colon \mathrm{Module}(\mathcal{S})(\mathcal{A})(-;?)\\ &\longrightarrow \mathrm{Module}(\mathcal{S})(\mathcal{B})(G-;\mathrm{Param}(\mathcal{S})(G)?)\colon \mathrm{Param}(\mathcal{S})(\mathcal{A}) \longrightarrow\!\!\!\!\!\!\!+\; \mathcal{A}\end{aligned}$$

  for each $G\colon \mathcal{A} \longrightarrow \mathcal{B}$ in $\mathrm{Dom}(\mathcal{S})$, the *defining module morphism of $\mathcal{S}$ for G*.

Example 2.6.4. 'Binary products' is a universal structure defined by the following data (omitting 'binary products' from Dom('binary product') *etc.*):

- $\mathrm{Dom} = \mathbf{Cat}$
- $\mathrm{Param}\colon \mathbf{Cat} \longrightarrow \mathbf{Cat}$, $\mathcal{C} \mapsto \mathcal{C} \times \mathcal{C}$,
- $\mathrm{Module}(\mathcal{C}) = \mathrm{Pairs}(\mathcal{C})\colon \mathcal{C} \times \mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\; \mathcal{C}$,
- $$\begin{aligned}&\mathrm{Module}(G) = \mathrm{Pairs}(G)\\ &\quad\colon \mathrm{Pairs}(\mathcal{A})(-;(?_0,?_1)) \longrightarrow \mathrm{Pairs}(\mathcal{B})(G-;(G\times G)(?_0,?_1)).\end{aligned}$$

The reason to introduce $\mathrm{Dom}(\mathcal{S})$ is that, for some universal structures on $\mathcal{C}$, extra structure on $\mathcal{C}$ is needed as it is the case for subobject classifiers.

Example 2.6.5. 'Subobject classifier' is a universal structure defined by

- $\mathrm{Dom} = \{$categories with pullbacks of monomorphisms and functors that preserve them$\}$.
- $\mathrm{Param}\colon \mathrm{Dom} \longrightarrow \mathbf{Cat}$, $\mathcal{C} \mapsto \mathbf{1}$
- $\mathrm{Module}(\mathcal{C}) = \mathrm{Sub}(\mathcal{C})\colon \mathbf{1} \longrightarrow\!\!\!\!\!\!\!+\; \mathcal{C}$,
- $\mathrm{Module}(G) = \mathrm{Sub}(G)\colon \mathrm{Sub}(\mathcal{A})(-;*)) \longrightarrow \mathrm{Sub}(\mathcal{B})(G-;*)$.

One can consider layered definitions of universal structures, a lower level definition providing extra structure for the domain categories of a higher structure.

Dually, a couniversal structure $\mathcal{T}$ is a universal structure where $\mathrm{Param}(\mathcal{T})(\mathcal{C})$ is replaced with $(\mathrm{Param}(\mathcal{T})(\mathcal{C}^{\mathrm{op}}))^{\mathrm{op}}$, $\mathrm{Module}(\mathcal{T})(\mathcal{C})$ with $(\mathrm{Module}(\mathcal{T})(\mathcal{C}^{\mathrm{op}}))^{\mathrm{op}}$. More concretely, it is given by

- a subcategory $\mathrm{Dom}(\mathcal{T})$ of **Cat**.
- a functor $\mathrm{Param}\colon \mathrm{Dom}(\mathcal{T}) \longrightarrow \mathbf{Cat}$.
- a module $\mathrm{Module}(\mathcal{T})(\mathcal{C})(-;?)\colon \mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\ \ \mathrm{Param}(\mathcal{C})$ for each $\mathcal{C}$ in $\mathrm{Dom}(\mathcal{T})$.
- a morphism of modules

  $$\begin{aligned}&\mathrm{Module}(\mathcal{T})(G)\colon \mathrm{Module}(\mathcal{T})(\mathcal{A})(-;?)\\&\longrightarrow \mathrm{Module}(\mathcal{T})(\mathcal{B})(\mathrm{Param}(\mathcal{T})(G)-;G?)\end{aligned}$$

  for each $G\colon \mathcal{A} \longrightarrow \mathcal{B}$.

This definition can be generalised in several ways. For example, to accommodate an initial algebra for $T\colon \mathcal{C} \longrightarrow \mathcal{C}$, Module('initial algebra')$(\mathcal{C})$ should take an extra argument T. One may allow $\mathrm{Dom}(\mathcal{S})$ to be not just a subcategory of **Cat** but a category equipped with a functor $\mathrm{Extra}(\mathcal{S})\colon \mathrm{Dom}(\mathcal{S}) \longrightarrow \mathbf{Cat}$, specifying the category of those extra arguments; for example, Extra('initial algebra')$(\mathcal{C}) = [\mathcal{C}, \mathcal{C}]$. The codomain of $\mathrm{Module}(\mathcal{S})(\mathcal{C})$ may be generalised from just $\mathcal{C}$ to a result category $\mathrm{Res}(\mathcal{S})(\mathcal{C}, T)$ which depends on $\mathcal{C}$ and an extra argument T. However, for the moment we keep our simple definition for brevity.

A later development may require uniformity with respect to natural transformations $G \longrightarrow G'\colon \mathcal{A} \longrightarrow \mathcal{B}$ in $\mathrm{Dom}(\mathcal{S})$, but at this stage there seems no need for this.

Also, it is more natural to change $\mathrm{Dom}(\mathcal{S})$ from a subcategory of **Cat** to "categories with extra structure needed to define $\mathcal{S}$", once one obtains a deeper understanding of how $\mathrm{Module}(\mathcal{S})$ arises from an underlying structure.

With this definition of universal structure, those categories with a specified universal structure and structure-preserving functors between them can be defined in the following uniform way.

Definition 2.6.6. Given a universal structure $\mathcal{S}$, a category $\mathcal{C}$ in $\mathrm{Dom}(\mathcal{S})$ *has the structure* $\mathcal{S}$ if $\mathrm{Module}(\mathcal{S})(\mathcal{C})$ is right representable.

Definition 2.6.7. Given a universal structure $\mathcal{S}$, a functor $G\colon \mathcal{A} \longrightarrow \mathcal{B}$ in $\mathrm{Dom}(\mathcal{S})$ *preserves* $\mathcal{S}$ if $\mathrm{Module}(\mathcal{S})(G)$ exhibits $\mathrm{Module}(\mathcal{S})(\mathcal{B})(-;\mathrm{Param}(\mathcal{S})(G)?)$ as a left Kan extension of $\mathrm{Module}(\mathcal{S})(\mathcal{A})(-;?)$ along $G \times \mathrm{id}_{\mathrm{Param}(\mathcal{A})}$.

$$\begin{tikzcd}
 & \mathcal{B}^{\mathrm{op}} \times \mathrm{Param}(\mathcal{A}) \arrow[dr, "\mathrm{Module}(\mathcal{B})(-;\mathrm{Param}(G)?)"] \arrow[d, phantom, "\Uparrow \mathrm{Module}(G)"] & \\
\mathcal{A}^{\mathrm{op}} \times \mathrm{Param}(\mathcal{A}) \arrow[ur, "G \times \mathrm{id}_{\mathrm{Param}(\mathcal{A})}"] \arrow[rr, "\mathrm{Module}(\mathcal{A})(-;?)"'] & & \mathbf{Set}
\end{tikzcd}$$

Dually, we define:

Definition 2.6.8. Given a couniversal structure $\mathcal{T}$, a category $\mathcal{C}$ in $\mathrm{Dom}(\mathcal{S})$ *has the structure* $\mathcal{T}$ if $\mathrm{Module}(\mathcal{T})(\mathcal{C})$ is left representable.

Definition 2.6.9. Given a couniversal structure $\mathcal{T}$, a functor $G\colon \mathcal{A} \longrightarrow \mathcal{B}$ in $\mathrm{Dom}(\mathcal{T})$ *preserves* $\mathcal{T}$ if $\mathrm{Module}(\mathcal{S})(G)$ exhibits $\mathrm{Module}(\mathcal{T})(\mathcal{B})(\mathrm{Param}(\mathcal{S})(G)-;?)$ as a left Kan extension of $\mathrm{Module}(\mathcal{S})(\mathcal{A})(-;?)$ along $\mathrm{id}_{\mathrm{Param}(\mathcal{A})} \times G$.

$$\begin{tikzcd}
 & \mathrm{Param}(\mathcal{A}) \times \mathcal{B}^{\mathrm{op}} \arrow[dr, "\mathrm{Module}(\mathcal{B})(\mathrm{Param}(G)?;-)"] \arrow[d, phantom, "\Uparrow \mathrm{Module}(G)"] & \\
\mathrm{Param}(\mathcal{A}) \times \mathcal{A}^{\mathrm{op}} \arrow[ur, "\mathrm{id}_{\mathrm{Param}(\mathcal{A})} \times G"] \arrow[rr, "\mathrm{Module}(\mathcal{A})(-;?)"'] & & \mathbf{Set}
\end{tikzcd}$$

2.7 Changing variance

As explained in Chapter 1, we try to capture the notion of universal structure as independently as possible from the particular choice of representation. To do so, one needs to obtain, from a module $P: \mathcal{A} \longrightarrow\!\!\!\!\!+\ \mathcal{C}$ with a right representation R, another $\bar{P}: \mathcal{C} \longrightarrow\!\!\!\!\!+\ \mathcal{A}$ which has R as its left representation; the reason for this is as follows.

The spirit of category theory is to concentrate on how objects behave rather than what they are. A universal construction embodies this spirit in that one specifies, up to isomorphism, one object by describing how it behaves with respect to all objects in terms of morphisms. This description, however, appears to be incomplete, giving information either on all morphisms into the specified object, or on those out of it.

For example, if A and B are objects of a category $\mathcal{C}$ with products, $A \times B$ is specified by the information that the morphisms into it naturally correspond to pairs of morphisms to A and B: $\mathcal{C}(-, A \times B) \overset{\tilde{\times}}{\cong} \mathcal{C}(-, A) \times \mathcal{C}(-, B)$. There is no reference to morphisms out of $A \times B$.

What can be said about those morphisms out of $A \times B$? If there is no simple formula for them that does not depend on what $A \times B$ is, one might imagine that one must commit to a particular choice of $A \times B$ among its isomorphic copies.

Fortunately, a simple application of the Yoneda lemma gives a description of morphisms out of $A \times B$ independent of the representation $(\times, \tilde{\times})$.

$$\begin{aligned} \mathcal{C}(A \times B, -) &\cong [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](\mathcal{C}(?, A \times B), \mathcal{C}(?, -)) \\ &\cong [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](\mathcal{C}(?, A) \times \mathcal{C}(?, B), \mathcal{C}(?, -)) \end{aligned}$$

This can be systematically generalised as follows. First recall that $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$ is the free cocompletion of $\mathcal{C}$ (see [40], Th 4.51).

Theorem 2.7.1. (Kan's theorem)

For a cocomplete $\mathcal{B}$, $\mathrm{Cocts}[[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}], \mathcal{B}] = \mathrm{Ladj}[[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}], \mathcal{B}] \simeq [\mathcal{C}, \mathcal{B}]$. ∎

The equivalence is given, in the forward direction, by composition with the Yoneda embedding $y\colon \mathcal{C} \longrightarrow [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$, and in the reverse direction, by Lan_y-. Given a cocontinuous $S\colon [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}] \longrightarrow \mathcal{B}$, the left adjoint T to S is given by $TB = \mathcal{B}(Sy-, B)$.

By taking $\mathcal{B}$ to be $[\mathcal{C}, \mathbf{Set}]^{\mathrm{op}}$, one has the adjoint pair $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}] \underset{\perp}{\xrightarrow{\mathrm{Lan}_y y'}} [\mathcal{C}, \mathbf{Set}]^{\mathrm{op}}$ that corresponds to the Yoneda embedding $y'^{\mathrm{op}}\colon \mathcal{C} \longrightarrow [\mathcal{C}, \mathbf{Set}]^{\mathrm{op}}$, $C \mapsto \mathcal{C}(C, -)$:

$$\begin{tikzcd}
{[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]} \arrow[rr, shift left, "\mathrm{Lan}_y y'"] \arrow[rr, phantom, "\perp"] & & {[\mathcal{C}, \mathbf{Set}]^{\mathrm{op}}} \arrow[ll, shift left] \\
& \mathcal{C}. \arrow[ul, "y"] \arrow[ur, "y'"'] &
\end{tikzcd}$$

Proposition 2.7.2. Given $P\colon \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ with a representation $\mathcal{C}(-, R) \cong P$, $(\mathrm{Lan}_y y')(P)\colon \mathcal{C} \longrightarrow \mathbf{Set}$ is representable with the representing object being R.

Proof This follows from that y is fully faithful and a general fact that the unit of $\mathrm{Lan}_K G$ is an isomorphism if K is fully faithful ([40], Proposition 4.23). Direct calculation is also easy:

$$\begin{aligned}
(\mathrm{Lan}_y y')(P) &\cong [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](P?, \mathcal{C}(?, -)) \\
&\cong [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](\mathcal{C}(?, R), \mathcal{C}(?, -)) \\
&\cong \mathcal{C}(R, -),
\end{aligned}$$

with the right hand side of the first line being the functor that sends $C \in \mathcal{C}$ to $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](P?, \mathcal{C}(?, C)) \in \mathbf{Set}$. ∎

Corollary 2.7.3. For two categories $\mathcal{A}$ and $\mathcal{B}$, there is an adjoint pair

$$\mathbf{Mod}(\mathcal{A}, \mathcal{C}) \underset{\perp}{\xrightarrow{\overline{(-)}^{\mathcal{A},\mathcal{C}}}} \mathbf{Mod}(\mathcal{C}, \mathcal{A})^{\mathrm{op}}$$

such that, for $P\colon \mathcal{A} \longrightarrow\!\!\!\!\!+\; \mathcal{C}$ with a right representation (R, ρ), $\overline{P}^{\mathcal{A},\mathcal{C}}$ admits a left representation (R, λ) for some λ.

Proof It follows by parameterising Proposition 2.7.2 with $\mathcal{A}$:

$$[\mathcal{C}^{\mathrm{op}} \times \mathcal{A}, \mathbf{Set}]$$

$$\cong [\mathcal{A}, [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]] \underset{\longleftarrow}{\overset{[\mathcal{A}, \mathrm{Lan}_{\mathcal{Y}}\mathcal{Y}']}{\xrightarrow{\perp}}} [\mathcal{A}, [\mathcal{C}, \mathbf{Set}]^{\mathrm{op}}]$$
$$= [\mathcal{A}^{\mathrm{op}}, [\mathcal{C}, \mathbf{Set}]]^{\mathrm{op}} \cong [\mathcal{A}^{\mathrm{op}} \times \mathcal{C}, \mathbf{Set}]^{\mathrm{op}}$$

∎

Note that this is not an instance of Proposition 2.7.2 with $\mathcal{C}^{\mathrm{op}}$ replaced by $\mathcal{C}^{\mathrm{op}} \times \mathcal{A}$.

More concretely, $\overline{(-)}^{\mathcal{A},\mathcal{C}}$ is given by

$$\overline{(-)}^{\mathcal{A},\mathcal{C}} : [\mathcal{C}^{\mathrm{op}} \times \mathcal{A}, \mathbf{Set}] \longrightarrow [\mathcal{A}^{\mathrm{op}} \times \mathcal{C}, \mathbf{Set}]^{\mathrm{op}}$$

$$\begin{array}{ccc} PCA & & [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](P?A, \mathcal{C}(?, C)) \\ t_{C,A}\Big\downarrow & \mapsto & \Big\uparrow [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](t_{?A}, \mathrm{id}_{\mathcal{C}(?,C)}) \\ QCA & & [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](Q?A, \mathcal{C}(?, C)) \end{array}$$
$$(A \in \mathcal{A},\ C \in \mathcal{C})$$

From the self-duality of the diagram before Proposition 2.7.2, it is obvious that the right adjoint to $\overline{(-)}^{\mathcal{A},\mathcal{C}}$ is $\overline{(-)^{\mathrm{op}}}^{\mathcal{C}^{\mathrm{op}},\mathcal{A}^{\mathrm{op}}}$, sending a left representable module from $\mathcal{C}$ to $\mathcal{A}$ to a right representable module from $\mathcal{A}$ to $\mathcal{C}$.

So, now we have an explicit formula to reverse the variance of right/left representable module, which does not depend on the choice of representations.

It is also obvious that the adjoint restricts to an equivalence between the subcategory of right representable modules from $\mathcal{A}$ to $\mathcal{C}$ and that of left representables modules from $\mathcal{C}$ to $\mathcal{A}$, both being equivalent to $[\mathcal{A}, \mathcal{C}]$ by Proposition 2.2.11. This equivalence may mislead one to believe that $\overline{P}$ determines P, so in particular that if $\overline{P}$ is left (right) representable then P is right (left) representable. This is not true. The precise statement is as follows.

Proposition 2.7.4. For $P\colon \mathcal{A} \longrightarrow\!\!\!\!\!\!\!\!+\;\; \mathcal{C}$, the following are equivalent.

(i) P has a right representation.

(ii) $\overline{P}$ has a left representation $(L\colon \mathcal{A} \longrightarrow \mathcal{C},\ \lambda\colon \mathcal{C}(L?, -) \cong \overline{P})$ and the counit $\lambda_{A,LA}(\mathrm{id}_{LA})\colon P?A \longrightarrow \mathcal{C}(?, LA)$ has a left inverse.

(iii) $\overline{P}$ has a left representation and the following is a left Kan extension diagram for each $A \in \mathcal{A}$.

$$\begin{tikzcd}
& \mathbf{Set}^{\mathrm{op}} \arrow[dr, "{[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](P?A, [-, P?A])}"] & \\
\mathcal{C} \arrow[ur, "(P-A)^{\mathrm{op}}"] \arrow[rr, "\overline{P}A-"'] \arrow[rr, phantom, "\Uparrow \phi", shift left=6] & & \mathbf{Set},
\end{tikzcd}$$

where ϕ is defined by

$$\begin{aligned}\phi_C \colon \overline{P}AC &\equiv [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](P?A, \mathcal{C}(?, C)) \\ &\xrightarrow{[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](\mathrm{id}, (P-A)_{?C})} [\mathcal{C}^{\mathrm{op}}, \mathbf{Set}](P?A, [PCA, P?A])\end{aligned}$$

Proof By routine calculation. (Also see Theorem 4.80 of [40]; the left Kan extension diagram shows that $P - A$ preserves the index colimit mentioned there.) ∎

This extra condition on a representable $\overline{P}$ is not strong, since this holds for the most common types of P that are constructed from Hom-sets by limit constructions and exponentials in **Set**. In such case, the representability of P can be checked by that of $\overline{P}$.

Example 2.7.5. (Initial object)

As the simplest application of Proposition 2.7.4, we show that the initial object of a category $\mathcal{C}$ coincides with the limit of $\mathrm{id}_{\mathcal{C}}$ (*cf.* Lemma X.1.1. [49]). The limit $\mathrm{Lim}\ \mathrm{id}_{\mathcal{C}}$ is a representing objects for the functor

$$[\mathcal{C}, \mathbf{Set}](\Delta - (?), \mathrm{id}_{\mathcal{C}}(?)) \colon \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$$

and the initial object 0 is a representing object for the functor

$$\Delta 1 \colon \mathcal{C} \longrightarrow \mathbf{Set},$$

where $\Delta - (C) \equiv -$ for all $C \in \mathcal{C}$. Then, since we have

$$\begin{aligned}[\mathcal{C}, \mathbf{Set}](\Delta - (?), \mathrm{id}_{\mathcal{C}}(?)) &\cong [\mathcal{C}, \mathbf{Set}](\Delta 1(?), \mathcal{C}(-, ?)) \\ &\cong \overline{\Delta 1}^{\mathbf{1}, \mathcal{C}},\end{aligned}$$

and since, for any $C \in \mathcal{C}$, any $t\colon \Delta 1(?) \longrightarrow \mathcal{C}(C,?)$ trivially has a left inverse, Proposition 2.7.4 shows that $[\mathcal{C},\mathbf{Set}](\Delta - (?), \mathrm{id}_{\mathcal{C}}(?))$ is representable if and only if $\Delta 1$ is, with the same representing objects.

Example 2.7.6. (Exponential)

Exponentials are an example where one has to change the variance of an module to define another. Suppose $\mathcal{C}$ has binary products, i.e., Pairs: $\mathcal{C} \times \mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ is right representable. An exponentials B^A may be defined with a particular right representation $\times$ for Pairs.

$$\mathcal{C}(-, B^A) \cong \mathcal{C}(- \times A, B)$$

One can avoid this choice by replacing the right hand side by

$$\mathcal{C}(-, B^A) \cong \overline{\mathrm{Pairs}}(-, A; B).$$

Thus, if the module Func: $\mathcal{C}^{\mathrm{op}} \times \mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$

$$\begin{array}{llll} \mathrm{Func} & : \mathcal{C}^{\mathrm{op}} \times (\mathcal{C}^{\mathrm{op}} \times \mathcal{C}) & \longrightarrow & \mathbf{Set} \\ & \quad (X; A, B) & \mapsto & \overline{\mathrm{Pairs}}^{\mathcal{C}\times\mathcal{C},\mathcal{C}}(X, A; B) \end{array}$$

has a right representation

$$\mathrm{Exp} = ((?)^{(-)}\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \longrightarrow \mathcal{C},\ \widetilde{\mathrm{Exp}}\colon \mathcal{C}(X, B^A) \cong \mathrm{Func}(X; A, B)),$$

then $\mathcal{C}$ has exponentials. Note that $\mathrm{Func}(-; A, B)$ is an exponential of $\mathcal{C}(-, B)$ to $\mathcal{C}(-, A)$ in $[\mathcal{C}^{\mathrm{op}}, \mathbf{Set}]$.

This definition is more natural than the usual one in a sense, because although $(-)^A$ is right adjoint to $- \times A$ for each A, it is clumsy to describe the whole functor $(-)^?$ in the language of adjointness.

To finish this section, we present the universal structure 'exponentials' using Func. Recall that, given $\mathcal{A}$ with binary products and exponentials, and given a product preserving $G\colon \mathcal{A} \longrightarrow \mathcal{B}$, G is said to preserve exponentials if

$$G(B^A) \times GA \cong G(B^A \times A) \xrightarrow{G(\varepsilon)} GB,$$

where ε is the evaluation map in $\mathcal{A}$, is the universal morphism exhibiting $G(B^A)$ as GB^{GA} in $\mathcal{B}$.

Translating this in terms of Func, the universal structure 'exponentials' is

- Dom = { categories with binary products and product preserving functors.}
- Param: $\mathcal{C} \mapsto \mathcal{C}^{\mathrm{op}} \times \mathcal{C}$.
- $\mathrm{Module}(\mathcal{C}) = \mathrm{Func}(\mathcal{C})(-; ?_0, ?_1)\colon \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$.
- $\mathrm{Module}(G) = \mathrm{Func}(G)\colon \mathrm{Func}(\mathcal{A})(-; ?_0, ?_1) \longrightarrow \mathrm{Func}(G-; G?_0, G?_1)$, where $\mathrm{Func}(G)$ is defined by

$$\begin{aligned}
\mathrm{Func}(G)_{XAB}\colon\ & \mathrm{Func}(\mathcal{A})(X; A, B) \\
& \equiv \overline{\mathrm{Pairs}(\mathcal{A})}^{\mathcal{A}\times\mathcal{A},\mathcal{A}}(X, A; B) \\
& \equiv [\mathcal{A}^{\mathrm{op}}, \mathbf{Set}](\mathrm{Pairs}(\mathcal{A})(?; X, A), \mathcal{A}(?, B)) \\
& \xrightarrow{[\mathcal{A}^{\mathrm{op}},\mathbf{Set}](\mathrm{id}, G_{?B})} [\mathcal{A}^{\mathrm{op}}, \mathbf{Set}](\mathrm{Pairs}(\mathcal{A})(?; X, A), \mathcal{B}(G?, GB)) \\
& \cong [\mathcal{B}^{\mathrm{op}}, \mathbf{Set}](\mathrm{Pairs}(\mathcal{B})(?; GX, GA), \mathcal{B}(?, GB)) \\
& \equiv \mathrm{Func}(GX; GA, GB).
\end{aligned}$$

The isomorphism is the one induced by the left Kan extension diagram that expresses G preserves products:

$$\begin{tikzcd}
 & \mathcal{B}^{\mathrm{op}} \arrow[dr, "\mathrm{Pairs}(\mathcal{B})(?; GX, GA)"] & \\
\mathcal{A}^{\mathrm{op}} \arrow[ur, "G"] \arrow[rr, "\mathrm{Pairs}(\mathcal{A})(?; X, A)"'] \arrow[urr, phantom, "\Uparrow \mathrm{Pairs}(G)_{?XA}"] & & \mathbf{Set}
\end{tikzcd}$$

It is routine to check that G preserves exponentials in the usual sense if and only if G does so in the sense of Definition 2.6.7 with $\mathcal{S} =$'exponentials'.

2.8 Composite representations

Composing right and left representations of given modules, one can obtain a new functor. One may also obtain some non-functors, e.g., $(X)^{(X)}$, too, but these are not considered here. It is obvious that the choice of representations does not essentially affect the resultant functor. To reason about this functor independently of these choices, one needs to construct, from the original modules, a module that converges to the resultant functor. This can be treated more abstractly in terms of the inclusion of **Cat** into the bicategory of modules and module calculus [12]. However, we would like to keep track of explicit formulas for those composite modules.

At first sight, the question seems trivial. For example, $-_0 \times (-_1 \times -_2)$ is trivially a right representation of the module that gives a set of triples of morphisms. One can simply unwind the definition of Pairs twice.

$$\begin{aligned}\mathcal{C}(X, A \times (B \times C)) &\cong \mathcal{C}(X, A) \times \mathcal{C}(X, B \times C) \\ &\cong \mathcal{C}(X, A) \times (\mathcal{C}(X, B) \times \mathcal{C}(X, C)).\end{aligned}$$

However, for $A \otimes (B \otimes C)$ in **SL** of Example 2.4.4, simply unwinding the definition of Bilin does not work.

$$\begin{aligned}\mathbf{SL}(A \otimes (B \otimes C); X) &\cong \mathrm{Bilin}(A, B \otimes C; X) \\ &\cong \{h \colon UA \times U(B \otimes C) \longrightarrow UX | ......\} \\ &\cong \cdots ?\end{aligned}$$

Therefore, one needs a simple formula that does not unwind the definition of modules involved.

The desired formula is given by Lemma 2.2.8. Recall that, for $G \colon \mathcal{C}^{\mathrm{op}} \longrightarrow \mathbf{Set}$, $\int^{Y \in \mathcal{C}} \mathcal{C}(D, Y) \times GY \cong GD$. Setting $G = \mathcal{C}(A \otimes -, X)$ and $D = B \otimes C$, one can show that

$$\begin{aligned}\mathcal{C}(A \otimes (B \otimes C), X) &\cong \int^{Y} \mathcal{C}(B \otimes C, Y) \times \mathcal{C}(A \otimes Y, X) \\ &\cong \int^{Y} \mathrm{Bilin}(B, C; Y) \times \mathrm{Bilin}(A, Y; X)\end{aligned}$$

Similarly,

$$
\begin{aligned}
&\mathcal{C}((A \otimes B) \otimes (C \otimes D), X) \\
&\qquad \cong \int^{Y,Z} \mathcal{C}(A \otimes B, Y) \times \mathcal{C}(C \otimes D, Z) \times \mathcal{C}(Y \otimes Z, X) \\
&\qquad \cong \int^{Y,Z} \mathrm{Bilin}(A, B; Y) \times \mathrm{Bilin}(C, D; Z) \times \mathrm{Bilin}(Y, Z; X).
\end{aligned}
$$

One may need to change the variance of modules if right and left representations are combined. Suppose $\mathcal{C}$ has both tensor products and cartesian binary products.

$$
\begin{aligned}
\mathcal{C}(A \otimes (B \times C), X) &\cong \textstyle\int^{Y} \mathcal{C}(B \times C, Y) \times \mathcal{C}(A \otimes Y, X) \\
&\cong \textstyle\int^{Y} \overline{\mathrm{Pairs}}(B, C; Y) \times \mathrm{Bilin}(A, Y; X)
\end{aligned}
$$

It is routine to generalise this to all combinations of right/left representation; namely, one can give a corresponding composite module to each functor (or more precisely, functor expression) which is constructed from right/left representations with the usual cartesian structure on **Cat**. Briefly, it goes as follows. Write right(P) (left(P)) for a right (left) representation of a right (left) representable module P.

Then, one can find a construction of the desired composite module in parallel to that of the composite left/right representation.

$$
\frac{\mathcal{A} \overset{P}{\relbar\!\!\mapstochar\!\!\rightarrow} \mathcal{B} \overset{Q}{\relbar\!\!\mapstochar\!\!\rightarrow} \mathcal{C}}{\mathrm{right}(Q)\mathrm{right}(P) \cong \mathrm{right}(Q \circ P), \qquad \mathrm{left}(P)\mathrm{left}(Q) \cong \mathrm{left}(Q \circ P).}
$$

$$
\frac{\mathcal{A} \overset{P}{\relbar\!\!\mapstochar\!\!\rightarrow} \mathcal{B} \overset{Q}{\leftarrow\!\!\mapstochar\!\!\relbar} \mathcal{C}}{\mathrm{left}(Q)\mathrm{right}(P) \cong \mathrm{left}(\overline{P}^{\mathcal{A},\mathcal{B}} \circ Q)}
$$

$$
\frac{\mathcal{A} \overset{P}{\leftarrow\!\!\mapstochar\!\!\relbar} \mathcal{B} \overset{Q}{\relbar\!\!\mapstochar\!\!\rightarrow} \mathcal{C}}{\mathrm{right}(Q)\mathrm{left}(P) \cong \mathrm{right}(Q \circ \overline{P}^{\mathcal{B},\mathcal{A}})}
$$

$$
\frac{\mathcal{A} \overset{P}{\leftarrow\!\!\mapstochar\!\!\relbar} \mathcal{C} \overset{Q}{\relbar\!\!\mapstochar\!\!\rightarrow} \mathcal{B}}{\langle \mathrm{right}(P), \mathrm{right}(Q) \rangle \cong \mathrm{right}(S), \qquad \text{where } S(A, B; C) = PAC \times QBC}
$$

$$\frac{\mathcal{A} \overset{P}{\longrightarrow\!\!\!\!\!\!\!+\;\;} \mathcal{C} \overset{Q}{\longrightarrow\!\!\!\!\!\!\!+\;\;} \mathcal{B}}{\langle \mathrm{left}(P), \mathrm{right}(Q) \rangle \cong \mathrm{right}(S), \qquad \text{where } S(A, B; C) = \overline{P}^{\mathcal{A},\mathcal{C}} AC \times QBC}$$

$$\frac{\mathcal{A} \overset{P}{\longrightarrow\!\!\!\!\!\!\!+\;\;} \mathcal{C} \overset{Q}{\;\;+\!\!\!\!\!\!\!\longleftarrow} \mathcal{B}}{\langle \mathrm{left}(P), \mathrm{left}(Q) \rangle \cong \mathrm{right}(S), \qquad \text{where } S(A, B; C) = \overline{P}^{\mathcal{A},\mathcal{C}} AC \times \overline{Q}^{\mathcal{B},\mathcal{C}} BC}$$

Chapter 3

Formalisation of representability of modules in LEGO

3.1 Introduction

This chapter presents our current implementation of a computer checked language for categories with universal structure. The language is to describe and reason about morphisms / objects of those categories. It is implemented in the LEGO proof development system [48,64,62] as a set of LEGO definitions and LEGO theorems. These can be classified into three kinds. The first provides the definitions of types for, *e.g.,* categories, modules, and representations. The second defines commands that construct and manipulate those entities; for example, given a module $P\colon \mathbf{1} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ with a representation R, a command `rIntro` constructs $X \longrightarrow R$ from an element $x \in PX$. The type checking by LEGO system guarantees that these operations are combined validly. This is in that loose sense that we have a language for categories; but if LEGO had a parser-generator for object level languages, *e.g.*, one with the Isabelle system ([60]), these commands provide a basis for a language with a more definitive look. The third proves properties of these manipulations, *e.g.,* that the above operation is bijective.

From the viewpoint of the refinement style proof development, the language works in the following manner. Assume we have formalised a category $\mathcal{C}$ with limits of shape $\mathcal{I}$, and we want to derive some morphism from an object A to the limit of a diagram h. The commands of the language allow us to derive such a morphism from a cone over h with the vertex A. The cone is a collection of hopefully simpler morphisms, so we have reduced our problem to a collection of simpler sub-problems. It is in this sense that the analysis of Chapter 2 may give rise to a type theory technique for deriving morphisms by refining a problem into simpler sub problems.

In this chapter, we review our current LEGO code that gives these definitions and theorems. As examples, it defines structure of binary product, terminal object, equalizers, and pullbacks. Preservation of structure is not yet implemented. Finally, as an example of using those structure, we give a construction of equalizers from terminal objects and pullbacks.

But first, we compare our work with that of Rydeheard and Burstall [71] in Section 3.2, give a summary of the LEGO system in Section 3.3, and consider general implementation issues in Section 3.4.

3.2 Comparison with "Computational Category Theory"

The most extensive mechanised treatment of universality is found in the book "Computational Category Theory" by Rydeheard and Burstall [71]. Using the ML functional programming language, they defined ML types of categories *etc.* and constructed functions that compute, for example, coproducts, coequalizers, and more generally finite colimits, adjunctions, *etc.* in the category of finite sets. Our computer checked language advances this in two respects

First, we uniformly construct our structures based on the abstract framework of universal structure. We provide a set of generic commands that work for all structure. Each structure $\mathcal{S}$ on $\mathcal{C}$ is obtained by constructing the parameter category $\mathrm{Param}(\mathcal{S})(\mathcal{C})$, the defining module $\mathrm{Module}(\mathcal{S})(\mathcal{C})$, and finally some abbreviations for the particular instances of generic commands, if necessarily. In contrast, Rydeheard and Burstall [71] represented each structure by functions specific to the structure. As the form of functions varies from structure to structure, the treatment was not uniform. For example, a binary product was represented by a product-cone and a 'universality function' $(f, g) \mapsto \langle f, g \rangle$, and a equalizer of (f, g) by a universal arrow e and a universality function that sends h with $fh = gh$ to $\bar{h}$ with $e\bar{h} = h$. There is little in common in the two cases.

Second, Rydeheard and Burstall [71] concentrated on construction of morphisms / objects in a category; the validity of the construction is checked informally. This was because ML's type system was too weak to express axioms involved, e.g., associativity of composition. (see Section 3.7 of [71]). Also, even for simple operations, *e.g.,* $(f, g) \mapsto fg$, it was a user's responsibility to ensure that the codomain of g is the same as the domain of f. In our implementation, the embedded logic of LEGO is used to express those axioms so that type checking of LEGO system guarantees the validity of construction. Also, dependent types allow us to specify operations with finer domains than ML types; *e.g.,* our composition of two arrows is a function from $\mathrm{Hom}(b, c) \times \mathrm{Hom}(a, b)$ to $\mathrm{Hom}(a, c)$, thus an attempt to compose an invalid pair of morphisms results in a type-checking error.

3.3 LEGO syntax and commands

Here, we give only a summary of syntax and commands necessary to follow the LEGO code given in Section 3.5. The full explanation can be obtained from the manuals [48,64,62]. We use LEGO in a mode that implements the Extended Calculus of Constructions [47].

Syntax LEGO's ASCII representation of ECC syntax is as follows. Here, v and T run over variables and terms, respectively.

`{v:T}T,{v|T}T`	$\Pi v{:}T.T$
`[v:T]T,[v|T]T`	$\lambda x{:}T.T$
`TT,T|T`	TT
<v:T>T	$\Sigma v{:}T.T$
`T.1`	$\pi_1(T)$
`T.2`	$\pi_2(T)$

The syntax `{v|T}T` is used for types of functions that trigger LEGO's *argument synthesis* feature. For example, given `f:{x|A}{y:B x}C`, `a:A`, and `M:B a`, one may form, as possibly not well-typed term, `f M`. The mark `{x|A}` triggers an attempt for the type checker to compute whatever application of type `A`. In this example, it first calculates the type of `M` and finds that it unifies with `B x` with `x`=`a` and accept `f M` as a well-typed term. The application of `a` is *implicit*. When one needs to supply a specific argument `b` for such an implicit application, the syntax `f|b` is used. A λ-term whose argument should be synthesised is written with the syntax `[v|T]T`, so `f` may be defined by `[x|A][y:B x]c` with `c:C`.

The type *Prop* and and a type universe $Type(n)$ are represented by `Prop` and `Type(`n`)`.

Commands LEGO has two interactive modes; the LEGO state manages the context in which a user works, and the proof state assists user to develop a proof. Somewhat simplified explanation for important commands is as follows.

- `[v:T];` – extends the current context by assuming a variable `v` of type `T`. (`T` must be of type `Prop` or `Type(`n`)`.)
- `[v = T];` or `v == T;` – defines the name `v` to be a term `T`. One can optionally specify the type of `T'` of `T` by `[v = T:T'];` or `v:T' == T;`.
- `Discharge v;` – discharges the variables introduced after `v` from the definitions made after the introduction of `v`.
- `Goal T;` – provided `T` is of type `Prop` or `Type(`n`)`, puts LEGO into proof state with the initial goal `?:T`.
- `Intros v;` – provided the current goal is of the form {`u:T_0`}`T_1`, introduces the variable `v` of type `T_0` and generates a new goal `?:T_1`[u:=v]. One may provide several variables to substitute several successive `Intros`.
- `Intros #;` – provided the current goal is of the form `<u:T_0>T_1`, generates two new goals `?:T_0` and `?':T_1`[u:=?]
- `Refine f;` – tries to unify the current goal with the type of `f`. When `f` is a function, it also tries `f`x, `f`xy, *etc.*, with free variables $x, y \ldots$; if this succeeds, the command generates new goals that correspond to uninstantiated free variables. Otherwise, it fails.
- `Qrepl t;` – provided `t` is of type `Eq M N`, replaces all occurrence of `M` in the current goal with `N`.
- `Equiv T;` – provided `T` is convertible with the current goal, replaces the current goal with `T`.
- `Save v;` – after successfully finishing a proof, puts LEGO into the LEGO state and adds the definition `[v=M]` to the current context, where `M` is the proof term.

We also use the extension of ECC to inductive types. Particularly, we use inductive product types and Σ types defined in the standard LEGO library [39]. The library allows a user to specify the level of type universe in which inductive types are defined. However, because of the two sizes of categories, we have to manage two levels of universes at once; namely `SET==Type(1)` for possibly large sets and `Set==Type(0)` for small sets. Accordingly, we have two versions of definitions for each inductive types, which we distinguish by a prefix `s` for the latter. For example, `prod:SET->SET->SET` defines the products of two large sets and `sprod:Set->Set->Set` defines those of small sets. Other than this prefixing `s`, most names taken from the library are self-explanatory.

Definition by proving The proof assistant of the LEGO system is used not only for checking correctness of a LEGO theorem. When `P` is of type `Type(`n`)` or `Prop`, rather than directly typing in a definition `[x=M:P]`, one may prove `P` with a specific proof that corresponds to `M` and then define `x` by `Save x` command. When `M` is a complex term (*e.g.*, several screenfuls), developing a corresponding proof with the above command is more convenient than typing it in directly. This use of proofs is not found in proof development systems where proofs are metatheoretic objects, *e.g.*, the Isabelle system [60]. In Section 3.5 we will note if a proof is used to define a specific term or to check correctness of a LEGO theorem.

3.4 General implementation issues

The size of categories Formalisation makes the problem of size more acute syntactically than it is in a usual mathematical argument. Usually, small sets and possibly large sets have operations on them with the same notation, and we need not have separate definitions for, *e.g.*, functor categories $[\mathcal{A}, \mathcal{B}]$ depending on the size of $\mathcal{A}$ and that of $\mathcal{B}$. In our formalisation, we are forced to have ramified definitions for various operations depending on the size of categories involved. In implementing ECC's cumulative universe hierarchy [47], the LEGO system provides some facility to address this issue of size, namely, type checking with typical ambiguity and universe polymorphism [31,63]. For example, a small category `C:Cat` can be type checked as a possibly large locally small category `C:CAT`. However, universe polymorphism is not flexible enough to avoid the above mentioned duplication. For example, there is no provision to make a single definition of `Fun`, the functor category forming operation, to work both as `Cat->Cat->Cat` and as `Cat->CAT->CAT`, depending on the size of the second argument.

Equality on Hom-sets In our implementation, a Hom-set of a category is formalised by a type in the universe `Set`. Given two parallel arrows, we compare them by the Leibniz equality. This simplifies the development, but leaves out the possibility of using a given equivalence relation as equality for morphisms. In the current implementation, we introduce *ad hoc* coercions from proofs of equivalence to those of the Leibniz equality when it is necessary. In future, we hope to adopt quotient types studied in Hofmann's thesis [35]. There, he gives a syntactic model of quotient types in an intensional type theory; this can be seen as providing a set of macros to treat a user defined equality. A more direct approach is to formalise a Hom-set by more complex structure than a type, *e.g*, a type paired with a equivalence relation. In the library [68] for the Coq system [20], some basic category theory is formalised in this way. We avoid the extra complexity since our development is largely independent of this detail. More precisely, it is pointed out that there is a difficulty in formalising the category **Cat** of small categories in this approach, but our current implementation needs only the types of categories and functors, *i.e.,* **Cat** is used metatheoretically; so, there should be little difficulty in adopting this approach.

Extensionality and proof irrelevance Extensionality for functions is needed, for example, to define the category `Setc` of small sets and the Hom-functors. As usual, we assumed the axiom `ext` of type `{K|Type}{L:K->Type}{f,g:{x:K}L x}({x:K}Eq (f x) (g x))->Eq f g`. Proof irrelevance is needed to formalise subsets; *e.g.*, natural transformations are those functions that satisfy a naturality condition. We circumvent the lack of proof irrelevance in the LEGO system by providing *ad hoc* coercions of equality of the superset to that of the subsets. Our assumption of extensionality and proof irrelevance may seem *ad hoc* and in conflict with the intensional type system of LEGO, which is decided for with computational consideration. Once again, we hope that Hofmann's work [35] can be applied to give more systematic solutions for these problems.

Inductive vs. Native Σ type LEGO and its standard library provide two version of Σ type; one natively implements Σ type of ECC and the other is inductively defined. Native Σ types have more concise syntax but lack η rule. Inductive ones have clumsier syntax but are equipped with elimination constants that prove (propositional) η rule. We use an inductive Σ type when we use equality on that type, and a native one otherwise. Although it is not a problem to assume propositional η rule for native Σ types as an axiom, elimination constants for inductive ones are found to be more flexible and easier to use than such an axiom. (Note: the current version of LEGO has the `Record` syntax sugar that avoids the clumsiness of inductive Σ types defined in the standard library.)

3.5 The LEGO implementation

Our LEGO code is divided into a few LEGO modules (= units of compilation, not to be confused with categorical modules).

`categories.l`	the basic category theory definitions
`modules.l`	the definition of modules, right and left representations, and generic commands for all structures
`termmod.l`	terminal objects
`prodmod.l`	binary products
`eqmod.l`	equalizers
`pbmod.l`	pullbacks
`eqex.l`	an example construction of equalizers from pullbacks and terminal objects

In this section, we review each module explaining the fine points of implementation.

3.5.1 Basic definitions

Categories First, we define the type of small categories. A category is represented by a dependent tuple of 1. an object set, 2. a Hom-set function, 3. a composition function, 4. a proof of left identity 5. that of right identity, and 6.

that of associativity. The Σ type used here is the native one, since we do not ask if two categories are equal.

```
(* Type of small categories *)
Cat ==
(* Signature *)
<Ob:Set>
<H:Ob->Ob->Set>
<id:{x:Ob}(H x x)>
<o:{x,y,z|Ob}(H y z)->(H x y)->(H x z)>
(* Axioms *)
<ul:{x,y|Ob}{f:(H x y)}(Eq f (o (id y) f))>
<ur:{x,y|Ob}{f:(H x y)}(Eq f (o f (id x)))>
{x,y,z,w|Ob}{f:(H z w)}{g:(H y z)}{h:(H x y)}
        (Eq (o f (o g h)) (o (o f g) h));
```

For each component of any Σ type, we define a selector. The name of the selector should usually be the same as the variable name used in the definition. It is vital for the selector to be appropriately type cast using those selectors on which it depends; otherwise, the whole development would be swamped by meaningless expressions like `X.2.2.`$\cdots$`.2.1`. Careful use of argument synthesis is also important to mimic the usual notation.

```
Ob      == [X:Cat] X.1:Cat->Set;

[X|Cat];
H       == X.2.1:(Ob X)->(Ob X)->Set;
id      == X.2.2.1:{x:Ob X}(H x x);
o       == X.2.2.2.1:{x,y,z|Ob X}(H y z)->(H x y)->(H x z);
ul      == X.2.2.2.2.1:{x,y|Ob X}{f:(H x y)}(Eq f (o (id y) f));
ur      == X.2.2.2.2.2.1:{x,y|Ob X}{f:(H x y)}(Eq f (o f (id x)));
o_a     == X.2.2.2.2.2.2
        :{x,y,z,w|Ob X}{f:(H z w)}{g:(H y z)}{h:(H x y)}
        (Eq (o f (o g h)) (o (o f g) h));
o_b     == [x,y,z,w|Ob X]
           [f:H z w][g:H y z][h:H x y]Eq_sym (o_a f g h);
        (* assoc. for the reverse direction *)
Discharge X ;
```

Henceforth, we omit the listing of such selectors. (Note: `Record` syntax of the current version of LEGO would make needs for such definitions obsolete.)

As explained in the previous section, we also need a separate definition of the type `CAT` for locally small but possibly large categories. We omit the definition, since the only difference from that of `Cat` is that the type of object set is `SET` rather than `Set`. The names of the selectors for `CAT` are the same as those for `Cat` except for a prefix `B`, *e.g.*, `BH:{X|CAT}(BOb X)->(BOb X)->Set`. ECC's universe hierarchy means that any small category `C:Cat` is also a large category `C:CAT`.

Next we define the large category `Setc` whose objects are sets, *i.e.*, types `T:Set` and whose morphisms are functions between them. The definition is given by a specific proof of `CAT`, rather than typing it in directly.

```
(* Category Setc of Set *)
Goal CAT;
Intros #; Refine Set;                    (* Ob Setc *)
Intros # x y; Refine x->y;               (* Hom set *)
Intros # x; Refine [T:Type][t:T]t;       (* identity *)
Intros # x y z f g h; Refine f(g(h));    (* composition *)
Intros # x y f;Refine ext [u:x]y;Intros z;Refine Eq_refl;
                                         (* left id *)
Intros # x y f;Refine ext [u:x]y;Intros z;Refine Eq_refl;
                                         (* right id *)
Intros x y z w f g h;Refine Eq_refl;     (* associativity *)

Save Setc;
```

One can examine the result by

```
Lego> Setc;
value of Setc =
        (Set,
        [x,y:Set]x->y,
        [x:Set]([T:Type][t:T]t) x,
        [x,y,z|Set]
          [f:([x'4,y'5:Set]x'4->y'5) y z]
          [g:([x'5,y'6:Set]x'5->y'6) x y]
          [h:x]
          f (g h),
              ⋮
type of Setc  = CAT
```

Note that to prove the identity laws for `Setc`, one needs extensionality axiom `ext`. For this instance, it is possible to avoid the use of `ext` by redefining the sense of equality for functions, as in [52].

The opposite category is obtained by reversing the order of arguments for the Hom-set function `H`.

```
(* Opposite category (small) *)
[A:Cat];
Goal Cat;
Intros #; Refine Ob A;                   (* objects *)
Intros #; Refine [x:Ob A][y:Ob A]H y x;  (* morphisms *)
Intros #; Refine [x:Ob A]id x;           (* identity *)
```

```
Intros # x y z f g; Refine o g f;       (* composition *)
  ⋮
Save opcat;
Discharge A;
```

During the development, we used some templates for proof components of a small category to speed up compilation once actual proofs are done. For example,

```
(* constants to substitute real proofs *)
[claim_ul:
        {Ob:Set}{H:Ob->Ob->Set}{id:{x:Ob}H x x}
        {o:{x,y,z|Ob}(H y z)->(H x y)->H x z}
        {x,y|Ob}{f:H x y}(Eq f (o (id y) f))];
```

They also proved to be useful for "rapid prototyping"; in our uniform treatment, most proofs fall into the two patterns of proving that something is a category or a functor, so a small number of templates can be used widely.

Functors and natural transformations Now we turn to our definitions of functors and natural transformations. Depending on the size of domain and codomain categories, at least two types for functors are needed; `Fun0` for those from a small category to a large category and `ssFun0` for those from a small category to a small category. Although there is an implicit type inclusion from `ssFun0` to `Fun0`, the former is needed to properly type functions that send functors to functors.

Similarly to `Cat`, a functor is represented by a dependent tuple of 1. an object-part function, 2. a morphism-part function, 3. a proof of identity preservation, and 4. that of composition preservation. We list here only the definition of `Fun0`.

```
(* Type of functors (small -> big) *)
[A:Cat][B:CAT];
Fun0:SET ==
        <O:(Ob A)->(BOb B)>
        <M:{x,y|(Ob A)}(H x y)->(BH (O x) (O y))>
        <pid:{x:(Ob A)}(Eq (M (id x)) (Bid (O x)))>
        {x,y,z:Ob A}{f:(H y z)}{g:(H x y)}
             (Eq (M (o f g)) (Bo (M f) (M g))) ;
Discharge A;
```

The selectors for `Fun0`'s four components are named as `O`, `M`, `pid`, `pcomp`; those for `ssFun0` are `sO`, `sM`, `spid`, `spcomp`. The names of templates for the proof components of a functor are `claim_pid` and `claim_pcomp`.

In the usual notation, a functor $F\colon \mathcal{A} \longrightarrow \mathcal{B}$ is at the same time $F\colon \mathcal{A}^{\mathrm{op}} \longrightarrow \mathcal{B}^{\mathrm{op}}$, but we need explicit operation `opfun` for this.

```
value of opfun = ···
type of opfun  = {A,B|Cat}(ssFun0 A B)->ssFun0 (opcat A) (opcat B)
```

Given `F,G:Fun0 A B`, a natural transformation `t:Nat F G` consists of a function `cpt t` that sends an object of `A` to an morphism of `B`, together with a proof `natcom t` of the usual commutative square.

```
(* Type of natural transformations *)
value of Nat = [A|Cat][B|CAT][F,G:Fun0 A B]
                <cpt:{x:Ob A}BH (O F x) (O G x)>
                {x,y|Ob A}{f:H x y}
                Eq (Bo (cpt y) (M F f)) (Bo (M G f) (cpt x))
type of Nat  = {A|Cat}{B|CAT}(Fun0 A B)->(Fun0 A B)->Set
```

Then we proceed to give other basic definitions of functor calculus, but we just list the type of each defined terms.

```
  (* Composition of natural transformations *)
  ott = ... : {A|Cat}{B|CAT}{F,G,K|Fun0 A B}
               (Nat G K)->(Nat F G)->Nat F K
  (* Identity functor *)
  id' = ... : {A:Cat}ssFun0 A A
  (* Identity natural transformation *)
  idt = ... : {A|Cat}{B|CAT}{F:Fun0 A B}Nat F F
  (* Composition of functors of various size *)
  fun_sfun_comp = ...
      : {A,B|Cat}{C|CAT}(Fun0 B C)->(ssFun0 A B)->Fun0 A C
  sfun_sfun_comp = ...
      : {A,B,C|Cat}(ssFun0 B C)->(ssFun0 A B)->ssFun0 A C
```

To define a functor category, one needs equality for natural transformation. We introduce a constant to ignore the difference in the proof part.

```
  natEq : {A|Cat}{B|CAT}{F,G|Fun0 A B}{t,s:Nat F G}
          ({x:Ob A}Eq (cpt t x) (cpt s x))->Eq t s
```

Now the category `Fun A B:CAT` of functors of type `Fun0 A B` can be defined as follows.

```
(* Functor category *)

[A:Cat][B:CAT];
Goal CAT;
```

```
Intros #; Refine Fun0 A B;                         (* objects *)
Intros # F G; Refine Nat F G;                      (* morphisms *)
Intros # F; Refine idt;                            (* identity *)
Intros # F G H; Refine ott;                        (* composition *)
Intros # F G t; Refine natEq; Intros x; Refine Bul; (* left id *)
Intros # F G t; Refine natEq; Intros x; Refine Bur; (* right id *)
Intros F G K L s t u; Refine natEq; Intros x; Refine Bo_a;
                                                   (* assoc *)
Save Fun;
Discharge A;
```

Product categories To define a module $P\colon \mathcal{A} \longrightarrow\!\!\!\!\!\!+\ \mathcal{C}$ as a functor from $\mathcal{C}^{\mathrm{op}} \times \mathcal{A} \longrightarrow \mathbf{Set}$, we provide the definition of the product category `prodcat A B` for small categories `A,B:Cat`.

```
(* Product categories *)
[A,B:Cat];
Goal Cat;
Intros #; Refine sprod (Ob A) (Ob B);              (* object *)
Intros # x y;                                      (* morphism *)
Refine sprod (H (sFst x) (sFst y)) (H (sSnd x) (sSnd y));
Intros # x;                                        (* identity *)
Refine sPair (id (sFst x)) (id (sSnd x));
Intros # x y z f g;                                (* composition *)
Refine sPair (o (sFst f) (sFst g)) (o (sSnd f) (sSnd g));
...
Save prodcat;
Discharge A;
```

We also define the product of two functors, the currying of functors from `prodcat A B` to `C`; the types of these operations are:

```
type of prodfun  = {A,B,C,D|Cat}(ssFun0 A C)->(ssFun0 B D)
                         ->ssFun0 (prodcat A B) (prodcat C D)
type of curr  = {A,B|Cat}{C|CAT}
                 (Fun0 (prodcat A B) C)->Fun0 A (Fun B C)
```

Natural isomorphisms A natural isomorphism is represented by a dependent pair of a natural transformation `t` and a proof that each component of `t` is an isomorphism.

```
(* Natural isomorphism *)
value of IsoNat = [A|Cat][B|CAT][F,G:Fun0 A B]
                  <t:Nat F G>{x:Ob A}isIso (cpt t x)
type of IsoNat  = {A|Cat}{B|CAT}(Fun0 A B)->(Fun0 A B)->Set
```

The selector for the first component is `isonat`.

```
type of isonat  = {A|Cat}{B|CAT}{F,G|Fun0 A B}(IsoNat F G)->Nat F G
```

A lemma provides the construction of the inverse natural transformation from a natural isomorphism

```
value of isonatinv = ···
type of isonatinv  = {A|Cat}{B|CAT}{F,G|Fun0 A B}
                     (IsoNat F G)->Nat G F
```

Hom functors Our last basic definition is for Hom functors:

```
(* Hom functors *)
[A|Cat];

Goal Fun0 (prodcat (opcat A) A) Setc;
Intros # xy; Refine H|A (sFst xy) (sSnd xy);    (* obj part *)
Intros # xy zw fg h; Refine o|A; Refine +1 o|A; (* mor part *)
Refine +3 h; Refine (sSnd fg); Refine (sFst fg);
      ⋮
Save Hom;
Discharge A;

type of Hom  = {A|Cat}Fun0 (prodcat (opcat A) A) Setc
```

3.5.2 Modules, representations, and generic commands

This section reviews our LEGO implementation of modules and representations, and generic commands to manipulate them.

Modules The type `Mod0 A C` of modules from a small category `A` to another `C` is simply that of functors from $C^{\mathrm{op}} \times A$ to `Setc`. So, the type of module morphisms is given by that of natural transformation.

```
(* Type of module *)
Mod0 [A,C:Cat] == Fun0 (prodcat (opcat C)  A) Setc:SET;
Mod1 [A,C|Cat][P,Q:Mod0 A C] == Nat P Q:Set;
Mod [A,C:Cat] == Fun (prodcat (opcat C) A) Setc:CAT;
```

Now we define the two operations that send $F: \mathcal{A} \longrightarrow \mathcal{C}$ to F_* and F^* considered in Proposition 2.2.11 by $F_* = \mathcal{A}(-,?) \circ (\mathrm{id}_{\mathcal{A}}^{\mathrm{op}} \times F)$ and similarly for F_*. By the duality of definition, we can use F_* to give to give $F^* = (F^{\mathrm{op}})_*$, but we found it too clumsy to put extra `opfun` and to convert `(opcat (opcat A))` to `A`, *etc.* So, in our implementation, we give explicit symmetric definitions for the dual cases. We only list those definition concerning right representations.

```
(* Cat --> Mod *)
value of fun_rMod
        = [A,C|Cat][F:ssFun0 A C]
          fun_sfun_comp (Hom|C) (prodfun (id' (opcat C)) F)
type of fun_rMod  = {A,C|Cat}(ssFun0 A C)->Mod0 A C
```

Right / left representations Given a module `P:Mod0 A C` and a functor `R:Fun0 A C`, `rRep P R` is the type of natural isomorphism P$\cong$ R$_*$. A right representation can be given as an element of the Σ type `<R:Fun0 A C>rRep P R`, but we found that it is slightly more convenient to have R and the natural isomorphism separately.

```
(* Representability of modules *)

[A,C|Cat][P:Mod0 A C][Q:Mod0 C A][F:ssFun0 A C];

rRep == IsoNat P (fun_rMod F);
lRep == IsoNat Q (fun_lMod F);

Discharge A;
```

Generic commands and lemmas Our uniform treatment of all universal structure means that we can give generic commands and lemmas about them at this level of generality. The commands themselves are simple terms; they re-package and conveniently type cast the natural isomorphism of a right or left representation. To use a command, a user refine a goal with such a term.

The main generic commands are `rIntro` and `rElim`. Given a module $P\colon \mathcal{A} \longrightarrow\!\!\!\!\!\!+ \mathcal{C}$ and a functor $F\colon \mathcal{A} \longrightarrow \mathcal{C}$ which right represents P as $\rho\colon F_* \cong P$, the command `rIntro`ρ produce the morphism $C \longrightarrow FA$ uniquely determined by an $x \in P(C, A)$ ($C \in \mathcal{C}$ and $A \in \mathcal{A}$); `rElim` ρ is for the reverse direction. They are so named since they correspond to introduction rules and elimination rules of type theory, respectively.

```
(* Introduction and Elimination *)
value of rIntro = [A,C|Cat][P|Mod0 A C][F|ssFun0 A C]
                  [p:rRep P F][c:Ob C][a:Ob A][x:O P (sPair c a)]
                  cpt (isonat p) (sPair c a) x
type of rIntro  = {A,C|Cat}{P|Mod0 A C}{F|ssFun0 A C}
```

```
                 (rRep P F)->{c:Ob C}{a:Ob A}(O P (sPair c a))
                 ->H c (sO F a)
value of rElim = [A,C|Cat][P|Mod0 A C][F|ssFun0 A C]
                 [p:rRep P F][c:Ob C][a:Ob A][f:H c (sO F a)]
                 cpt (isonatinv p) (sPair c a) f
type of rElim  = {A,C|Cat}{P|Mod0 A C}{F|ssFun0 A C}
                 (rRep P F)->{c:Ob C}{a:Ob A}(H c (sO F a))
                 ->O P (sPair c a)
```

Next we prove lemmas that prove these are bijective, injective, natural, *etc.*. These unwind the definition of isomorphisms instantiated in `Setc` and give a typing easier to use. We only list the types of those lemmas.

```
type of rIEbij  = {A,C|Cat}{P|Mod0 A C}{F|ssFun0 A C}
                  {p:rRep P F}{c:Ob C}{a:Ob A}{f:H c (sO F a)}
                  Eq (rIntro p c a (rElim p c a f)) f
type of rEIbij  = {A,C|Cat}{P|Mod0 A C}{F|ssFun0 A C}
                  {p:rRep P F}{c:Ob C}{a:Ob A}{x:O P (sPair c a)}
                  Eq (rElim p c a (rIntro p c a x)) x
type of rElim_inj  = {A,C|Cat}{P|Mod0 A C}{F|ssFun0 A C}
                     {p:rRep P F}{c:Ob C}{a:Ob A}{f,g:H c (sO F a)}
                     (Eq (rElim p c a f) (rElim p c a g))->Eq f g
type of rI_nat1  = {A,C|Cat}{P|Mod0 A C}{F|ssFun0 A C}
                   {p:rRep P F}{c:Ob C}{a:Ob A}{d:Ob C}
                   {h:H d c}{x:O P (sPair c a)}[ha=sPair h (id a)]
                Eq (rIntro p d a (M P ha x)) (o (rIntro p c a x) h)
```

3.5.3 Binary products

Given the definitions and lemmas we reviewed so far, one can develop LEGO code for a specific structure in a uniform way by simply giving a module that defines the structure. Our first example is binary products structure. The following code provides the definition of module Pairs($\mathcal{C}$): $\mathcal{C} \times \mathcal{C} \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ in Section 2.4. Due to some name conflict with the standard library, here the module is called `prodmod`.

Assuming `C:Cat` is given, `prodmod` is defined as follows.

```
[C:Cat];
(* the module for product functor *)

(* the object part *)
[prodmod0 =
        [zxy:Ob (prodcat (opcat C) (prodcat C C))]
        sprod (H|C (sFst3 zxy) (sSnd3 zxy))
                    (H|C (sFst3 zxy) (sThd3 zxy))];
(* the morphism part *)
```

```
[prodmod1 =
        [zxy,wuv|Ob (prodcat (opcat C) (prodcat C C))]
        [hfg:H zxy wuv]
        [kl:prodmod0 zxy]
        sPair (o (sSnd3 hfg) (o (sFst kl) (sFst3 hfg)))
              (o (sThd3 hfg) (o (sSnd kl) (sFst3 hfg)))];

Goal Mod0 (prodcat C C) C;
Intros #; Refine prodmod0;
Intros #; Refine prodmod1;
⋮
Save prodmod;
Discharge C;
```

One may also derive from the generic commands some abbreviation specific to binary products for ease of use. For example, `rIntro` command can be re-typed as below.

```
timesIntro = ... : C|Cat
        times|ssFun0 (prodcat C C) C(rRep (prodmod C) times)
        ->{a,b,x:Ob C}
        (H x a)->(H x b)->H x (sO times (sPair a b))
```

The simplest use is as follows. In the context defined in the above display, to solve the goal $\mathcal{C}(\mathtt{x}, \mathtt{a} \times \mathtt{b})$, a user can type in `Refine timesIntro p;`.

```
  ?13 : H x (O times (sPair a b))
Lego> Refine timesIntro p;
Refine by timesIntro p
  ?19 : H x a
  ?20 : H x b
```

3.5.4 Terminal objects

The next example is the structure of terminal objects. The terminal object in a category $\mathcal{C}$ is defined by the module `termmod` $: \mathbf{1} \longrightarrow\!\!\!\!\!\!+\ \mathcal{C}$ which sends every $C \in \mathcal{C}$ to the singleton set.

We first define the terminal category `termcat`.

```
(* the terminal category *)

Goal Cat;
Intros #; Refine sunit;
Intros # x y; Refine sunit;
Intros # x; Refine svoid;
Intros # x y z f g; Refine svoid;
```

```
⋮
Save termcat;
```

; Similarly, the definition of `termmod` is given by

```
(* the module for terminal objects *)
[C:Cat];

Goal Mod0 termcat C;
Intros # x; Refine sunit;
Intros # x y h; Refine [t:sunit]t;
⋮
Save termmod;
Discharge C;
```

An abbreviation of `rIntro` is

```
termIntro = ... : {C|Cat}
        {termfun:ssFun0 termcat C}(rRep (termmod C) termfun)
        ->{x:Ob C}H x (s0 termfun svoid)
```

We also give a lemma that proves any two morphism from an object X to a terminal object T in $\mathcal{C}$ are the same. Given two morphisms $f, g: X \longrightarrow T$, both are determined by the unique element of the singleton set $\mathtt{termmod}(X, *)$, and hence are the same.

```
(* All morphisms into the terminal object are the same *)
[C|Cat];
[termfun:ssFun0 termcat C][p:rRep (termmod C) termfun];
[x:Ob C][f,g:H x (s0 termfun svoid)];

Goal Eq f g;
Refine rElim_inj p;
Refine all_sunitvoid; Refine Eq_sym; Refine all_sunitvoid;

Save sh_unique;

Discharge C;
```

3.5.5 Equalizers

LEGO code for equalizers is developed in the same manner. The module that define the equalizer structure on $\mathcal{C}$ is Ker: $[\mathbf{Para}, \mathcal{C}] \longrightarrow\!\!\!\!\!\!+\ \mathcal{C}$ given in Section 2.4. As before, we first construct the domain category `paracat C` for a category `C` based on the concrete description of $[\mathbf{Para}, \mathcal{C}]$ given in Section 2.4. This may be given as a functor category, but a direct definition is found to be easier to use. First, we define the type of objects.

```
[C:Cat];
(* The category of parallel pairs *)
(* Objects *)
paracat0:Set == sSigma (Ob C)
                ([x:Ob C] sSigma (Ob C)
                ([y:Ob C] sprod (H x y)(H x y)));
```

The four selectors are `paradom`, `paracod`, `parafst`, and `parasnd`. Given `f,g:H x y`, the constructor `paracons0 f g` gives an element of `paracat0` using argument synthesis.

Hom-sets are given by

```
(* Hom-sets *)
[fg,kl:paracat0];
paracat1
        == sSigma (H|C (paradom fg) (paradom kl))
           ([u:(H|C (paradom fg) (paradom kl))]
           sSigma (H|C (paracod fg) (paracod kl))
           ([v:(H|C (paracod fg) (paracod kl))]
            (and (Eq (o v (parafst fg)) (o (parafst kl) u))
                (Eq (o v (parasnd fg)) (o (parasnd kl) u)))));
Discharge fg;
```

The selectors are `paramor0`, `paramor1`, `parmorcom0`, and `paramorcom1`.

Next, we define identities and composition. These are given by specific proofs of appropriate types.

```
(* Identities *)
Goal {fg:paracat0}paracat1 fg fg;
Intros fg;
Refine sdep_pair; Refine id;
Refine sdep_pair; Refine id;
Refine pair;
Qrepl (Eq_sym (ul (parafst fg))); Qrepl (Eq_sym (ur (parafst fg)));
Refine Eq_refl;
Qrepl (Eq_sym (ul (parasnd fg))); Qrepl (Eq_sym (ur (parasnd fg)));
Refine Eq_refl;

Save paraid;

(* The composition *)
Goal {fg,ij,kl:paracat0}{st:paracat1 ij kl}{uv:paracat1 fg ij}
        paracat1 fg kl;
Intros fg ij kl st uv;
Refine sdep_pair; Refine o (paramor0 st) (paramor0 uv);
Refine sdep_pair; Refine o (paramor1 st) (paramor1 uv);
Refine pair;
```

```
Refine o_b; Qrepl Eq_sym (paramorcom0 st);
Refine o_a; Qrepl Eq_sym (paramorcom0 uv);
Refine o_b; Refine Eq_refl;
Refine o_b; Qrepl Eq_sym (paramorcom1 st);
Refine o_a; Qrepl Eq_sym (paramorcom1 uv);
Refine o_b;

Save paracomp;
```

The category `paracat:Cat` is obtained by packing these definitions.

Then we define the module `eqmod:Mod0 (paracat C) C`, which is called Ker in Section 2.4. The object part are as follows, with the selectors `eqmod0mor` and `eqmod0prf`.

```
(* The module for equalizers *)
(* The object part *)
[x:Ob C][fg:paracat0];
eqmod0 == sSigma (H x (paradom fg))
          ([h:H x (paradom fg)]
               Eq (o (parafst fg) h) (o (parasnd fg) h));
Discharge x;
```

The morphism part is given by the following specific proof.

```
(* The morphism part *)
[x,y:Ob C][fg,kl:paracat0][w:H y x][uv:paracat1 fg kl];
[hp:eqmod0 x fg];
Goal  eqmod0 y kl;
Refine sdep_pair;
Refine (o (o (paramor0 uv) (eqmod0mor hp)) w);
Qrepl o_a (parafst kl) (o (paramor0 uv) (eqmod0mor hp)) w;
Qrepl o_a (parafst kl) (paramor0 uv) (eqmod0mor hp);
Qrepl Eq_sym (paramorcom0 uv);
Qrepl o_b (paramor1 uv) (parafst fg) (eqmod0mor hp);
Qrepl eqmod0prf hp;
Qrepl o_a (paramor1 uv) (parasnd fg) (eqmod0mor hp);
Qrepl paramorcom1 uv;
Qrepl o_a (parasnd kl) (o (paramor0 uv) (eqmod0mor hp)) w;
Qrepl o_a (parasnd kl) (paramor0 uv) (eqmod0mor hp);
Refine Eq_refl;

Save eqmod1;
Discharge x;
```

The module `eqmod` is defined by packing these definitions with appropriate re-typing.

```
(* Packing these definitions to the module eqmod *)
eqmod0' [xfg:Ob (prodcat (opcat C) paracat)]
        == eqmod0 (sFst xfg) (sSnd xfg);
eqmod1' [xfg,ykl:Ob (prodcat (opcat C) paracat)]
```

```
        [wuv:H xfg ykl]
        [hp:eqmod0' xfg]
        == eqmod1 (sFst xfg) (sFst ykl) (sSnd xfg) (sSnd ykl)
                (sFst wuv) (sSnd wuv) hp;

Goal Mod0 paracat C;
Intros #; Refine eqmod0';
Intros #; Refine eqmod1';
⋮
Save eqmod;
```

Finally, we introduce an abbreviation for the instance of `rIntro` for `eqmod`.

```
value of eqlIntro = ···
type of eqlIntro  = {C|Cat}
        {eql:ssFun0 (paracat|C) C}(rRep (eqmod|C) eql)
        ->a,b|Ob Cf,g:H a bx:Ob C(eqmod0 x (paracons0 f g))
        ->H x (s0 eql (paracons0 f g))
```

3.5.6 Pullbacks

First, given a category $\mathcal{C}$, we define the domain category `pbcat` $\mathcal{C}$ of the defining module `pbmod`: `pbcat`$(\mathcal{C}) \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$. `pbcat`$(\mathcal{C})$ is the functor category $[\cdot \rightarrow \cdot \leftarrow \cdot, \mathcal{C}]$; but for the same reason as $[\mathbf{Para}, \mathcal{C}]$, we define the small category `pbcat`$(\mathcal{C})$ directly. An object of `pbcat`$(\mathcal{C})$ is a pair of morphisms $A \xrightarrow{f} C \xleftarrow{g} B$ with the same codomain. In LEGO, this is represented by a dependent tuple of 1. the domain of f, 2. the domain of g, 3. the codomain of f and g, and 4. the pair (f, g). The morphism of `pbcat`$(\mathcal{C})$ from $A \xrightarrow{f} C \xleftarrow{g} B$ to $D \xrightarrow{k} F \xleftarrow{g} E$ is a triple $(u\colon A \longrightarrow D, v\colon B \longrightarrow E, w\colon C \longrightarrow F)$, such that two squares below commutes.

$$
\begin{tikzcd}
A \arrow[r, "f"] \arrow[d, "u"'] & C \arrow[d, "w"] & B \arrow[l, "g"'] \arrow[d, "v"] \\
D \arrow[r, "k"'] & F & E \arrow[l, "l"]
\end{tikzcd}
$$

In LEGO, this is represented by a dependent tuple of u, v, w, and the proof that the two squre commutes. The identity and composition are defined in the obvious way, but in LEGO one has to supply the proof that they make relevant squares commute. The type of objects are given as follows.

```
[C:Cat];
```

```
(* Domain cat of pbmod *)
(* The objects *)
pbcat0:Set == sSigma (Ob C)
        ([a:Ob C] sSigma (Ob C)
        ([b:Ob C] sSigma (Ob C)
        ([c:Ob C] sprod (H a c) (H b c))));
```

The selectors for the five components are `pbcat0a`, `pbcat0b`, `pbcat0c`, `pbcat0fst`, and `pbcat0snd`. Given `f:H a c` and `g:H b c`, the constructor `pbcat0cons f g` gives an element of `pbcat0`. The Hom-sets are given by

```
(* Hom-sets *)
pbcat1 [fg,kl:pbcat0]
       == sSigma (H (pbcat0a fg) (pbcat0a kl))
       ([u:(H (pbcat0a fg) (pbcat0a kl))]
        sSigma (H (pbcat0b fg) (pbcat0b kl))
       ([v:(H (pbcat0b fg) (pbcat0b kl))]
        sSigma (H (pbcat0c fg) (pbcat0c kl))
       ([w:(H (pbcat0c fg) (pbcat0c kl))]
       (and (Eq (o w (pbcat0fst fg)) (o (pbcat0fst kl) u))
            (Eq (o w (pbcat0snd fg)) (o (pbcat0snd kl) v)))))):Set;
```

with the selectors `pbcat1a`, `pbcat1b`, `pbcat1c`, `pbcat1com0`, and `pbcat1com1`. The identities and the composition are defined by the following proof.

```
(* Identities *)
[fg:pbcat0];
Goal pbcat1 fg fg;
Refine sdep_pair; Refine id (pbcat0a fg);
Refine sdep_pair; Refine id (pbcat0b fg);
Refine sdep_pair; Refine id (pbcat0c fg);
Refine pair;
Qrepl Eq_sym (ul (pbcat0fst fg)); Qrepl Eq_sym (ur (pbcat0fst fg));
Refine Eq_refl;
Qrepl Eq_sym (ul (pbcat0snd fg)); Qrepl Eq_sym (ur (pbcat0snd fg));
Refine Eq_refl;

Save pbcatid;
Discharge fg;

(* Composition *)
[fg,kl,ij:pbcat0] [uvw:pbcat1 kl ij] [rst:pbcat1 fg kl];
Goal pbcat1 fg ij;
Refine sdep_pair; Refine o (pbcat1a uvw) (pbcat1a rst);
Refine sdep_pair; Refine o (pbcat1b uvw) (pbcat1b rst);
Refine sdep_pair; Refine o (pbcat1c uvw) (pbcat1c rst);
Refine pair;
Refine o_b; Qrepl Eq_sym (pbcat1com0 uvw);
Refine o_a; Qrepl Eq_sym (pbcat1com0 rst);
Refine o_b;
Refine o_b; Qrepl Eq_sym (pbcat1com1 uvw);
Refine o_a; Qrepl Eq_sym (pbcat1com1 rst);
Refine o_b;
```

```
Save pbcatcomp;
Discharge fg;
```

The category `pbcat C` for `C:Cat` is obtained by packing these definitions.

The module $\mathtt{pbmod}(\mathcal{C})\colon \mathtt{pbcat}(\mathcal{C}) \longrightarrow\!\!\!\!\!\!\!+\;\; \mathcal{C}$ sends $(X \in \mathcal{C}, (f, g) \in \mathtt{pbcat}(\mathcal{C}))$ to the set of pairs of morphisms (p, q) such that the following diagram commutes.

$$\begin{tikzcd} X \arrow[r, "p"] \arrow[d, "q"'] & A \arrow[d, "f"] \\ B \arrow[r, "g"] & C \end{tikzcd}$$

This is represented in LEGO by a dependent tuple of p, q, and the proof that $fp = gq$. Given $t\colon Y \longrightarrow X$ in $\mathcal{C}$ and $(u, v, w)\colon (f, g) \longrightarrow (k, l)$ in $\mathtt{pbcat}(\mathcal{C})$, the morphism part of $\mathtt{pbmod}(\mathcal{C})$ sends $(p, q) \in \mathtt{pbmod}(\mathcal{C})(X, (f, g))$ to (upt, vqt). The conditions on (p, q) and (u, v, w) ensure that (upt, vqt) is in $\mathtt{pbmod}(\mathcal{C})(Y, (k, l))$. The proof of this fact in LEGO supplies the proof part of the representation of (upt, vqt) in LEGO. A representation $F\colon \mathtt{pbcat}(\mathcal{C}) \longrightarrow \mathcal{C}$ then sends (f, g) to the vertex of a pullback square for f and g with the counit giving the two morphisms that complete the square. First, the object part is given by

```
(* The module pbmod *)
(* The object part, a temporary definition *)
pbmod0 [a,b,c:Ob C][f:H a c][g:H b c][x:Ob C]
        == sSigma (H x a)
        ([p:H x a] sSigma (H x b)
        ([q:H x b] Eq (o f p) (o g q))):Set;
```

with the selectors `pbmod0p`, `pbmod0q`, and `pbmod0r` and

```
(* The object part *)
pbmod0' [xfg:Ob (prodcat (opcat C) pbcat)]
        == [fg = sSnd xfg]
                  pbmod0 (pbcat0a fg)
                  (pbcat0b fg)
                  (pbcat0c fg)
                  (pbcat0fst fg)
                  (pbcat0snd fg)
                  (sFst xfg):Set;
```

Given `f:H a c`, `g:H b c`, `p:H x a`, `q:H x b`, and `t:Eq (o f p) (o g q)`, the constructor `pbmod0cons f g p q t` gives an element of `pbmod0`. The morphism part is defined by

```
(* The morphism part, a temporary definition *)
[x:Ob C][fg:Ob pbcat];
[y:Ob C][kl:Ob pbcat];
[t:H y x][uvw:pbcat1 fg kl];
[pqr:pbmod0' (sPair x fg)];
Goal pbmod0' (sPair y kl);
Refine sdep_pair; Refine o (pbcat1a uvw) (o (pbmod0p pqr) t);
Refine sdep_pair; Refine o (pbcat1b uvw) (o (pbmod0q pqr) t);
Qrepl o_a (pbcat0fst kl) (pbcat1a uvw) (o (pbmod0p pqr) t);
Qrepl o_a (pbcat0snd kl) (pbcat1b uvw) (o (pbmod0q pqr) t);
Qrepl Eq_sym (pbcat1com0 uvw);
Qrepl Eq_sym (pbcat1com1 uvw);
Qrepl o_b (pbcat1c uvw) (pbcat0fst fg) (o (pbmod0p pqr) t);
Qrepl o_b (pbcat1c uvw) (pbcat0snd fg) (o (pbmod0q pqr) t);
Qrepl o_a (pbcat0fst fg)(pbmod0p pqr) t;
Qrepl o_a (pbcat0snd fg)(pbmod0q pqr) t;
Qrepl pbmod0r pqr;
Refine Eq_refl;

Save pbmod1;
Discharge x;

(* The morphism part *)
pbmod1' [xfg,ykl:Ob (prodcat (opcat C) pbcat)]
        [tuvw:H xfg ykl]
        [pqr:pbmod0' xfg]
        ==
        pbmod1 (sFst xfg) (sSnd xfg) (sFst ykl) (sSnd ykl)
               (sFst tuvw) (sSnd tuvw) pqr:pbmod0' ykl;
```

The module `pbmod` is defined by packing these definitions.

Finally, we provide some abbreviations for `rIntro` instantiated with `pbmod`.

```
value of pbIntro = ···
type of pbIntro  = {C|Cat}
        {pb:ssFun0 (pbcat C) C}(rRep (pbmod C) pb)
        ->{a,b,c|Ob C}{f:H a c}{g:H b c}
        {x:Ob C}{p:H x a}{q:H x b}
        (Eq (o f p) (o g q))
        ->H x (s0 pb (pbcat0cons f g))
```

We can not just compare two elements (p, q) and (p', q') of $\texttt{pbmod}(\mathcal{C})(X, (f, g))$ with `Eq` because of their proof components. To ignore any difference in their proof parts, we define a constant `pbmod0Eq` that produce $\texttt{Eq}((p, q), (p', q'))$ from $\texttt{Eq}(p, p')$ and $\texttt{Eq}(q, q')$.

```
(* For the proof irrelevance *)
[C|Cat][a,b,c|Ob C][f|H a c][g|H b c];
[x|Ob C][pqr,pqr':pbmod0 C a b c f g x];
$[pbmod0Eq:(Eq (pbmod0p C pqr) (pbmod0p C pqr'))->
           (Eq (pbmod0q C pqr) (pbmod0q C pqr'))->Eq pqr pqr'];
Discharge C;
```

3.5.7 An example construction

As an example use of the commands defined in the previous sections, we construct equalizers from pullbacks and terminal objects. The construction goes as follows. Let $\mathcal{C}$ be a category with a terminal object T and pullbacks, and $A \underset{g}{\overset{f}{\rightrightarrows}} B$ a parallel pair of morphisms in $\mathcal{C}$. Construct the object C in $\mathcal{C}$ as the vertex of the pullback

$$\begin{tikzcd} C \arrow[r] \arrow[d] & B \arrow[d, "!_B"] \\ B \arrow[r, "!_B"'] & T \end{tikzcd}$$

where $!_B$ is the unique morhpism from B to T. The object C is a binary product of B with itself, but we adhere to its definition by the pullback for explanatory purpose. By the universality of T, $!_B f$ and $!_B g$ are the same, hence (f, g) is an element of $\mathtt{pbmod}(\mathcal{C})(A, (!_B, !_B))$. Let $h\colon A \longrightarrow C$ be the morphism determined by (f, g). Also let $\Delta\colon B \longrightarrow C$ be the morphism determined by the element $(\mathrm{id}_B, \mathrm{id}_B)$ of $\mathtt{pbmod}(\mathcal{C})(B, (!_B, !_B))$. Finally, let E be the vertex of the following pullback.

$$\begin{tikzcd} E \arrow[r, "e"] \arrow[d] & A \arrow[d, "h"] \\ B \arrow[r, "\Delta"'] & C \end{tikzcd}$$

Then, by the universality of the two pullbacks above, E is an equalizer of (f, g) and e the universal arrow.

In LEGO, this argument can be formalized as follows. First, we define some abbreviations.

```
[C:Cat];

[pb:ssFun0 (pbcat C) C];
[pbprf:rRep (pbmod C) pb];
CpbIntro == pbIntro pb pbprf;
CpbElim == pbElim pb pbprf;


[termfun:ssFun0 termcat C];
[termprf:rRep (termmod C) termfun];
Cterm == s0 termfun svoid:Ob C;
Csh == termIntro termfun termprf:{b:Ob C}H b Cterm;
Csh_unique == sh_unique termfun termprf;
```

Next, we construct the object C, and a function `tpair` that sends $(f, g\colon X \longrightarrow B)$ to $\langle f, g\rangle\colon X \longrightarrow C$; this is an instance of `rIntro`.

```
[a,b|Ob C];

(* (b x b) by pb's *)
[f,g:H a b];

Goal pbmod0 C b b Cterm (Csh b) (Csh b) a;
Refine sdep_pair; Refine f; Refine sdep_pair;
Refine g; Refine Csh_unique;

Save tpmod0cons;

Goal H a (s0 pb (pbcat0cons (Csh b) (Csh b)));
Refine CpbIntro (Csh b) (Csh b);
Refine f; Refine g;
Refine Csh_unique;

Save tpair;
Discharge f;
```

The following are some lemmas about `tpair` that are instances of the generic lemmas given in Section 3.5.2.

```
tpair_ext1 = ... : {a,b|Ob C}
        {f,g,k,l:H a b}(Eq (tpair f g) (tpair k l))->Eq f k
tpair_ext2 = ... : {a,b|Ob C}
        {f,g,k,l:H a b}(Eq (tpair f g) (tpair k l))->Eq g l
(* Naturality of tpair                          *)
(* (tpair f g) o h = (tpair  (f o h) (g o h)) *)
tpair_nat = ... : {a,b|Ob C}
        {f,g:H a b}{x:Ob C}{h:H x a}
        Eq (tpair (o f h) (o g h)) (o (tpair f g) h)
```

Now we give the definition of $\Delta\colon B \longrightarrow C$ and E, which are denoted `delta b` and `eql` in the code, respectively.

```
delta [b:Ob C] == tpair (id b) (id b);

(* The construction of equalizer *)
[a,b|Ob C][f,g:H a b];
eql == s0 pb (pbcat0cons (tpair f g) (delta b));
```

Finally, we give the correspondence between a morphism $X \xrightarrow{h} A$ with $fh = gh$ and a morphism $X \longrightarrow E$. First, in the forward direction, we use the properties of `tpair` to deduce from $fh = gh$ that (h, fh) is an element of $\texttt{pbmod}(\mathcal{C})(X, (\texttt{tpair}(f, g), \Delta))$, which in turn determines $X \longrightarrow E$.

```
[x:Ob C];
[h:H x a];
[prf:Eq (o f h) (o g h)];

Goal H x eql;
Refine CpbIntro; Refine h; Refine (o f h);
Qrepl Eq_sym (tpair_nat f g x h);
Expand delta; Qrepl Eq_sym (tpair_nat (id b) (id b) x (o f h));
Qrepl prf; Qrepl Eq_sym (ul (o g h));
Refine Eq_refl;

Save eqlIntro;
Discharge h;
```

In the reverse direction, the corresponding morphism is obtained as the first component of the element $(p, q) \in \texttt{pbmod}(\mathcal{C})(X, (\texttt{tpair}(fg), \Delta))$ determined by the given $X \longrightarrow E$.

```
[h:H x eql];
Goal H x a;
Refine pbmod0p; Refine +4 CpbElim; Refine +4 h;
Save eql_mor;
```

Not only the construction of the morphism eql_mor, one can also show that this indeed satisfies that $f \circ \texttt{eql\_mor} = g \circ \texttt{eql\_mor}$. First, the second component and the proof part of (p, q) given in the above paragraph is extracted.

```
Goal H x b;
Refine pbmod0q; Refine +4 CpbElim; Refine +4 h;
Save eql_wit;

Goal Eq (o (tpair f g) eql_mor) (o (delta b) eql_wit);
Refine pbmod0r; Refine +4 CpbElim; Refine +4 h;
Save eql_temp0;
```

With these and the properties of tpair, one can deduce the desired equality.

```
Goal Eq (o f eql_mor) (o (id b) eql_wit);
Refine tpair_ext1;
Refine (o g eql_mor); Refine (o (id b) eql_wit);
Qrepl tpair_nat f g x eql_mor;
Qrepl tpair_nat (id b) (id b) x eql_wit;
Refine eql_temp0;

Save eql_temp1;

Goal Eq (o g eql_mor) (o (id b) eql_wit);
Refine tpair_ext2;
Refine (o f eql_mor); Refine (o (id b) eql_wit);
Qrepl tpair_nat f g x eql_mor;
Qrepl tpair_nat (id b) (id b) x eql_wit;
Refine eql_temp0;
```

```
Save eql_temp2;

Goal Eq (o f eql_mor) (o g eql_mor);
Refine Eq_trans; Refine +1 eql_temp1; Refine Eq_sym (eql_temp2);

Save eql_prf;

Discharge h;
```

To complete a proof that the construction gives an equalizer, one further needs to prove that the above correspondence is bijective and that it is also natural. They can be shown similarly with suitable instances of generic lemmas given in Section 3.5.2.

Chapter 4

Fibrations with extra structure

4.1 Introduction

In this section, we study fibrations and structures on them. This is to illustrate how sophisticated categorical structures can be developed in terms of universality. This also provides a purely categorical description of the structure on fibrations used in Chapter 5 and 6 in a type theoretic context. Fibrations are of particular interest to computer science; their applications includes models for type theory, the study of logical predicates, and the semantics for concurrency (see, *e.g.*, [37, 34,67]).

In the spirit of the analysis in Chapter 2, our presentation pays attention to making clear what functor is required to be representable for a fibration to have certain structure. Since a fibration consists of its total category and a base category related by a functor, a structure on a fibration can be broken down into structures on these two categories and a relationship between them. We can therefore apply our definition of universal structure in Chapter 2. However, in this chapter, we do not restrict ourselves to technically applying our definition of universal structure *for ordinary categories*. This is because we favour the view that a fibration is a parameterised category [8,78]. So, we give a presentation where a fibration is dealt with as a whole. Particularly, we emphasise the role of $\mathcal{E}_u(-,-)$, that is, a collection of morphisms in a total category $\mathcal{E}$ over a specified base morphism u. We also give a detailed description of locally small fibrations, with which one can internalise arguments based on $\mathcal{E}_u(-,-)$. Our study is not enough to give a definition of universal structure on a fibration with the same technical detail as that for ordinary categories, but our presentation here is a start towards finding such a framework. We discuss a direction for further development in Chapter 8.

Fibrations are defined in Section 4.2. We give an alternative formulation based on $\mathcal{E}_u(-,-)$ showing that a cartesian morphism is the counit of a certain representation. Section 4.3 gives a few example of fibrations and Section 4.4 explains the relationship between fibrations and indexed categories. Section 4.5 defines morphisms and 2-cells of fibrations, giving the 2-category **Fib** of fibrations. Products and exponentials of fibrations are also defined. In Section 4.6, we describe locally small fibrations in detail. Fibred limits and fibred cartesian closed categories are defined in Section 4.7. For the particular cases of terminal objects and binary products, we reformulate the general definition in terms of $\mathcal{E}_u(-,-)$ and internalise it using local smallness. Finally, Section 4.8 describes products and coproducts indexed by base objects.

The novelty of this chapter is the new emphasis in the presentation and some corresponding reformulation of known material; the original work can be found, for example, in papers by Grothendiek [28], Giraud [24], Bénabou [7,6], and Celeyrette [15].

4.2 Fibrations and cartesian morphisms

Given a functor $p\colon \mathcal{E} \longrightarrow \mathcal{B}$, a diagram $D\colon \mathcal{I} \longrightarrow \mathcal{E}$ in $\mathcal{E}$ is said to be *over* the diagram $pD\colon \mathcal{I} \longrightarrow \mathcal{B}$. For $I \in \mathcal{B}$, the *fibre* $\mathcal{E}_I$ *over* I is the subcategory of $\mathcal{E}$ given by objects and morphisms over I and id_I, respectively; *i.e.*, those objects X of $\mathcal{E}$ for which $pX = I$, and those morphisms sent to id_I by p. A morphism in a fibre is called *vertical*. A fibre $\mathcal{E}_I$ is also written $p^{-1}(I)$.

Given $u\colon J \longrightarrow I$, $X \in \mathcal{E}_I$, and $Y \in \mathcal{E}_J$, $\mathcal{E}_u(Y,X)$ is the subset of $\mathcal{E}(Y,X)$ consisting of those morphisms over u, *i.e.*,

$$\mathcal{E}_u(Y,X) \equiv \{f\colon Y \longrightarrow X \mid pf = u\}.$$

Given $u\colon pY \longrightarrow pX$ in $\mathcal{B}$, we will write $f\colon Y \xrightarrow[(u)]{} X$ or $Y \xrightarrow[(u)]{f} X$ if $f \in \mathcal{E}_u(Y,X)$. Given $v\colon K \longrightarrow J$ in $\mathcal{B}$ and $Z \in \mathcal{E}_K$, the function

$$\mathcal{E}_u(Z,f)\colon \mathcal{E}_v(Z,Y) \longrightarrow \mathcal{E}_{uv}(Z,X)$$

is the restriction of

$$\mathcal{E}(Z,f)\colon \mathcal{E}(Z,Y) \longrightarrow \mathcal{E}(Z,X).$$

Dually, given $v\colon I \longrightarrow K$ in $\mathcal{B}$ and $Z \in \mathcal{E}_K$,

$$\mathcal{E}^u(f,Z)\colon \mathcal{E}_v(X,Z) \longrightarrow \mathcal{E}_{vu}(Y,Z)$$

is the restriction of

$$\mathcal{E}(f,Z)\colon \mathcal{E}(X,Z) \longrightarrow \mathcal{E}(Y,Z),$$

We now introduce our central definitions. These are not the usual version in the literature, but it is easily observable that they are equivalent.

Definition 4.2.1. Given $X \in \mathcal{E}_I$ and $u\colon J \longrightarrow I$ in $\mathcal{B}$, a morphism $f\colon Y \xrightarrow[(u)]{} X$ in $\mathcal{E}$ is called *cartesian over* u if, for all $v\colon K \longrightarrow J$ and $Z \in \mathcal{E}_K$,

$$\mathcal{E}_u(Z,f)\colon \mathcal{E}_v(Z,Y) \longrightarrow \mathcal{E}_{uv}(Z,X)$$

is a bijection.

Diagrammatically, for each g over uv, there exists unique g' over v such that the above of the following commutative diagrams lies over the lower.

$$\begin{tikzcd}
Z \arrow[rr, dashed, "g'"] \arrow[dr, "g"] & & Y \arrow[dl, "f"'] \\
& X \arrow[d, mapsto, "p"] & \\
& {} & \\
K \arrow[rr, "v"'] \arrow[dr, "uv"'] & & J \arrow[dl, "u"] \\
& I. &
\end{tikzcd}$$

We write $f\colon Y \relbar\joinrel\mapstochar\joinrel\rightarrow X$ if f is cartesian; so, a cartesian morphism f over u is written $f\colon Y \underset{(u)}{\relbar\joinrel\mapstochar\joinrel\rightarrow} X$. In this chapter, the notation $\mathcal{A} \relbar\joinrel\mapstochar\joinrel\rightarrow \mathcal{C}$ for modules is not used.

For conciseness, we often fold a diagram in $\mathcal{E}$ over another diagram in $\mathcal{B}$ into one diagram using the notation $X \xrightarrow[(u)]{f} Y$; *e.g.*, The above diagram may be written

$$\begin{tikzcd}
Z \arrow[rr, dashed, "g'", "(v)"'] \arrow[dr, "g", "(uv)"'] & & Y \arrow[dl, "f"', "(u)"] \\
& X &
\end{tikzcd}$$

Definition 4.2.2. A functor $p\colon \mathcal{E} \longrightarrow \mathcal{B}$ is a *fibration* if, for each pair (X, u) with $X \in \mathcal{E}_I$ and $u\colon J \longrightarrow I$, there exists a cartesian morphism over u with codomain X.

In many cases, p is the only fibration associated with $\mathcal{E}$ and $\mathcal{B}$ and is clear from the context; so we will often omit explicit mention of p, saying simply that $\mathcal{E}$ is fibred over $\mathcal{B}$.

The categories $\mathcal{E}$ and $\mathcal{B}$ are called the *total* category and the *base* category of the fibration, respectively.

We now show that cartesian morphisms are determined up to unique isomorphism in $\mathcal{E}_J$.

Lemma 4.2.3. If $f\colon Y \underset{(u)}{\relbar\joinrel\mapstochar\joinrel\rightarrow} X$ and $f'\colon Z \underset{(u)}{\relbar\joinrel\mapstochar\joinrel\rightarrow} X$ are cartesian over $u\colon J \longrightarrow I$, there is a unique isomorphism $c\colon Y \cong Z$ in $\mathcal{E}_J$ such that $f = f'c$.

Proof Let c and c' be the morphisms over id_J, which exist (uniquely) by Definition 4.2.1, such that the following commute.

$$\begin{tikzcd}
Y \arrow[rr, dashed, "c", "(\mathrm{id}_J)"'] \arrow[dr, "f", "(u)"', "|" marking] & & Z \arrow[dl, "f'"', "(u)", "|" marking] \\
& A &
\end{tikzcd}
\qquad
\begin{tikzcd}
Y \arrow[rr, dashed, "c'", "(\mathrm{id}_J)"'] \arrow[dr, "f'", "(u)"', "|" marking] & & Z \arrow[dl, "f"', "(u)", "|" marking] \\
& A &
\end{tikzcd}$$

Then, the composition cc' makes the outer triangle of the diagram below commute.

$$\begin{tikzcd} Y \arrow[r, "c", "(\mathrm{id}_J)"'] \arrow[dr, "f"'] & Z \arrow[r, "c'", "(\mathrm{id}_J)"'] \arrow[d, "f'"'] & Y \arrow[dl, "f"] \\ & A & \end{tikzcd}$$

But, $Y \xrightarrow{\mathrm{id}_X} Y$ over id_J also makes this commute. Since f is cartesian, there is only one such morphism over id_J and cc' must be equal to $\mathrm{id}_{A'}$. Similarly, $c'c = \mathrm{id}_{A''}$. ▮

Henceforth, we may omit the name of a cartesian morphism, only specifying the underlying base morphism. For example, we will write $Y \underset{(u)}{\longrightarrow\!\!\!\!\!\!+\;} X$ for $Y \overset{f}{\underset{(u)}{\longrightarrow\!\!\!\!\!\!+\;}} X$.

Equivalently, a cartesian morphism can be seen as the counit of a representation which describes the domain Y of $Y \underset{(u)}{\longrightarrow\!\!\!\!\!\!+\;} X$ in terms of morphisms into it over specified base morphisms, as we now see:

Proposition 4.2.4. Given $(X, u\colon J \longrightarrow I)$ with $X \in \mathcal{E}_I$, an object $Y \in \mathcal{E}_J$ is the domain of a cartesian morphism $Y \underset{(u)}{\longrightarrow\!\!\!\!\!\!+\;} X$ over u if and only if, for each $Z \in \mathcal{E}_K$ and $v\colon K \longrightarrow J$, there is a bijection $\phi_{(v,Z)}\colon \mathcal{E}_v(Z, Y) \cong \mathcal{E}_{uv}(Z, X)$ satisfying the following naturality condition: given $h\colon W \longrightarrow Z$ in $\mathcal{E}$ over $w\colon L \longrightarrow K$ in $\mathcal{B}$, the following commutes.

$$\begin{tikzcd} \mathcal{E}_v(W, Y) \arrow[r, "\phi_{(v,W)}"] \arrow[d, "\mathcal{E}^w(h, Y)"'] & \mathcal{E}_{uv}(W, X) \arrow[d, "\mathcal{E}^w(h, X)"] \\ \mathcal{E}_{vw}(Z, Y) \arrow[r, "\phi_{(vw,Z)}"] & \mathcal{E}_{uvw}(Z, X) \end{tikzcd}$$

Proof The naturality condition ensures that $f \equiv \phi_{(\mathrm{id}_J, Y)}(\mathrm{id}_Y)$ is cartesian, *i.e.*, $\phi_{(v,Z)} = \mathcal{E}_u(Z, f)$. ▮

We indicate this naturality in Z by the notation

$$\frac{Z \underset{(v)}{\longrightarrow} X'}{Z \underset{(uv)}{\longrightarrow} X.} \cong$$

The natural bijection determines Y up to unique isomorphism. This is by the same argument as that a representing object $C \in \mathcal{C}$ of a functor $\mathcal{C} \longrightarrow \mathbf{Set}$ is determined up to isomorphism. However, the difference between specifying morphisms into Y and specifying morphisms into Y *over a specified base morphism* is important. Particularly, Y is determined up to isomorphism *in* $\mathcal{E}_J$, not just in $\mathcal{E}$.

The definition has immediate consequences in composition and cancellation of cartesian morphisms.

Lemma 4.2.5. (i) Any cartesian morphism over an identity is an isomorphism.

(ii) If f and g are cartesian, so is fg.

(iii) If fg and f is cartesian, so is g.

Proof (i) is obvious. Let $f\colon Y \longrightarrow X$, $g\colon Z \longrightarrow Y$, $u = pf$, and $v = pg$. For (ii), given any $W \in \mathcal{E}_L$ and $w\colon L \longrightarrow pZ$, one has

$$\begin{array}{ll} \dfrac{W \underset{(w)}{\longrightarrow} Z}{W \underset{(vw)}{\longrightarrow} Y} \cong & ;\ g \circ - \\ \overline{W \underset{(uvw)}{\longrightarrow} X,} \cong & ;\ f \circ - \end{array}$$

and one can check that this bijection is natural in W and w, and that $Z \xrightarrow{\mathrm{id}_Z} Z$ corresponds to fg. Similarly for (iii), given any $W \in \mathcal{E}_L$ and $w\colon L \longrightarrow pZ$, one has

$$\begin{array}{ll} \dfrac{W \underset{(w)}{\longrightarrow} X}{W \underset{(uvw)}{\longrightarrow} Z} \cong & ;\ fg \circ - \\ \overline{W \underset{(vw)}{\longrightarrow} Y.} \cong & ;\ \text{the inverse of } f \circ - \end{array}$$

∎

Lemma 4.2.6. Any morphism $Y \underset{(u)}{\longrightarrow} X$ in the total category of a fibration factorises as a cartesian morphism $Z \underset{(u)}{\longrightarrow\!\!\!\!\!+} X$ over u followed by a vertical morphism $\bar{u}\colon Y \underset{(\mathrm{id}_{pY})}{\longrightarrow} Z$. The factorisation is unique up to isomorphism in $\mathcal{E}_{pY}$.

Proof Immediate from the definition. ∎

It is sometimes convenient to chose particular cartesian morphisms for all pairs $(X, u\colon J \longrightarrow I)$ with $X \in \mathcal{E}_I$. In the light of Proposition 4.2.4, this is similar to choosing particular product cones for all pairs of objects in a category with binary products.

Definition 4.2.7. A *cleavage* for a fibration p is a choice, for each $(X, u\colon J \longrightarrow I)$ with $X \in \mathcal{E}_I$, of a cartesian morphism over u with codomain X. A *cloven* fibration is a fibration together with a cleavage. The specified cartesian morphism for (X, u) is written $\widehat{u}(X)\colon X[u] \mathrel{\longrightarrow\mkern-18mu+\mkern6mu} X$. A *splitting* is a cleavage which satisfies the conditions: $\widehat{\mathrm{id}_I}(X) = \mathrm{id}_X$, and for $X \in \mathcal{E}_I$, $u\colon J \longrightarrow I$, and $v\colon K \longrightarrow J$, $\widehat{v}(X[u])\widehat{u}(X) = \widehat{uv}(X)$. A *split* fibration is a fibration with a splitting.

In a diagram, we may omit $\widehat{u}(X)$ and (u) from $X[u] \underset{(u)}{\overset{\widehat{u}(X)}{\mathrel{\longrightarrow\mkern-18mu+\mkern6mu}}} X$ and write $X[u] \mathrel{\longrightarrow\mkern-18mu+\mkern6mu} X$.

Using the axiom of choice, one can assume, without a loss of generality, that a fibration is always equipped with a cleavage. Also, we will later see that any cloven fibration is equivalent, in a 2-category of fibrations defined below, to a split fibration. The axiom of choice is an axiom of classical set theory and it does not, in general, hold in an topos or a type theory in which one may work internally. However, the existence of a cleavage is often a reasonable assumption as it is *not* the assumption of the axiom of choice in the base category (which may be a topos and whose internal logic one may use), but in the metatheory.

The universality of $X[u]$ with $u\colon J \longrightarrow I$ and $X \in \mathcal{E}_I$ extends the assignment $X \mapsto X[u]$ to a functor $\mathcal{E}_I \longrightarrow \mathcal{E}_J$.

Definition 4.2.8. Given a cleavage and $u\colon J \longrightarrow I$ in $\mathcal{B}$, the *substitution functor* $(-)[u]\colon \mathcal{E}_I \longrightarrow \mathcal{E}_J$ sends $Y \xrightarrow{f} X$ in $\mathcal{E}_I$ to the unique morphism $Y[u] \xrightarrow{f[u]} X[u]$ in $\mathcal{E}_J$ that satisfies

$$\begin{tikzcd} Y[u] \arrow[r, "{f[u]}", "(\mathrm{id}_J)"'] \arrow[d, "+" description] & X[u] \arrow[d, "+" description] \\ Y \arrow[r, "f", "(\mathrm{id}_I)"'] & X. \end{tikzcd}$$

The functoriality of $(-)[u]$ is immediate from the uniqueness of $f[u]$. Since the substitution functor for every cleavage is isomorphic to each other, we loosely say, for example, the "substitution functor preserves a limit" omitting "for any given cleavage".

The assignment $I \mapsto \mathcal{E}_I$, together with $u \mapsto (-)[u]$, satisfies the coherence condition to form an *indexed category* [59]. As we will see later, fibration and indexed category are equivalent concepts.

When a fibration p is regarded as a generalised category [8], the following definition gives the "opposite category " of p [6].

Definition 4.2.9. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$, the *opposite fibration* of p, written $p^{(\mathrm{op})}: \mathcal{E}^{(\mathrm{op})} \longrightarrow \mathcal{B}$, is defined as follows. The category $\mathcal{E}^{(\mathrm{op})}$ has the same objects as $\mathcal{E}$. Given $X, Y \in \mathcal{E}$, consider the set of pairs of morphisms $(X \xleftarrow[(\mathrm{id})]{f} Y' \xrightarrow{c} Y)$ with f vertical and c cartesian. Define the equivalence relation on this set by

$$(X \xleftarrow[(\mathrm{id})]{f} Y' \xrightarrow{c} Y) \sim (X \xleftarrow[(\mathrm{id})]{g} Y'' \xrightarrow{d} Y)$$

$$\Leftrightarrow$$

$$\exists\; i: Y' \xrightarrow[(\mathrm{id})]{\cong} Y''.$$

$$\begin{tikzcd}
& Y' \arrow[dl, "f"', "(\mathrm{id})"] \arrow[dr, "c", "\shortmid" marking] \arrow[dd, "i \cong"', "(\mathrm{id})"] & \\
X & & Y \\
& Y'' \arrow[ul, "g", "(\mathrm{id})"'] \arrow[ur, "d"', "\shortmid" marking] &
\end{tikzcd}$$

commutes

A morphism of $\mathcal{E}^{(\mathrm{op})}$ from X to Y is such an equivalence class of pairs of morphisms. Given two $\mathcal{E}^{(\mathrm{op})}$ morphisms $[(f, c)]: Y \longrightarrow Z$ and $[(g, d)]: X \longrightarrow Y$ where (f, c) and (g, d) are representatives for respective equivalence classes, the composition

$[(f,c)] \circ [(g,d)] \colon X \longrightarrow Z$ is the equivalence class given by (gf', cc'):

$$\begin{tikzcd}
X & & \\
Y' \arrow[u, "g"] \arrow[r, "d", "\shortmid" marking] & Y & \\
Z'' \arrow[u, dashed, "f'"] \arrow[r, "c'", "(pd)"', "\shortmid" marking] & Z' \arrow[u, "f"'] \arrow[r, "c", "\shortmid" marking] & Z,
\end{tikzcd}$$

where c' is any cartesian morphism over pd and f' is the unique vertical morphism that makes the square commutes. It is easy to check that the choice of (f,c), (g,d), and c' does not affect the equivalence class $[(gf', cc')]$. The fibration $p^{(\mathrm{op})}$ sends X to pX and $[(f,c)]$ to pc.

Fibrewise, one has $(\mathcal{E}^{(\mathrm{op})})_I \cong (\mathcal{E}_I)^{\mathrm{op}}$.

4.3 Examples of fibrations

Family fibration

An intuition behind the definition of fibrations is to abstract the situation where a category, or its object, is parameterized by objects of another category. The most basic example in this regard is given as follows.

Definition 4.3.1. Let $\mathcal{C}$ be a category.

- The category $\mathbf{Fam}(\mathcal{C})$ of families of objects of $\mathcal{C}$ is given by

  Objects Families $(X_i)_{i \in I}$ ($X_i \in \mathcal{C}$, $I \in \mathbf{Set}$) of objects of $\mathcal{C}$.

  Morphisms $\mathbf{Fam}(\mathcal{C})((Y_j)_{j \in J}, (X_i)_{i \in I}) = \{(u,f) \mid u \colon J \longrightarrow I, f = (f_j \colon Y_j \longrightarrow X_{uj})_{j \in J}\}$.

  Composition

  $$\frac{(v,g) \colon (Z_k)_{k \in K} \longrightarrow (Y_j)_{j \in J} \quad (u,f) \colon (Y_j)_{j \in J} \longrightarrow (X_i)_{i \in I}}{(u,f) \circ (v,g) \equiv (uv, (f_{vk} g_k)_{k \in K}) \colon (Z_k)_{k \in K} \longrightarrow (X_i)_{i \in I}}$$

- The split fibration $p_{\mathcal{C}}: \mathbf{Fam}(\mathcal{C}) \longrightarrow \mathbf{Set}$ sends $(u, f): (Y_j)_{j \in J} \longrightarrow (X_i)_{i \in I}$ to $u: J \longrightarrow I$. For $((X_i)_{i \in I}, u: J \longrightarrow I)$, the canonical splitting is given by

$$\hat{u}(X) = (u, (\mathrm{id}_{X_{uj}})_{j \in J}): (X_{uj})_j \longrightarrow (X_i)_{i \in I}.$$

So, a cartesian morphism of $p_{\mathcal{C}}$ over u gives re-indexing of families along u. The inverse of $\mathcal{E}_u(Z, \hat{u}(X))$ required in Definition 4.2.1 is trivially given by

$$\frac{Z \xrightarrow[(uv)]{(uv,(f_{uv(k)})_{k \in pZ})} X}{Z \xrightarrow[(v)]{(v,(f_{uv(k)})_{k \in pZ})} X[u].} \cong$$

The fibre $\mathbf{Fam}(\mathcal{C})_I$ over $I \in \mathbf{Set}$ is isomorphic to $\mathcal{C}^I$, *i.e.*, $\mathcal{C}$ parameterised by the set I.

Codomain fibrations

There is a trivial bijection between I-indexed families of disjoint sets and maps from some set J into I. Generalizing this, one may regard a morphism $J \longrightarrow I$ in a category as a family of object indexed by I.

Definition 4.3.2. Given a category $\mathcal{B}$ with pullbacks, the codomain fibration $\mathrm{cod}: \mathcal{B}^{\mathbf{2}} \longrightarrow \mathcal{B}$ sends an object $f: X \longrightarrow I$ to I and a morphism (u, h) from $f: X \longrightarrow I$ to $g: Y \longrightarrow J$ to u. The cartesian morphisms are pullback squares.

The composition with the pullback square

$$\begin{tikzcd}
Y \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d] \\
J \arrow[r, "u"'] & I
\end{tikzcd}$$

has the inverse mapping $(uv, f) \mapsto (v, \bar{f})$, as required in Definition 4.2.1

$$\frac{\begin{tikzcd} Z \arrow[rr, "f"] \arrow[d, "x"'] & & X \arrow[d] \\ K \arrow[r, "v"'] & J \arrow[r, "u"'] & I \end{tikzcd}}{\begin{tikzcd} Z \arrow[r, dashed, "\bar{f}"] \arrow[d, "x"'] & Y \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & X \arrow[d] \\ K \arrow[r, "v"'] & J \arrow[r, "u"'] & I \end{tikzcd}} \cong$$

where $\bar{f}\colon Z \dashrightarrow Y$ is the unique morphism determined by f and vx. The fibre $\mathcal{B}_I$ over $I \in \mathcal{B}$ is the slice category $\mathcal{B}/I$.

Domain fibrations

The following fibration plays an important role in conjunction with the *change of base* operation defined later.

Definition 4.3.3. Given a category $\mathcal{B}$ and an object I in $\mathcal{B}$, the split fibration $\mathrm{Y}_I\colon \mathcal{B}/I \longrightarrow \mathcal{B}$ sends an object $f\colon J \longrightarrow I$ to J and a morphisms $h\colon J \longrightarrow K$ from $f\colon J \longrightarrow I$ to $g\colon K \longrightarrow I$ to h. the canonical splitting for $(f\colon J \longrightarrow I, u\colon K \longrightarrow J)$ is given by $f[u] = fu$ and $\widehat{u}(f) = u\colon (fu) \longrightarrow f \in \mathcal{B}/I$.

4.4 Indexed categories

Indexed categories [59] provide an alternative formulation of fibrations. We give the definition and the *Grothendieck construction* of a fibration from an indexed category. In some applications, indexed categories are more intuitive than fibrations; for a computer science application, see *e.g.* [85]. But technically they are less convenient than fibrations. A detailed comparison of the two concepts can be found in [8].

Definition 4.4.1. Given a category $\mathcal{B}$, an indexed category $\mathcal{S}$ with the base $\mathcal{B}$ is given by the data

- for each $I \in \mathcal{B}$, a category $\mathcal{S}_I$.
- for each $u: J \longrightarrow I$ in $\mathcal{B}$, a functor $u^*: \mathcal{S}_I \longrightarrow \mathcal{S}_J$,
- for each $I \in \mathcal{B}$, a natural isomorphism $\gamma_I: \mathrm{id}_{\mathcal{S}_I} \cong \mathrm{id}_I^*$
- for each composable pair $u: J \longrightarrow I$ and $v: K \longrightarrow J$, a natural isomorphism $\delta_{u,v}: v^* u^* \cong (uv)^*]$.

satisfying the conditions

- for any $u: J \longrightarrow I$, $\delta_{u,\mathrm{id}_J} \circ \gamma_J u^* = \mathrm{id}_{u^*}$ and $\delta_{\mathrm{id}_I,u} \circ u^* \gamma_I = \mathrm{id}_u^*$.
- for any $u: J \longrightarrow I$, $v: K \longrightarrow J$, and $w: L \longrightarrow K$, $\delta_{uv,w} \circ w^* \delta_{u,v} = \delta_{u,vw} \circ u^* \delta_{v,w}$.

Note that a functor $\mathcal{S}: \mathcal{B}^{\mathrm{op}} \longrightarrow \mathbf{Cat}$ is an indexed category with all γ_I and $\delta_{u,v}$ being identity. In fact, indexed categories are exactly pseudo functors [27] from $\mathcal{B}^{\mathrm{op}}$ to $\mathbf{Cat}$.

Any cloven fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ gives rise to an indexed category with the data $I \mapsto \mathcal{E}_I$, $u \mapsto (-)[u]$, and natural isomorphisms given by the universality of each $X[u]$ with $X \in \mathcal{E}_I$ and $u: J \longrightarrow I$. We write this indexed fibration as $\mathcal{F}(p)$.

Conversely, given an indexed category $\mathcal{S}$, one can construct a cloven fibration.

Definition 4.4.2. (Grothendieck construction)

Given an indexed category $\mathcal{S}$, the *Grothendieck* category $\mathcal{G}(\mathcal{S})$ of $\mathcal{S}$ is defined by

- an object of $\mathcal{G}(\mathcal{S})$ is a pair $(I \in \mathcal{B}, X \in \mathcal{S}_I)$.
- a morphism of $\mathcal{G}(\mathcal{S})$ from (J, Y) to (I, X) is a pair (u, f) with $u: J \longrightarrow I$ in $\mathcal{B}$ and $f: Y \longrightarrow u^* X$ in $\mathcal{S}_J$.
- the composition of $(u, f): (J, Y) \longrightarrow (I, X)$ and $(v, g): (K, Z) \longrightarrow (J, Y)$ is given by $(uv, \delta_{u,v} \circ v^* f \circ g)$.
- the identity on (I, X) is $(\mathrm{id}_I, \gamma_{I,X})$.

It is straightforward to check the following.

Proposition 4.4.3. The functor $p_{\mathcal{S}}: \mathcal{G}(\mathcal{S}) \longrightarrow \mathcal{B}$ given by the first projections for both objects and morphisms is a cloven fibration. The cleavage for $((I, X), u)$ with $u: J \longrightarrow I$ and $(I, X) \in \mathcal{G}(\mathcal{S})_I$ is given by the morphism (u, id_{u^*X}). ▮

Particularly, if all γ_I, $\delta_{u,v}$ are identity, *i.e.*, if $\mathcal{S}$ is a mere functor from $\mathcal{B}^{\mathrm{op}}$ to **Cat**, then the fibration $p_{\mathcal{S}}$ has a splitting.

Further, it is also straightforward to check that $\mathcal{F}$ and $\mathcal{G}$ are mutually inverse constructions, *i.e.*, for any indexed category $\mathcal{S}$, $\mathcal{F}(\mathcal{G}(\mathcal{S}))$ is isomorphic to $\mathcal{S}$, and for any cloven fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ $p_{\mathcal{F}(p)}: \mathcal{G}(\mathcal{F}(p)) \longrightarrow \mathcal{B}$ is isormophic to p. But to make this precise, one needs to define the 2-category of fibrations.

4.5 2-category Fib

We define the 2-category **Fib** of fibrations and the sub 2-category **Fib**$(\mathcal{B})$ of fibrations with the base $\mathcal{B}$. Then we define products and exponentials of fibrations in **Fib**$(\mathcal{B})$. The latter is applied to show that every cloven fibration is equivalent to a split fibration.

Definition 4.5.1. • Given two fibrations $p: \mathcal{E} \longrightarrow \mathcal{B}$ and $q: \mathcal{F} \longrightarrow \mathcal{C}$, a *fibred 1-cell* from p to q is a pair $(F: \mathcal{B} \longrightarrow \mathcal{C}, G: \mathcal{E} \longrightarrow \mathcal{F})$ of functors such that the diagram

$$\begin{tikzcd} \mathcal{E} \arrow[r, "G"] \arrow[d, "p"'] & \mathcal{F} \arrow[d, "q"] \\ \mathcal{B} \arrow[r, "F"] & \mathcal{C} \end{tikzcd}$$

commutes and moreover, for every cartesian morphism f in $\mathcal{E}$, Gf is cartesian in $\mathcal{F}$ over Fpf. If p and q are cloven (split), (F, G) is called *cleavage preserving* (*splitting preserving*, respectively) if G sends the chosen cartesian morphism for $(X \in \mathcal{E}_I, u: J \longrightarrow I)$ in $\mathcal{E}$ to that for (GX, Fu) in $\mathcal{F}$.

- Given two fibred 1-cells (F, G) and (K, L) from a fibration p to q, a *fibred 2-cell* from (F, G) to (K, L) is a pair $(s\colon F \longrightarrow K, t\colon G \longrightarrow L)$ of natural transformations such that, for every $X \in \mathcal{E}$, $qt_X = s_X p$.

$$\begin{tikzcd}
\mathcal{E} \arrow[r, bend left, "G"] \arrow[r, bend right, "L"'] \arrow[r, phantom, "\Downarrow t"] \arrow[d, "p"'] & \mathcal{F} \arrow[d, "q"] \\
\mathcal{B} \arrow[r, bend left, "F"] \arrow[r, bend right, "K"'] \arrow[r, phantom, "\Downarrow s"] & \mathcal{C}
\end{tikzcd}$$

- The 2-category **Fib** has as objects fibrations, as 1-cells fibred 1-cells, and as 2-cells fibred 2-cells. The composition is given componentwise using that of **Cat**. The subcategory $\mathbf{Fib}_s$ of **Fib** has as objects split fibrations, as morphisms splitting preserving fibred 1-cells, and as 2-cells fibred 2-cells. For each $\mathcal{B}$ in **Cat**, the subcategory $\mathbf{Fib}(\mathcal{B})$ of **Fib** has as objects fibrations with the base $\mathcal{B}$, as morphisms fibred 1-cells with their first component being $\mathrm{id}_{\mathcal{B}}$, as 2-cells fibred 2-cells with their first component $\mathrm{id}_{\mathrm{id}_{\mathcal{B}}}$. A fibred 1-cells in $\mathbf{Fib}(\mathcal{B})$ is also called a *cartesian functor*.

Pulling back a fibration along a functor, one obtains another fibration.

Proposition 4.5.2. Given a fibration $p\colon \mathcal{E} \longrightarrow \mathcal{B}$ and a functor $F\colon \mathcal{C} \longrightarrow \mathcal{B}$, let the *change of base* diagram be the following pullback in **Cat**.

$$\begin{tikzcd}
F^*(\mathcal{E}) \arrow[r] \arrow[d, "F^*p"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathcal{E} \arrow[d, "p"] \\
\mathcal{C} \arrow[r, "F"] & \mathcal{B}
\end{tikzcd}$$

Then, F^*p is a fibration.

Proof Given $(C, X) \in \mathcal{F}^*(\mathcal{E})_C$, *i.e.*, (C, X) with $FC = pX$, let $u\colon D \longrightarrow C$ be a morphism in $\mathcal{C}$ and $c\colon Y \underset{(Fu)}{\longrightarrow\!\!\!\!\!\!\!+\,\,\,} X$ a cartesian morphism with respect to p. Then, a F^*p-cartesian morphism over u with codomain (C, X) is given by $(u, c)\colon (D, Y) \longrightarrow (C, X)$. ∎

This extends to the *change of base* 2-functor $F^*(-)\colon \mathbf{Fib}(\mathcal{B}) \longrightarrow \mathbf{Fib}(\mathcal{C})$ [37, 34].

When F is also a fibration, one has the following.

Proposition 4.5.3. Given two fibrations $p\colon \mathcal{E} \longrightarrow \mathcal{B}$ and $q\colon \mathcal{F} \longrightarrow \mathcal{B}$, a product $p \times q$ of p and q in $\mathbf{Fib}(\mathcal{B})$ is given by the pullback diagram

$$\begin{tikzcd} & \mathcal{E} \times_{\mathcal{B}} \mathcal{F} \arrow[dl, "p^*q"'] \arrow[dr, "q^*p"] \arrow[dd, "p \times q" description] & \\ \mathcal{E} \arrow[dr, "p"'] & & \mathcal{F} \arrow[dl, "q"] \\ & \mathcal{B}. & \end{tikzcd}$$

Proof The functor $p \times q$ is a fibration since p^*q is a fibration and the composition of two fibrations is again a fibration. It is also obviously a product of p and q in $\mathbf{Cat}/\mathcal{B}$, *i.e.*, $(\mathbf{Cat}/\mathcal{B})(r, p \times q) \cong (\mathbf{Cat}/\mathcal{B})(r, p) \times (\mathbf{Cat}/\mathcal{B})(r, q)$. One can check that this restricts to $\mathbf{Fib}(\mathcal{B})(r, p \times q) \cong \mathbf{Fib}(\mathcal{B})(r, p) \times \mathbf{Fib}(\mathcal{B})(r, q)$ when r is a fibration. ▮

The product is fibrewise in the sense that, for $I \in \mathcal{B}$, $(\mathcal{E} \times_{\mathcal{B}} \mathcal{F})_I \cong \mathcal{E}_I \times \mathcal{F}_I$.

The 2-category $\mathbf{Fib}(\mathcal{B})$ also has the terminal object $\mathrm{id}_{\mathcal{B}}$.

The exponentials are defined with the aid of a fibred version of Yoneda lemma.

Lemma 4.5.4. (Yoneda lemma)

Given a cloven fibration $p\colon \mathcal{E} \longrightarrow \mathcal{B}$ and $I \in \mathcal{B}$, $\mathbf{Fib}(\mathcal{B})(\mathrm{Y}_I, p) \simeq \mathcal{E}_I$.

Proof The equivalence is given, in the forward direction, $F \in \mathbf{Fib}(\mathcal{B})(\mathrm{Y}_I, p) \mapsto F_I(\mathrm{id}_I) \in \mathcal{E}_I$ and $t\colon F \longrightarrow G \mapsto t_I(\mathrm{id}_I)$. In the reverse direction, let $X \in \mathcal{E}_I$. Let $F_X\colon \mathrm{Y}_I \longrightarrow p$ to be a functor that sends $f \in \mathcal{B}/I$ to $X[f]$, and $h\colon f \longrightarrow g$ in $\mathcal{B}/I$

to the unique morphism $F_X h$ over h that makes the diagram

$$\begin{tikzcd}
X[f] \arrow[rr, dashed, "F_X h", "(h)"'] \arrow[dr, "|"'] & & X[g] \arrow[dl, "|"] \\
& X &
\end{tikzcd}$$

commutes. Any morphism h of $\mathcal{B}/I$ is cartesian, but $F(h)$ is cartesian by Lemma 4.2.5, so F_X is a cartesian functor. One can similarly define, for each $k: X \longrightarrow Y$, a natural transformation from F_X to F_Y. The verification of the equivalence is straightforward. ▮

Given $u: I \longrightarrow J$, define a cartesian functor $\mathrm{Y}_u: \mathrm{Y}_I \longrightarrow \mathrm{Y}_J$ in $\mathbf{Fib}(\mathcal{B})$ by $u \circ (-)$.

Definition 4.5.5. Given two fibrations $p: \mathcal{E} \longrightarrow \mathcal{B}$ and $q: \mathcal{F} \longrightarrow \mathcal{B}$, the *exponential fibration* q^p in $\mathbf{Fib}(\mathcal{B})$ is given by the Grothendieck construction on the functor $\mathbf{Fib}(\mathcal{B})(\mathrm{Y}_- \times p, q)$ from $\mathcal{B}^{\mathrm{op}}$ to $\mathbf{Cat}$ that sends I to the category $\mathbf{Fib}(\mathcal{B})(\mathrm{Y}_I \times p, q)$ and $u: I \longrightarrow J$ to the functor

$$\mathbf{Fib}(\mathcal{B})(\mathrm{Y}_u \times p, q): \mathbf{Fib}(\mathcal{B})(\mathrm{Y}_J \times p, q) \longrightarrow \mathbf{Fib}(\mathcal{B})(\mathrm{Y}_I \times p, q).$$

▮

In [15], q^p is shown to have the universal property

$$\mathbf{Fib}(\mathcal{B})(r, q^p) \simeq \mathbf{Fib}(\mathcal{B})(r \times p, q).$$

The proof is somewhat involved, but in the special case where $r = \mathrm{Y}_I$, one can easily see that

$$\mathbf{Fib}(\mathcal{B})(\mathcal{Y}_I, q^p) \cong (q^p)^{-1}(I) = \mathbf{Fib}(\mathcal{B})(\mathcal{Y}_I \times p, q).$$

Note that $\mathbf{Fib}(\mathcal{B})(\mathrm{Y}_- \times p, q)$ is a functor, not just a pseudo functor, so q^p has a splitting. By taking p to be the terminal object of $\mathbf{Fib}(\mathcal{B})$, *i.e.*, $\mathrm{id}_{\mathcal{B}}$, one has

Proposition 4.5.6. Every cloven fibration p is equivalent to a split fibration $p^{\mathrm{id}_{\mathcal{B}}}$.

4.6 Local smallness

In this section, we study *local smallness* [15,6] of fibrations. With a locally small fibration p, one can form an object in the base category $\mathcal{B}$ that plays a role of a Hom-set for an ordinary category. This allows one to internalise various concepts on p in $\mathcal{B}$, generalising the relationship between an ordinary category and the category **Set** of sets [8]. More interestingly to computer scientists, local smallness provides the key connection between type theory and fibrations. We will see this in detail in Chapter 5 and 6.

Definition 4.6.1. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$, the *groupoid* fibration $p_{\mathbf{gpd}}: \mathcal{E}_{\mathbf{gpd}} \longrightarrow \mathcal{B}$ is a restriction of p to the subcategory $\mathcal{E}_{\mathbf{gpd}}$ given by all cartesian morphisms of $\mathcal{E}$.

Each fibre $(\mathcal{E}_{\mathbf{gpd}})_I$ is a groupoid, *i.e.*, every morphism in a fibre $(\mathcal{E}_{\mathbf{gpd}})_I$ is an isomorphism.

Definition 4.6.2. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ and a pair of objects $A \in \mathcal{E}_I$ and $B \in \mathcal{E}_J$, define the functor $H_{AB}: \mathcal{E}_{\mathbf{gpd}}^{\mathrm{op}} \longrightarrow \mathbf{Set}$ by

$$H_{AB}: \quad \mathcal{E}_{\mathbf{gpd}}^{\mathrm{op}} \longrightarrow \mathbf{Set}$$

$$\begin{tikzcd}
X \arrow[r, phantom, ""] & \{\, (c,g) \mid A & X \arrow[l, "c"', "\shortmid" marking] \arrow[r, "g"] & B\} \arrow[d, phantom, ""] \\
Y \arrow[u, "f", "\shortmid" marking] \arrow[r, phantom, "\mapsto" near start] & & & \\
& \{\, (c',g') \mid A & Y \arrow[l, "c'"', "\shortmid" marking] \arrow[r, "g'"] & B\}.
\end{tikzcd}
\qquad
\{(c,g)\} \xrightarrow{(c,g) \mapsto (cf,gf)} \{(c',g')\}$$

We say p is *locally small* at (A,B) if H_{AB} is representable; p is locally small if p is so at every (A,B).

The chosen counit of a representation for H_{AB} is denoted by

$$\begin{tikzcd}
A & A' \arrow[l, "|" marking] \arrow[r, "\chi_{AB}"] \arrow[d, mapsto, "p"] & B \\
I & \mathrm{Hom}(A,B) \arrow[l, "d_{0,AB}"'] \arrow[r, "d_{1,AB}"'] & J
\end{tikzcd}$$

usually suppressing the suffix AB. The morphisms d_0 and d_1 are called Hom-*projections*. The definition amounts to require, for each (u,f) below, the unique existence of a cartesian morphism $\bar{f}$ that makes the following commute.

$$\begin{tikzcd}
& A & \\
A'' \arrow[ur, "|" marking, "(u)"'] \arrow[rr, dashed, "|" marking, "\bar{f}"] \arrow[dr, "f"] & & A' \arrow[ul, "|" marking, "(d_0)"] \arrow[dl, "\chi"', "(d_1)"] \\
& B &
\end{tikzcd}$$

Proposition 4.6.3. Given a locally small fibration $p\colon \mathcal{E} \longrightarrow \mathcal{B}$ with finitely complete $\mathcal{B}$ and $A \in \mathcal{E}$, one can extend the assignment $(A,B) \mapsto \mathrm{Hom}(A,B)$ to a fibred 1-cell

$$\mathrm{Hom}(A,-)\colon \quad p \longrightarrow \mathrm{cod}$$

$$\begin{tikzcd}
\mathcal{E} \arrow[r, "\mathrm{Hom}(A,-)"] \arrow[d, "p"'] & \mathcal{B}^{\mathbf{2}} \arrow[d, "\mathrm{cod}"] \\
\mathcal{B} \arrow[r, "I \times (-)"'] & \mathcal{B}
\end{tikzcd}$$

Proof The functor $\mathrm{Hom}(A,-)$ sends $B \in \mathcal{E}_J$ to $\mathrm{Hom}(A,B) \xrightarrow{\langle d_{0,AB}, d_{1,AB}\rangle} I \times J$, and $B \xrightarrow{f} C$ over $J \xrightarrow{u} K$ to

$$\begin{tikzcd}
\mathrm{Hom}(A,B) \arrow[r, "\mathrm{Hom}(A,f)"] \arrow[d, "{\langle d_{0,AB}, d_{1,AB}\rangle}"'] & \mathrm{Hom}(A,C) \arrow[d, "{\langle d_{0,AC}, d_{1,AC}\rangle}"'] \\
I \times J \arrow[r, "I \times u"'] & I \times K
\end{tikzcd}$$

where $\mathrm{Hom}(A, f)$ is the morphism which underlies the unique cartesian morphism f' below.

$$\begin{tikzcd}
A' \arrow[rr, dashed, "f'" , "+" marking] \arrow[dr, "(d_{0,AB})"'] \arrow[dd, "\chi_{AB}"'] & & A'' \arrow[dl, "(d_{0,AC})"] \arrow[dd, "\chi_{AC}"] \\
& A & \\
B \arrow[rr, "f"'] & & C
\end{tikzcd}$$

It remains to show that $\mathrm{Hom}(A, -)$ sends a cartesian morphism f in $\mathcal{E}$ to a cartesian morphism in $\mathcal{B}^{\mathbf{2}}$, *i.e.*, a pullback square. Suppose $f \colon B \longrightarrow C$ is cartesian. Then, we have, for any $\langle v, w\rangle \colon L \longrightarrow I \times J$,

$$\begin{aligned}
\mathcal{B}/I \times J(\langle v, w\rangle, \langle d_{0,AB}, d_{1,AB}\rangle) &\cong \{(c, h) | A \xleftarrow[(v)]{c} \cdot \xrightarrow[(w)]{h} B\} \\
&\cong \{(c, k) | A \xleftarrow[(v)]{c} \cdot \xrightarrow[(uw)]{k} C\} \qquad ; \text{by } (c, h) \mapsto (c, fh) \\
&\cong \mathcal{B}/I \times K((I \times u)\langle v, w\rangle, \langle d_{0,AC}, d_{1,AC}\rangle),
\end{aligned}$$

which shows that $\langle d_{0,AB}, d_{1,AB}\rangle$ is a pullback of $\langle d_{0,AC}, d_{1,AC}\rangle$ along $(I \times u)$. ▮

For a morphism f, to say that $\mathrm{Hom}(A, f)$ gives the above pullback square is an internal way to say that f is cartesian.

Dually, the (contravariant) fibred 1-cell

$$\mathrm{Hom}(-, A)\colon \quad p^{(\mathrm{op})} \longrightarrow \mathrm{cod}$$

$$\begin{tikzcd}
\mathcal{E}^{(\mathrm{op})} \arrow[r, "\mathrm{Hom}(-, A)"] \arrow[d, "p^{(\mathrm{op})}"'] & \mathcal{B}^{\mathbf{2}} \arrow[d, "\mathrm{cod}"] \\
\mathcal{B} \arrow[r, "(-) \times I"'] & \mathcal{B}
\end{tikzcd}$$

sends $B \in \mathcal{E}_J$ to $\langle d_{0,BA}, d_{1,BA}\rangle\colon \mathrm{Hom}(B,A) \longrightarrow J \times I$, and $[(f,c)]\colon C \longrightarrow B$ in $\mathcal{E}^{(\mathrm{op})}$ over $u\colon K \longrightarrow J$, *i.e.*, $C \xleftarrow[(\mathrm{id})]{f} \cdot \xrightarrow[(u)]{c} B$ to

$$\begin{tikzcd}
\mathrm{Hom}(C,A) \arrow[r, "{\mathrm{Hom}([(f,c)],A)}"] \arrow[d, "{\langle d_{0,CA}, d_{1,CA}\rangle}"'] & \mathrm{Hom}(B,A) \arrow[d, "{\langle d_{0,BA}, d_{1,BA}\rangle}"'] \\
K \times I \arrow[r, "u \times I"'] & J \times I
\end{tikzcd}$$

where $\mathrm{Hom}([(f,c)],A)$ underlies the unique h determined by any representative of the equivalence class $[(f,c)] \circ [(\chi_{CA}, d)]$, *i.e.*, the morphism h that corresponds to $(cd', \chi_{CA}f')$ below by the universality of $(d_{0,BA}, \chi_{BA})$.

$$\begin{tikzcd}
 & \cdot \arrow[dl, "f'"'] \arrow[dr, "\shortmid" marking, "d'", "(d_{0,CA})"'] \arrow[rrr, dashed, "\shortmid" marking, "h"] & & & A' \arrow[ddl, "\shortmid" marking, "(d_{0,BA})"] \arrow[ddd, "\chi_{BA}"] \\
A'' \arrow[dr, "\shortmid" marking, "d", "(d_{0,CA})"'] \arrow[dd, "\chi_{CA}"'] & & \cdot \arrow[dl, "f"'] \arrow[dr, "\shortmid" marking, "c", "(u)"'] & & \\
 & C & & B & \\
A \arrow[rrrr, equal] & & & & A
\end{tikzcd}$$

When p is cloven and $\mathcal{B}$ has binary products, the local smallness of $p\colon \mathcal{E} \longrightarrow \mathcal{B}$ at $(A \in \mathcal{E}_I, B \in \mathcal{E}_J)$ can be expressed by the representability of the functor $H'_{AB}\colon (\mathcal{B}/I \times J)^{\mathrm{op}} \longrightarrow \mathbf{Set}$ that sends $\langle u, v\rangle\colon K \longrightarrow I \times J$ to $\mathcal{E}_K(A[u], B[v])$ and $w\colon \langle u,v\rangle \longrightarrow \langle k,l\rangle$ to the function $f \mapsto f[w]$. The representing object is $\langle d_{0,AB}, d_{1,AB}\rangle\colon \mathrm{Hom}(A,B) \longrightarrow I \times J$ and the counit $\bar{\chi}_{A,B}\colon A[d_0] \longrightarrow B[d_1]$ in $\mathcal{E}_{\mathrm{Hom}(A,B)}$ is the vertical factor of $\chi_{AB}\colon A' \xrightarrow[(d_1)]{} B$.

Example 4.6.4. When $p = p_{\mathcal{C}} \colon \mathbf{Fam}(\mathcal{C}) \longrightarrow \mathbf{Set}$, $p_{\mathcal{C}}$ is locally small if and only if $\mathcal{C}$ is locally small in the usual sense. With the counit

$$\begin{tikzcd}
& A[d_0] \arrow[dl, "|" marking] \arrow[dr, "\chi \equiv (d_1, (f \colon A_i \to B_j)_{(i,j,f)})"] & \\
A & & B
\end{tikzcd}$$

$$\Big\downarrow p_{\mathcal{C}}$$

$$\begin{tikzcd}
& \mathrm{Hom}(A,B) \equiv \{(i,j,f) \mid f \colon A_i \to B_j\} \arrow[dl, "d_0 \equiv ((i,j,f) \mapsto i)"'] \arrow[dr, "d_1 \equiv ((i,j,f) \mapsto j)"] & \\
I & & J,
\end{tikzcd}$$

the natural bijection $\mathcal{E}_K(A[u], B[v]) \cong \mathcal{B}/(I \times J)(\langle u, v \rangle, \langle d_1, d_0 \rangle)$ is given by

$$(A[u] \xrightarrow{(\mathrm{id}_K, (f_k \colon A_{uk} \to B_{vk})_k)} B[v]) \quad \mapsto \quad \begin{pmatrix} K & \longrightarrow & \mathrm{Hom}(A,B) \\ k & \mapsto & (uk, vk, f_k) \end{pmatrix}$$

Note that the morphism $\langle d_0, d_1 \rangle \colon \mathrm{Hom}(A,B) \longrightarrow I \times J$ corresponds to the family $(\mathcal{C}(A_i, B_j))_{i \in I, j \in J}$.

The following "fibrewise" version of local smallness is equivalent to the above definition when $\mathcal{B}$ is finitely complete [78].

Definition 4.6.5. For $A, B \in \mathcal{E}_I$, p is *fibrewise* locally small at (A, B) *over* I if the functor

$$H_{I,AB} \colon \mathcal{E}^{\mathrm{op}}_{\mathbf{gpd}} \longrightarrow \mathbf{Set}$$

$$\begin{array}{ccc}
X & & \{\, (c,g) \mid A \xleftarrow[(u)]{\;c\;} X \xrightarrow[(u)]{\;g\;} B \} \\
f \Big\uparrow & \mapsto & \Big\downarrow (c,g) \mapsto (cf, gf) \\
Y & & \{\, (c',g') \mid A \xleftarrow[(u')]{\;c'\;} Y \xrightarrow[(u')]{\;g'\;} B \}.
\end{array}$$

is representable.

The chosen counit of a representation for $H_{I,AB}$ is denoted by

$$\begin{tikzcd}
A & A' \arrow[l, "|" marking] \arrow[r, "\chi'_{I,AB}"] \arrow[d, mapsto, "p"] & B \\
I & \mathrm{Hom}_I(A,B) \arrow[l, "d_{I,AB}"'] \arrow[r, "d_{I,AB}"'] & J
\end{tikzcd}$$

Proposition 4.6.6. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ with $\mathcal{B}$ finitely complete, p is fibrewise locally small if and only if p is locally small.

Proof For the forward direction, given $A \in \mathcal{E}_I$ and $B \in \mathcal{E}_J$, let $A' \underset{(\pi_{0,IJ})}{\longrightarrow\!\!\!\!\!\!\!+\;\;} A$ and $B' \underset{(\pi_{1,IJ})}{\longrightarrow\!\!\!\!\!\!\!+\;\;} B$ be cartesian morphisms over $\pi_{0,IJ}: I \times J \longrightarrow I$ and $\pi_{1,IJ}: I \times J \longrightarrow J$, respectively. Then, the counit of a representation of H_{AB} is given by

$$\begin{tikzcd}
A & A' \arrow[l, "|" marking, "(\pi_0)"] & A'' \arrow[l, "|" marking, "c"', "(d_{I\times J,A'B'})"] \arrow[r, "\chi_{I\times J,A'B'}"] & B' \arrow[r, "|" marking, "(\pi_1)"'] & B,
\end{tikzcd}$$

where $(c, \chi_{I\times J,A'B'})$ is the counit of a representation of $H'_{I\times J,A'B'}$. For the reverse direction, given $A, B \in \mathcal{E}_I$, the counit of a representation $H'_{I,AB}$ is given by

$$\begin{tikzcd}
A & A' \arrow[l, "|" marking, "d"', "(d_{0,AB})"] & A'' \arrow[l, "|" marking, "c"', "(e)"] \arrow[r, "|" marking, "c", "(e)"'] & A' \arrow[r, "\chi_{AB}", "(d_{1,AB})"'] & B,
\end{tikzcd}$$

where (d, χ_{AB}) is the counit of a representation of H_{AB} and c is any cartesian morphism over the equalizer e of $\mathrm{Hom}(A,B) \overset{d_{0,AB}}{\underset{d_{1,AB}}{\rightrightarrows}} I$ in $\mathcal{B}$. ∎

Analogous to the previous global definition, when p is cloven, $d_{I,AB}$ represents the functor $H'_{I,AB}: (\mathcal{B}/I)^{\mathrm{op}} \longrightarrow \mathbf{Set}$ that sends $u: K \longrightarrow I$ to $\mathcal{E}_K(A[u], B[u])$ and $w: u \longrightarrow v$ to a function $f \mapsto f[w]$. This gives a natural bijection $\mathcal{B}/I(u, d_{I,AB}) \cong \mathcal{E}_K(A[u], B[u])$ whose counit $\chi_{I,AB}: A[u] \longrightarrow B[u]$ is the vertical factor of $\chi'_{I,AB}$ above.

Further, one can equationally characterise the universality of $d_{I,AB}$ as follows. This provides the basis for our presentation of a locally small fibration as a type theory in Chapter 5. Let $\prec u, - \succ: \mathcal{E}_K(A[u], B[u]) \longrightarrow B/I(u, d_{I,AB})$ be a function. Then the three equations

$$d_{I,AB}\prec u, f \succ = u,$$

$$\chi_{I,AB}[\prec u, f\succ] = f,$$
$$\prec uw, f[w]\succ = \prec u, f\succ w \qquad (w\colon L \longrightarrow K)$$

express the natural bijection in the previous paragraph. These three equations make it clear that a Hom-object is similar to a disjoint sum, equipped with a first projection, a second projection, and a surjective dependent pairing; *cf.* Remark 4.1.9 of Jacobs' thesis [37] on encoded "disjoint sums" for comprehension categories.

The following lemma has an application in modelling dependent sum types and dependent coproduct types in type theory, see Section 4.8 and Chapter 5 and 6.

Proposition 4.6.7. Given a fibrewise locally small fibration $p\colon \mathcal{E} \longrightarrow \mathcal{B}$, $u\colon J \longrightarrow I$ in $\mathcal{B}$ and $A, B \in \mathcal{E}_I$, the following square in $\mathcal{B}$ is a pullback diagram.

$$\begin{tikzcd}
\mathrm{Hom}_J(A[u], B[u]) \arrow[r, "\prec u d_{J,A[u]B[u]},\, \chi_{J,A[u]B[u]}\succ"] \arrow[d, "d_{J,A[u]B[u]}"'] & \mathrm{Hom}_I(A, B) \arrow[d, "d_{I,AB}"] \\
J \arrow[r, "u"'] & I
\end{tikzcd}$$

Proof The top morphism underlies the unique cartesian morphism h that makes the following commutes.

$$\begin{tikzcd}
A[u] \arrow[r] & A \\
\cdot \arrow[u, "(d_{J,A[u]B[u]})"] \arrow[r, dashed, "h"] \arrow[d, "\chi_{J,A[u]B[u]}"'] & \cdot \arrow[u, "(d_{(I,AB)})"'] \arrow[d, "\chi_{I,AB}"] \\
B[u] \arrow[r] & B
\end{tikzcd}$$

The proof is similar to that of Proposition 4.6.3. ∎

This shows that the class of those morphisms isomorphic to $d_{I,AB}$ for some $I \in \mathcal{B}$ and $A, B \in \mathcal{E}_I$ satisfies the stability condition of display maps [86,36], *i.e.,* they are closed under pulling back along arbitrary morphisms.

When p has a fibred terminal object $1_p \in \mathcal{E}_1$, (see Section 4.7), for any $X \in \mathcal{E}_I$, the fibrewise Hom-projection $d_{I,1_p[!_I]X}\colon \mathrm{Hom}_I(1_p[!_I], X) \longrightarrow I$ is isomorphic to (global) Hom-projections $\langle d_{0,1_pX}, d_{1,1_pX}\rangle\colon \mathrm{Hom}(1_p, X) \longrightarrow 1 \times I$. Under this isomorphism, the top morphism of the above diagram corresponds to $\mathrm{Hom}(1_p, \hat{u}(X))\colon \mathrm{Hom}(1_p, X[u]) \longrightarrow \mathrm{Hom}(1_p, X)$. So, in this case, Proposition 4.6.7 is an instance of Proposition 4.6.3. This also shows that a fibration with a fibred terminal that is locally small at $(1, X)$ for every $X \in \mathcal{E}$ gives rise to a comprehension category [37] $\mathrm{Hom}(1, -)$. For more on locally small fibrations and comprehension categories, see Section 4.5 of *ibid.*

Example 4.6.8. When $p = \mathrm{cod}\colon \mathcal{B}^{\mathbf{2}} \longrightarrow \mathcal{B}$, p is fibrewise locally small if and only if $\mathcal{B}$ is locally cartesian closed [15]. For brevity, assume $\mathcal{B}$ has chosen pullbacks, *i.e.*, p is cloven. Then, for g, h in $\mathcal{B}/I$, and $u\colon J \longrightarrow I$, the natural bijection $\mathcal{B}/I(u, d_{I,fg}) \cong \mathcal{B}/J(f[u], g[u])$ gives that $d_{I,gh}$ satisfies the same universality in $\mathcal{B}/I$ as h^g.

$$
\begin{array}{cl}
u \longrightarrow d_{I,gh} & \\
\hline
g[u] \longrightarrow h[u] \in \mathcal{B}/J & \cong \\
\hline
u(g[u]) \longrightarrow h \in \mathcal{B}/I & \cong \\
\hline
u \times g \longrightarrow h \in \mathcal{B}/I & \cong
\end{array}
$$

4.7 Fibred limits and fibred ccc

Fibred version of the limit concept is defined by that for the ordinary category and the concept of limit preservation. So, the analysis of Chapter 2 applies.

Definition 4.7.1. Given a fibration $p\colon \mathcal{E} \longrightarrow \mathcal{B}$ and a functor $F\colon \mathcal{I} \longrightarrow \mathcal{E}$ with a small $\mathcal{I}$, a fibred limit of D in p is a limit of F in $\mathcal{E}$ that is preserved by p.

In the rest of this section, we spell out this definition for terminal objects and binary products and give several equivalent formulations of them. These equivalent formulations can be generalised for any fibred limit. For 2-categorical treatment of fibred limits, see [34].

4.7.1 Terminal

A fibred terminal object 1_p (the subscript is often omitted) of p is a terminal object of $\mathcal{E}$ such that $p(1_p)$ is a terminal object of $\mathcal{B}$.

Fibrewise, this is equivalent to the following.

Lemma 4.7.2. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$, p has a fibred terminal object if and only if each fibre has a terminal object preserved by the substitution functors.

Proof If $X \overset{c}{\longrightarrow\!\!\!\!\!\!\!+\;\;} 1_p$ then c is necessarily over the unique $pX \xrightarrow{!} p1_p$ and

$$\mathcal{E}_{pX}(Y, X) \cong \mathcal{E}_!(Y, 1_p) \cong \{*\}.$$

∎

Using $\mathrm{Hom}(-,-)$, one can internalise the definition.

Lemma 4.7.3. An object $T \in \mathcal{E}$ is a fibred terminal object if and only if, for any $X \in \mathcal{E}_I$, p is locally small at (X, T) and $d_{0,XT}: \mathrm{Hom}(X, T) \longrightarrow I \times pT$ is an isomorphism natural in X; *i.e.*, for any $f: X \longrightarrow Y$ in $\mathcal{E}^{(\mathrm{op})}$, $d_{0,XT}\mathrm{Hom}(f, T) = d_{0,YT}$.

Proof By the definition of $\mathrm{Hom}(X, T)$. ∎

Example 4.7.4. When $p = p_{\mathcal{C}}: \mathbf{Fam}(\mathcal{C}) \longrightarrow \mathbf{Set}$, p has a fibred terminal object if and only if $\mathcal{C}$ has a terminal object. When $p = \mathrm{cod}: \mathcal{B}^{\mathbf{2}} \longrightarrow \mathcal{B}$, p has a fibred terminal object if and only if $\mathcal{B}$ has a terminal object.

4.7.2 Binary products

Given two objects $A \in \mathcal{E}_I$ and $B \in \mathcal{E}_J$, a fibred binary product of A and B is given by a product diagram

$$A \xleftarrow{\pi_0} A \times B \xrightarrow{\pi_1} B$$

in $\mathcal{E}$ such that $I \xleftarrow{p\pi_0} p(A \times B) \xrightarrow{p\pi_1} J$ is a product diagram in $\mathcal{B}$.

Example 4.7.5. When $p = p_{\mathcal{C}}: \mathbf{Fam}(\mathcal{C}) \longrightarrow \mathbf{Set}$, p has fibred binary products if and only if $\mathcal{C}$ has binary products.

Proposition 4.7.6. Given $A \in \mathcal{E}_I$ and $B \in \mathcal{E}_J$, an object P of $\mathcal{E}_K$ is a fibred product of A and B if and only if K is a product of I and J and there is a natural bijection

$$E_u(X, P) \cong E_{\pi_0 u}(X, A) \times E_{\pi_1 u}(X, B).$$

where $I \xleftarrow{\pi_0} K \xrightarrow{\pi_1} J$ is a product diagram in $\mathcal{B}$.

Proof The necessity is obvious. To see this is sufficient, note that there is a natural bijection

$$\begin{aligned}
\mathcal{E}(X, P) &\cong \coprod_{u: L \longrightarrow K} \mathcal{E}_u(X, P) \\
&\cong \coprod_u (\mathcal{E}_{\pi_0 u}(X, A) \times \mathcal{E}_{\pi_1 u}(X, B)) \\
&\cong \coprod_{v: L \longrightarrow I, w: L \longrightarrow J} (\mathcal{E}_v(X, A) \times \mathcal{E}_w(X, B)) \\
&\cong \mathcal{E}(X, A) \times \mathcal{E}(X, B)
\end{aligned}$$

and that its counit is over (π_0, π_1).

When p is a locally small fibration, one have the following internal version.

Proposition 4.7.7. Let $p: \mathcal{E} \longrightarrow \mathcal{B}$ be a locally small fibration with $\mathcal{B}$ having finite limits. Given $A \in \mathcal{E}_I$ and $B \in \mathcal{E}_J$, for each $X \in \mathcal{E}_K$, consider the following pullback in $\mathcal{B}$.

$$\begin{tikzcd}
V_X \arrow[r, "\pi_{0X}"] \arrow[d, "\pi_{1X}"'] \arrow[dr, "d_{0,XA} \times d_{0,XB}" description] \arrow[dr, phantom, very near start, "\lrcorner"] & \mathrm{Hom}(X, A) \arrow[d, "d_{0,XA}"] \\
\mathrm{Hom}(X, B) \arrow[r, "d_{0,XB}"'] & K
\end{tikzcd}$$

Also, for each $f: X \longrightarrow Y$ in $\mathcal{E}^{(\mathrm{op})}$, let $V_f: V_X \longrightarrow V_Y$ be the morphism determined by $(\mathrm{Hom}(f, A)\pi_{0X}, \mathrm{Hom}(f, B)\pi_{1X})$ and the universality of V_Y. Then, $P \in \mathcal{E}_{I \times J}$ is a fibred binary product of A and B if and only if, for all $X \in \mathcal{E}_K$,

$$\begin{tikzcd}
V_X \arrow[r, "\langle d_{1,XA}\pi_{1X}, d_{1,XB}\pi_{0X} \rangle"] \arrow[d, "d_{0,XA} \times d_{0,XB}"'] & I \times J \\
K &
\end{tikzcd}$$

exhibits V_X as $\mathrm{Hom}(X, P)$ and V_f as $\mathrm{Hom}(f, P)$.

Proof Again, by unwinding the definition of $\mathrm{Hom}(-, -)$, it is straightforward to see that this is equivalent to the formulation given in Proposition 4.7.6. ▮

The fibrewise version of binary product is defined below and is equivalent to the original global definition.

Definition 4.7.8. Given $A, B \in \mathcal{E}_I$, $P \in \mathcal{E}_I$ is a fibrewise binary product, written $A \times_I B$, of A and B if there is a natural bijection

$$\mathcal{E}_u(X, P) \cong \mathcal{E}_u(X, A) \times \mathcal{E}_u(X, B).$$

For a fibrewise locally small fibration, this can be internally expressed by:

Proposition 4.7.9. Given a fibrewise locally small fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$, $P \in \mathcal{E}_I$ is a fibrewise binary product of A and B in $\mathcal{E}_I$ if and only if, for any $X \in \mathcal{E}_I$, $d_{I,XP} \cong d_{I,XA} \times d_{I,XB}$ in $\mathcal{B}/I$ and natural in X in the obvious sense. ▮

With the substitution functors, one has

Lemma 4.7.10. Given a cloven fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ and $A, B \in \mathcal{E}_I$, $P \in \mathcal{E}_I$ is a fibrewise binary product of A and B if and only if P is a product of them in $\mathcal{E}_I$ preserved by any substitution functor.

Proof For $u: J \longrightarrow I$ and $X \in \mathcal{E}_J$, there is a natural bijection

$$\begin{aligned}
\mathcal{E}_u(X, P) &\cong \mathcal{E}(X, P[u]) \\
&\cong \mathcal{E}(X, A[u]) \times \mathcal{E}(X, B[u]) \\
&\cong \mathcal{E}_u(X, A) \times \mathcal{E}_u(X, B)
\end{aligned}$$

∎

The equivalence of the global and fibrewise definitions can be seen as follows.

Lemma 4.7.11. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ with $\mathcal{B}$ having binary products, p has all fibred binary product if and only if p has all fibrewise binary products.

Proof For the forward direction, a fibrewise binary product of $A, B \in \mathcal{E}_I$ is given by $(A \times B)[\langle \mathrm{id}_A, \mathrm{id}_A \rangle]$. For the reverse direction, a fibred product of $A \in \mathcal{E}_I$ and $B \in \mathcal{E}_J$ is given by $A[\pi_{0,IJ}] \times_{I \times J} B[\pi_{1,IJ}]$. ∎

The forward direction of the above proof is a special case of the following, where fibrewise binary products arise from fibred binary products.

Lemma 4.7.12. Consider the following diagram

$$\begin{tikzcd}
A' \arrow[d, "(u)"'] & P' \arrow[l, dashed, "(\mathrm{id}_K)"] \arrow[r, dashed, "(\mathrm{id}_K)"'] \arrow[d, "(\langle u, v \rangle)"'] & B' \arrow[d, "(v)"] \\
A & P \arrow[l, "\pi"', "(\pi)"] \arrow[r, "\pi'", "(\pi')"'] & B
\end{tikzcd}$$

where $K \in \mathcal{B}$ and $A \longleftarrow P \longrightarrow B$ is a fibred product diagram over $I \xleftarrow{\pi} I \times J \xrightarrow{\pi'} J$. Then, $A' \underset{(\mathrm{id}_K)}{\dashleftarrow} P' \underset{(\mathrm{id}_K)}{\dashrightarrow} B'$ is a fibrewise product diagram.

Proof

$$\begin{aligned}\mathcal{E}_w(X,P') &\cong \mathcal{E}_{\langle u,v\rangle w}(X,P)\\ &\cong \mathcal{E}_{\pi\langle u,v\rangle w}(X,A)\times\mathcal{E}_{\pi'\langle u,v\rangle w}(X,B)\\ &\cong \mathcal{E}_{uw}(X,A)\times\mathcal{E}_{vw}(X,B)\\ &\cong \mathcal{E}_w(X,A')\times\mathcal{E}_w(X,B').\end{aligned}$$

4.7.3 Exponentials

A *fibred cartesian closed categories* is defined here in terms of each fibre. For a global version, see Hermida's thesis [34]. In Chapter 7, we see a concrete example of fibred ccc, deliverables.

Definition 4.7.13. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ be with fibred / fibrewise binary products, p is a *fibred cartesian closed category* if each fibre is cartesian closed and if the substitution functors preserve cartesian closure.

Example 4.7.14. Given a category $\mathcal{C}$, the family fibration $p_{\mathcal{C}}$ is a fibred ccc if and only if $\mathcal{C}$ is a ccc.

4.8 $\mathcal{B}$-coproducts and $\mathcal{B}$-products

Given a fibration, one can define coproducts and products of objects in the total category *indexed by objects of the base.* They are called $\mathcal{B}$-coproducts and $\mathcal{B}$-products [15]. They are of interest in computer science since a modified version of them are used to model dependent sum types and dependent product types in type theory, see Chapter 5 and 6 for the detail.

Definition 4.8.1. A *cofibration* $p: \mathcal{E} \longrightarrow \mathcal{B}$ is a fibration $p^{\mathrm{op}}: \mathcal{E}^{\mathrm{op}} \longrightarrow \mathcal{B}^{\mathrm{op}}$. A morphism is called *cocartesian* if it is cartesian for p^{op}.

In other words, $f: X \longrightarrow Y$ is cocartesian if the composition $(-) \circ f$ restricts to, for each $u: pY \longrightarrow I$ and $Z \in \mathcal{E}_I$, a bijection $\mathcal{E}_u(Y, Z) \cong \mathcal{E}_{u \circ pf}(X, Z)$. We denote a cocartesian morphism over u by $X \underset{(u)}{\multimap} Y$.

Definition 4.8.2. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$, p *has $\mathcal{B}$-coproducts* if it is a cofibration such that, for any $X \in \mathcal{E}_J$ and any pullback square in $\mathcal{B}$,

$$\begin{tikzcd} L \arrow[r, "v'"] \arrow[d, "u'"'] \arrow[dr, phantom, very near start, "\lrcorner"] & J \arrow[d, "u"] \\ K \arrow[r, "v"'] & I, \end{tikzcd}$$

the unique morphism f determined by any cartesian morphisms over v and v' and any cocartesian morphism over u, is cocartesian over u'.

$$\begin{tikzcd} X_1 \arrow[r, "+"{marking}, "(v')"'] \arrow[d, dashed, "f"', "(u')"] & X \arrow[d, "(u)"'] \\ X_2 \arrow[r, "+"{marking}, "(v)"'] & X_0 \end{tikzcd}$$

The condition in the definition is called the *Beck-Chevalley* condition.

Example 4.8.3. For $p = p_{\mathcal{C}}: \mathbf{Fam}(\mathcal{C}) \longrightarrow \mathbf{Set}$, p has $\mathcal{B}$-coproducts if and only if $\mathcal{C}$ has (small) coproducts.

When p is cloven, one can give a fibrewise formulation of this. Let us say p is *cocloven* if it is equipped with a choice, for each $(X, u: J \longrightarrow I)$ with $X \in \mathcal{E}_J$, of a cocartesian morphism $X \underset{(u)}{\multimap} X'$.

Proposition 4.8.4. Given a cloven fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$, p is cocloven and has $\mathcal{B}$-coproduct if and only if

- for any $u: J \longrightarrow I$, the substitution functor $(-)[u]: \mathcal{E}_I \longrightarrow \mathcal{E}_J$ has a left adjoint $\Sigma_u: \mathcal{E}_J \longrightarrow \mathcal{E}_I$, and

- for any $X \in \mathcal{E}_J$ and any pullback square

$$\begin{tikzcd} L \arrow[r, "v'"] \arrow[d, "u'"'] \arrow[dr, phantom, "\lrcorner", very near start] & J \arrow[d, "u"] \\ K \arrow[r, "v"'] & I, \end{tikzcd}$$

  the morphism $\Sigma_{u'}(X[v']) \xrightarrow[(\mathrm{id})]{} (\Sigma_u X)[v]$ given by

$$\cfrac{\cfrac{\cfrac{X \xrightarrow[(\mathrm{id})]{\eta_X} (\Sigma_u X)[u]}{X[v'] \xrightarrow[(\mathrm{id})]{} (\Sigma_u X)[uv']} \; ; f \mapsto f[v']}{X[v'] \xrightarrow[(\mathrm{id})]{} (\Sigma_u X)[vu']} \cong}{\Sigma_{u'}(X[v']) \xrightarrow[(\mathrm{id})]{} (\Sigma_u X)[v]} \cong$$

  is an isomorphism (η is the unit of the adjunction $\Sigma_u \dashv (-)[u]$).

Proof For the forward direction, define $\Sigma_u X$ by the codomain X' of the chosen cocartesian morphism $c_X : X \xrightarrow[(u)]{} X'$. One can extend this to a functor $\Sigma_u : \mathcal{E}_I \longrightarrow \mathcal{E}_J$ in the same way as the definition of substitution functors. Then, Σ_u is a left adjoint to $(-)[u]$ as

$$\cfrac{\cfrac{X \xrightarrow[(\mathrm{id})]{} Y[u]}{X \xrightarrow[(u)]{} Y} \cong; \qquad f \mapsto \widehat{u}(Y)f}{\Sigma_u X \xrightarrow[(\mathrm{id})]{} Y} \cong; \qquad \text{the inv. of } f \mapsto f c_X$$

The second condition is equivalent to the Beck-Chevalley condition. For the reverse direction, given $(X, u : J \longrightarrow I)$ with $X \in \mathcal{E}_J$, the morphism $\widehat{u}(X)\eta_X$ from X to $\Sigma_u X$ is cocartesian with the similar argument as the forward direction. Again, the Beck-Chevalley condition is equivalent to the second condition. ∎

The object $\Sigma_u X$ is said to be the coproduct of X *along* u.

The dual to $\mathcal{B}$-coproducts are $\mathcal{B}$-products.

Definition 4.8.5. Given a fibration $p : \mathcal{E} \longrightarrow \mathcal{B}$, p *has* $\mathcal{B}$-products if the opposite fibration $p^{(\mathrm{op})} : \mathcal{E}^{(\mathrm{op})} \longrightarrow \mathcal{B}$ has $\mathcal{B}$-coproducts.

By the definition of $p^{(\mathrm{op})}$, this is equivalent to requiring the following.

- For every $(X, u\colon I \longrightarrow J)$ with $X \in \mathcal{E}_J$, there is an object $\Pi_u X \in \mathcal{E}_J$ and a pair of morphisms $X \xleftarrow[(\mathrm{id})]{\varepsilon_X} \cdot \xrightarrow[(u)]{c_X} \Pi_u X$.
- Given any $v\colon J \longrightarrow K$ and $Z \in \mathcal{E}_K$, for each $X \xleftarrow[(\mathrm{id})]{f} \cdot \xrightarrow[(uv)]{a} Z$, there exists, unique up to the equivalence, a pair of morphisms $\Pi_u X \xleftarrow[(\mathrm{id})]{\bar{f}} \cdot \xrightarrow[(v)]{\bar{a}} Z$ that satisfies the following condition: the pair $(\bar{f}, \bar{a})$ satisfies $\varepsilon_X g = f$, where g is determined by $c_X g = \bar{f} b$ and b, in turn, is determined by $b\bar{a} = a$.

$$\begin{tikzcd}
Z & \cdot \arrow[l, "\bar{a}", "(v)"', "\shortmid" marking] \arrow[r, "\bar{f}", "(\mathrm{id})"'] & \Pi_u X \\
 & & \cdot \arrow[u, "(u)", "c_X"', "\shortmid" marking] \arrow[d, "(\mathrm{id})", "\varepsilon_X"'] \\
W \arrow[uu, "a", "(uv)"', "\shortmid" marking] \arrow[uur, "b", "(u)"', "\shortmid" marking] \arrow[urr, "g", "(\mathrm{id})"'] \arrow[rr, "f", "(\mathrm{id})"'] & & X
\end{tikzcd}$$

- Given $X \in \mathcal{E}_J$ and any pullback diagram in $\mathcal{B}$

$$\begin{tikzcd}
L \arrow[r, "v'"] \arrow[d, "u'"'] \arrow[dr, phantom, "\lrcorner", very near start] & J \arrow[d, "u"] \\
K \arrow[r, "v"'] & I,
\end{tikzcd}$$

the pair of morphisms (f, c) determined by any choice of the three horizontally drawn cartesian arrows has the same universal property as (ε_Y, c_Y) for (Y, u').

$$\begin{tikzcd}
\cdot \arrow[r, "(v')"', "\shortmid" marking] & \Pi_u X \\
\cdot \arrow[u, "c", "(u')"', "\shortmid" marking] \arrow[r, "(v')"', "\shortmid" marking] \arrow[d, "f"', "(\mathrm{id})"] & \cdot \arrow[u, "(u)", "c_X"', "\shortmid" marking] \arrow[d, "(\mathrm{id})"', "\varepsilon_X"] \\
Y \arrow[r, "(v)"', "\shortmid" marking] & X
\end{tikzcd}$$

We omit the dual to Proposition 4.8.4 as it is easy to see.

Example 4.8.6. The codomain fibration $\mathrm{cod}\colon \mathcal{B}^{\mathbf{2}} \longrightarrow \mathcal{B}$ has $\mathcal{B}$-product if and only if $\mathcal{B}$ is locally cartesian closed.

In computer science applications, often not all (co)products along all base morphisms are needed, and requiring all of them is too strong a condition. When we restrict our attention to fibrewise locally small fibrations, those fibrations having (co)products along $d_{I,AB}\colon \mathrm{Hom}_I(A,B) \longrightarrow I$ for some $A, B \in \mathcal{E}_I$ are of particular interest, since only those are needed to model dependent sum types and dependent product types. In this case, one can restrict pullbacks in the Beck-Chevelley condition to those pullbacks of Proposition 4.6.7. So, we define the following.

Definition 4.8.7. Given a fibrewise locally small fibration $p\colon \mathcal{E} \longrightarrow \mathcal{B}$, *p has products along* Hom-*projections* if, given any two objects A and B in $\mathcal{E}_I$, for any object X in $\mathcal{E}_{\mathrm{Hom}_I(A,B)}$ there exists a cocartesian morphism $X \underset{(d_{I,AB})}{\longrightarrow\!\!\circ\!\!\longrightarrow} X'$ over the projection $d_{I,AB}\colon \mathrm{Hom}_I(A,B) \longrightarrow I$ and moreover if, for any $u\colon J \longrightarrow I$, $A, B \in \mathcal{E}_I$, the morphism f determined by the three other cartesian and cocartesian morphisms below, is cartesian.

$$\begin{tikzcd}
X_1 \arrow[rr, "+", "(\prec u d_{J,A[u]B[u]},\, \chi_{J,A[u]B[u]} \succ)"'] \arrow[d, dashed, "f"', "(d_{J,A[u]B[u]})"] & & X \arrow[d, "\circ", "(d_{I,AB})"'] \\
X_2 \arrow[rr, "+", "(u)"'] & & X'
\end{tikzcd}$$

A fibration p has products along Hom-projections if $p^{(\mathrm{op})}$ has coproducts along Hom-projections.

Jacobs' [37] gave a formulation of products and coproducts along a restricted class of morphisms in terms of comprehension categories, which is closely related to locally small fibrations; see the remark after Proposition 4.6.7, also see Jacobs' thesis [37].

Chapter 5

Type theory and fibrations 1

5.1 Introduction

In Chapter 1 and 2, we developed a methodology to treat universal structure on categories in a computer checked manner using a proof development system LEGO. The following chapters study a categorical framework for type theory. This is motivated by the fact that the most common application of categorical structure in computer science is to provide models for type theories. Also, it is often the case that categorical structure of our objects of studies, *e.g.*, processes, becomes apparent via consideration of their type theory. By studying a categorical framework, we aim to raise availability of categorical methods for those who are more familiar with type theories. The methodology developed in the previous chapters can then be used to develop and formalise various application of type theory, as well as studies of type theory itself.

By a framework, we mean a uniform family of categorical structures that provide semantics for various type systems. This study of frameworks for semantics, rather than those for syntax *á la* Logical Framework [32,23], is proposed by Power; his papers [65,66] outline his program and give the references for the substantial development achieved so far. The framework chosen in his program is algebraic structure, or more generally, essentially algebraic structure on categories. From a different view point, Jacobs [37] classifies various type systems in terms of TS-settings and TS-features and then give their categorical counterparts as a way to organise categorical models for various type systems.

Our study differs from these in that we are interested not only in the *semantics* of a semantic framework, but also in its formalisation, including consideration for its syntax. Therefore, we provide syntax for categorical structures that model a general class of type systems and give, using the syntax, a detailed account of relationship between type systems and those structures. The syntax is presented as a type system that is directly derived from those structures. This allows us to give a detailed account of relationship between traditional type systems and these categorical structures.

There have been studies of categorical combinators which also provide syntax directly derived from categorical structure, *e.g.* Curien, Ritter *et. al.*'s Categorical Abstract Machines [18,19,70]. Hagino [30,29] studied Categorical Programming Language based on categorical structure of dialgebras. Cockett and Spencer [16, 17,75] advanced this with functorial strength in their work on CHARITY. Unlike these studies, which proposed alternative computation mechanisms , we do not give reduction rules for our syntax, but emphasise the detailed study of relationship with traditional type systems.

Formalisation of semantics of type theory within an ambient type theory has been studied by authors including McKinna and Pollack [53], Altenkirch [1], and Pollack [63]. The semantics used there is of an operational flavour.

In this chapter, we present a general class of four traditional type systems, and devise a new type system, $\mathcal{T}F$, that provides syntax for locally small fibrations with particular extra structure. This provides the basis for the next chapter, where we establish the relationship between these four type systems and $\mathcal{T}F$.

In Section 2, we define four type systems. Our terminology is that a type *theory* is specified by a type *system* together with sets of constants and axioms. We give detailed definitions to explain the complex interplay between well-formedness of expressions and derivability of judgement. We also define interpretations from a type theory to another, to form a category of type theories in Chapter 6.

Section 3 defines our type system, $\mathcal{T}F$, that describes fibrewise locally small fibrations with a fibred terminal object, products and coproducts along Hom-projections. The syntax of $\mathcal{T}F$ is directly derived from the categorical structure so that passing from the syntax to objects / morphisms of fibrations is trivial. As such, terms of $\mathcal{T}F$ have domain and codomain like morphisms; substitution is an explicit operation rather than a metatheoretic one; variables are replaced by generic terms, avoiding problems concerning variable names. Again, a notion of interpretations between $\mathcal{T}F$-theories is defined.

A locally small fibration is an instance of categories with universal structure. However, it is not convenient to strictly apply our definition of universal structure in Chapter 2 in order to study the role of the structure in the relationship with type theory. What we need is a study of universal structure on fibrations, rather than on mere categories, which is unfortunately unsatisfactory so far. Therefore, $\mathcal{T}F$ is based on algebraic presentation using operations and equations. This does not prevent us from applying our methodology in each fibre category when it is equipped with universal structure. In Chapter 8, we discuss a possible way to extend the first part of thesis to universal structures on fibrations so that we can treat structure across different fibre categories.

5.2 Type dependency

This section introduces four type systems having type dependency with the usual syntax. They are Generalised Algebra $\mathcal{G}A$ [13], λP1 with dependent products, $w\mathcal{ML}$ with dependent products and weak dependent coproducts, and $\mathcal{ML}$ with strong ones.

5.2.1 Generalised algebraic theories

A basic type system handling dependent types is *generalised algebra*($\mathcal{G}A$) introduced by Cartmell [13,14]. In the more modern terms of Jacobs' classification [37], this is the type system with setting ($Sort = \{*\}$, $\succ = \{(*, *)\}$) and feature 'constants'.

A *theory* in $\mathcal{G}A$ or a $\mathcal{G}A$ *theory* is a generalisation of a usual many-sorted equational theory. In $\mathcal{G}A$, the notion of type is generalised to that of *type expressions* constructed from *constant types*, which may contain variables of other types, e.g. $\mathrm{F}(x_1, x_2)$. A type expression containing variables, which can vary depending on terms substituted for variables, is called a *dependent type*.

The notation $E[x]$ for an expression E of a $\mathcal{G}A$ theory means that x is a free variable of E which may occur vacuously. $E[x/A]$ denote the substitution of the expression A for x and $E[\vec{x}/\vec{A}]$ the simultaneous substitution of $\vec{A} = A_1, \cdots, A_n$ for $\vec{x} = x_1, \cdots, x_n$. Parentheses are used to show the precedence, to denotes applications as in $\mathrm{F}(A_1, A_2)$, and to list all the free variables of E as in $E(x_1, \cdots, x_n)$.

For readers not familiar with type dependency, we give a leisurely account of the formal syntax of $\mathcal{G}A$ theories here and omit such details for the system $\mathcal{T}F$ later.

The precise definition of a $\mathcal{G}A$ *theory* is tedious because of the complex type discipline. In the generalised algebraic case, type and operator symbols must be introduced in a theory with justification, which is done by showing that relevant expressions are well-formed; in turn, well-formedness of an expression depends on derivability of equations in the theory whose language is now being defined. We follow Cartmell's treatment [14] with some modification to avoid this difficulty. (See also Hyland and Pitts [36].)

Let

- V be a set of (countably many) variables, for which we use metavariables $x, x_1, x_2, y, \cdots$.
- W_{Ty} be a set of constant type symbols, with metavariables $\mathrm{F}, \mathrm{F}_1, \cdots$.
- W_{Tm} be a set of constant term symbols, with metavariables $\mathrm{C}, \mathrm{C}_1, \cdots$.

Definition 5.2.1. The set of possibly not well-formed *type expressions* (written $K, L, G, H, K_1, \cdots$) and *term expressions* (written $A, A_1, B, \cdots$) relative to V and W are defined inductively by

- for all $x \in V$, x is a term expression.

- for all $\mathrm{F} \in W_{Ty}$ and $A_1, \cdots, A_n$ $(0 \leq n)$ term expressions, $\mathrm{F}(A_1, \cdots, A_n)$ is a type expression.
- for all $\mathrm{C} \in W_{Tm}$ and $A_1, \cdots, A_n$ $(0 \leq n)$ term expressions, $\mathrm{C}(A_1, \cdots, A_n)$ is a term expression.

Definition 5.2.2. A *context* $x_1{:}\,K_1, \cdots, x_n{:}\,K_n$, also written $\vec{x}{:}\,\vec{K}$, is a finite sequence of distinct pairs (variable, type expression).

A *judgement* is an expression of one of the following five forms.

(i) (Structural judgements for context) $\vec{x}{:}\,\vec{K}$ **Context** (read "$\vec{x}{:}\,\vec{K}$ is a well-formed context.")

(ii) (Structural judgements for types) $\vec{x}{:}\,\vec{K} \vdash H{:}\,\mathbf{Type}$ ("H is a well-formed type in context $\vec{x}{:}\,\vec{K}$.")

(iii) (Structural judgements for terms) $\vec{x}{:}\,\vec{K} \vdash A{:}\,H$ ("A is a well-formed term of type H under a context $\vec{x}{:}\,\vec{K}$.")

(iv) (Equality judgements for types) $\vec{x}{:}\,\vec{K} \vdash G = H{:}\,\mathbf{Type}$ ("G and H are equal well-formed types under a context $\vec{x}{:}\,\vec{K}$.")

(v) (Equality judgements for terms) $\vec{x}{:}\,\vec{K} \vdash A = B{:}\,H$ ("A and B are equal well-formed terms of type H in context $\vec{x}{:}\,\vec{K}$.")

Definition 5.2.3. A *pretheory* T is determined by following data (relative to V, W_{Ty}, and W_{Tm}).

(i) for each constant type symbol F, an associated structural judgement J^T_{F} of the form $\vec{x}{:}\,\vec{K} \vdash \mathrm{F}(\vec{x}){:}\,\mathbf{Type}$ with some $\vec{x}{:}\,\vec{K}$.

(ii) for each constant term symbol C, an associated structural judgement J^T_{C}, of the form $\vec{x}{:}\,\vec{K} \vdash \mathrm{C}(\vec{x}){:}\,H$ with some $\vec{x}{:}\,\vec{K}$ and H.

(iii) a set Ax_T of *axioms* consisting of equality judgements for types or terms.

Definition 5.2.4. A judgement J is *derivable* in a pretheory T if there is a labelled finite tree (*derivation*) such that the root node is labelled with J, the leaves are empty, and each branching has a label

$$\frac{J_1 \quad J_2 \quad \cdots \quad J_n}{J_0}$$

which is an instance of the following *inference rules*. An inference rule such that the conclusion J_0 is a structural (equality) judgement is called a *structural* (*equality*) *inference rule*. (Notations $\vec{x}\colon \vec{K}\vdash H, H'\colon \mathbf{Type}$ etc. are abbreviations of "$\vec{x}\colon \vec{K}\vdash H\colon \mathbf{Type}\ \vec{x}\colon \vec{K}\vdash H'\colon \mathbf{Type}$" etc..)

$$\frac{}{(\ )\ \mathbf{Context}}\ \text{(context1)}$$

$$\frac{\vec{x}\colon \vec{K}\vdash K_{n+1}\colon \mathbf{Type}}{\vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1}\ \mathbf{Context}}\ \text{(context2)}\quad \text{(provided } x_{n+1} \text{ is distinct from the } x_i\text{'s.)}$$

$$\frac{\vec{x}\colon \vec{K}\ \mathbf{Context}}{\vec{x}\colon \vec{K}\vdash F(\vec{x})\colon \mathbf{Type}}\ \text{(constant1)}\quad \text{(for each } J^T_{\mathrm{F}} \text{ of the form } \vec{x}\colon \vec{K}\vdash (F(\vec{x}))\colon \mathbf{Type})$$

$$\frac{\vec{x}\colon \vec{K}\vdash H\colon \mathbf{Type}}{\vec{x}\colon \vec{K}\vdash C(\vec{x})\colon H}\ \text{(constant2)}\quad \text{(for each } J^T_{\mathrm{C}} \text{ of the form } \vec{x}\colon \vec{K}\vdash C(\vec{x})\colon H)$$

$$\frac{\vec{x}\colon \vec{K}\vdash K_{n+1}\colon \mathbf{Type}}{\vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1}\vdash x_{n+1}\colon K_{n+1}}\ \text{(assumption)}$$

$$\frac{\vec{x}\colon \vec{K}\vdash K_{n+1}\colon \mathbf{Type} \quad \vec{x}\colon \vec{K}\vdash H\colon \mathbf{Type}}{(\vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1})\vdash H\colon \mathbf{Type}}\ \text{(weakening1)}$$

$$\frac{\vec{x}\colon \vec{K}\vdash K_{n+1}\colon \mathbf{Type} \quad \vec{x}\colon \vec{K}\vdash A\colon H}{(\vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1})\vdash A\colon H}\ \text{(weakening2)}$$

$$\frac{\vec{x}\colon \vec{K}\vdash \mathrm{F}(\vec{x})\colon \mathbf{Type} \quad \vec{y}\colon \vec{G}\vdash A_i\colon K_i[x_1, \cdots x_{i-1}/A_1, \cdots A_{i-1}] \quad {\scriptstyle (1\le i\le n)}}{\vec{y}\colon \vec{G}\vdash \mathrm{F}(\vec{A})\colon \mathbf{Type}}\ \text{(subst1)}$$

$$\frac{\vec{x}\colon \vec{K}\vdash \mathrm{C}(\vec{x})\colon H \quad \vec{y}\colon \vec{G}\vdash A_i\colon K_i[x_1, \cdots x_{i-1}/A_1, \cdots A_{i-1}] \quad {\scriptstyle (1\le i\le n)}}{\vec{y}\colon \vec{G}\vdash \mathrm{C}(\vec{A})\colon H[\vec{x}/\vec{A}]}\ \text{(subst2)}$$

$$\frac{\vec{x}\colon \vec{K}\vdash H, H'\colon \mathbf{Type}}{\vec{x}\colon \vec{K}\vdash H = H'\colon \mathbf{Type}}\ \text{(axiom1)}\quad \text{(provided } \vec{x}\colon \vec{K}\vdash H = H'\colon \mathbf{Type} \quad \in Ax_T)$$

$$\frac{\vec{x}\colon \vec{K}\vdash A, A'\colon H}{\vec{x}\colon \vec{K}\vdash A = A'\colon H}\ \text{(axiom2)}\quad \text{(provided } \vec{x}\colon \vec{K}\vdash A = A'\colon H \quad \in Ax_T)$$

$$\frac{\vec{x}\colon \vec{K}\vdash H\colon \mathbf{Type}}{\vec{x}\colon \vec{K}\vdash H = H\colon \mathbf{Type}}\ \text{(reflexivity1)} \qquad \frac{\vec{x}\colon \vec{K}\vdash A\colon H}{\vec{x}\colon \vec{K}\vdash A = A\colon H}\ \text{(reflexivity2)}$$

$$\frac{\vec{x}\colon \vec{K}\vdash H = H'\colon \mathbf{Type}}{\vec{x}\colon \vec{K}\vdash H' = H\colon \mathbf{Type}}\ \text{(symmetry1)} \qquad \frac{\vec{x}\colon \vec{K}\vdash A = A'\colon H}{\vec{x}\colon \vec{K}\vdash A' = A\colon H}\ \text{(symmetry2)}$$

$$\frac{\vec{x}{:}\,\vec{K}\vdash H_1 = H_2{:}\,\mathbf{Type} \quad \vec{x}{:}\,\vec{K}\vdash H_2 = H_3{:}\,\mathbf{Type}}{\vec{x}{:}\,\vec{K}\vdash H_1 = H_3{:}\,\mathbf{Type}}\ (\text{transitivity1})$$

$$\frac{\vec{x}{:}\,\vec{K}\vdash A_1 = A_2{:}\,H \quad \vec{x}{:}\,\vec{K}\vdash A_2 = A_3{:}\,H}{\vec{x}{:}\,\vec{K}\vdash A_1 = A_3{:}\,H}\ (\text{transitivity2})$$

$$\frac{\vec{x}{:}\,\vec{K}\vdash A{:}\,H \quad \vec{x}{:}\,\vec{K}\vdash H = H'{:}\,\mathbf{Type}}{\vec{x}{:}\,\vec{K}\vdash A{:}\,H'}\ (\text{replacement1})$$

$$\frac{\vec{x}{:}\,\vec{K}\vdash A = A'{:}\,H \quad \vec{x}{:}\,\vec{K}\vdash H = H'{:}\,\mathbf{Type}}{\vec{x}{:}\,\vec{K}\vdash A = A'{:}\,H'}\ (\text{replacement2})$$

$$\frac{\begin{pmatrix} \vec{x}{:}\,\vec{K}\vdash H = H'{:}\,\mathbf{Type} \\ \vec{y}{:}\,\vec{G}\vdash A_i = A_i'{:}\,K_i[x_1,\cdots x_{i-1}/A_1,\cdots A_{i-1}]\ (1\leq i\leq n) \end{pmatrix}}{\vec{y}{:}\,\vec{G}\vdash H[\vec{x}/\vec{A}] = H'[\vec{x}/\vec{A'}]{:}\,\mathbf{Type}}\ (\text{congruence1})$$

$$\frac{\begin{pmatrix} \vec{x}{:}\,\vec{K}\vdash B = B'{:}\,H \\ \vec{y}{:}\,\vec{G}\vdash A_i = A_i'{:}\,K_i[x_1,\cdots x_{i-1}/A_1,\cdots A_{i-1}]\ (1\leq i\leq n) \end{pmatrix}}{\vec{y}{:}\,\vec{G}\vdash B[\vec{x}/\vec{A}] = B'[\vec{x}/\vec{A'}]{:}\,H[\vec{x}/\vec{A}]}\ (\text{congruence2})$$

In the (context1) rule, () represents the empty context. The only starting rule of this inductive definition is the judgement () **Context**. (context2) rules out contexts such as $x_1{:}\,K_1(x_2), x_2{:}\,K_2(x_1)$.

(constant1) and (constant2) mean that introductions of constant type or term symbols are justified only when the associated structural judgements have well-formed contexts. (axiom1) and (axiom2) show that introductions of axioms are justified only when the expressions involved are well-formed. For these clauses, concluded judgements are called *well-formed*.

In (weakening1) and (weakening2), the added variable x_{n+1} is counted as a free variable of H or A which occurs vacuously, for technical convenience.

Through (replacement1) and (replacement2), typing and well-formedness of expressions are entangled with equality of types, which, in turn, is determined by equality of terms. This is the main source of extra complication in passing from the usual algebraic case to the generalised algebraic case.

We add the following constants and equalities as inference rules because we need these to relate $\mathcal{TF}$ with $\mathcal{GA}$. These rules are guaranteed to be well-formed in any $\mathcal{GA}$ theory.

Definition 5.2.4. (Continued.)

The following are inference rules of a pretheory in $\mathcal{G}A$.

$$\frac{}{()\vdash \mathbf{1}\colon \mathbf{Type}}\ (\mathbf{1}\mathrm{F})$$

$$\frac{}{()\vdash *\colon \mathbf{1}}\ (\mathbf{1}\mathrm{I})$$

$$\frac{\vec{x}\colon \vec{K}\vdash A\colon \mathbf{1}}{\vec{x}\colon \vec{K}\vdash A = *\colon \mathbf{1}}\ (\mathbf{1}\mathrm{E})$$

Definition 5.2.5. A $\mathcal{G}A$ *theory* T is a pretheory whose axioms and judgements $J^T_{\mathrm{F}}, J^T_{\mathrm{C}}$ associated to constant type or term symbols are all well-formed.

It should be remarked that a large, simultaneous, recursive definition of well-formed expressions and derivability of judgements will do the same task. But such a definition of a theory causes some problems of exposition. For example, we must introduce notions concerned with a language of a theory in the form of definition-proposition, proving that the definition makes sense.

Equality of types and terms is always determined under a specified context. This treatment of equality, *stratification*, is already required in many-sorted equational theories, see [25]. However, we omit contexts of equations whenever they are clearly understood from the "context" (in the usual sense). We also make implicit use of (transitivity), (congruence), etc. writing equations in usual style.

Phrases of the form "a term $\vec{x}\colon \vec{K}\vdash A\colon H$ ..." mean "a term A, which is of a type H and well-formed under a context $\vec{x}\colon \vec{K}$, ..." in the following.

5.2.2 Standard interpretations of $\mathcal{G}A$ theories

Now we informally introduce the standard interpretation of dependent types in the families of sets indexed by sets to give an insight into how locally small fibrations models type dependency later. First, we fix some notation.

Definition 5.2.6. Given I-indexed family $X = (X_i)_{i\in I}$ of sets, a map $u\colon J \to I$, and an J-indexed family $(f_j)_{j\in J}$ with $f_j \in X_{uj}$,

- $X[u]$ denotes the *re-indexing* of X *along* u, that is, the family $(X_{u(j)})_{j\in J}$ indexed by J.
- χ_X denotes a $\coprod_{i\in I} X_i$-indexed family of elements $(e)_{(i,e)\in\coprod_{i\in I} X_i}$.
- $d_X\colon \coprod_{i\in I} X_i \longrightarrow I$ is the map sending (i,e) to i.
- $\prec u, f\succ\colon J \longrightarrow \coprod_{i\in I} X_i$ is the map sending j to (uj, f_j).

The operators χ_{-}, $\prec -, -\succ$, and d_{-} describe the local smallness of the fibration $p_{\mathcal{C}}\colon \mathbf{Fam}(\mathbf{Set}) \longrightarrow \mathbf{Set}$.

We write $\overline{E}$ for the informal interpretation of $\mathcal{G}A$ expression E into the families of sets indexed by sets. A closed type $\vdash A\colon \mathbf{Type}$ (a type without free variables) and a closed term $\vdash A\colon K$ are interpreted as a set $\overline{K}$ and its element $\overline{A} \in \overline{K}$. Consider an open type $x\colon K\vdash L(x)\colon \mathbf{Type}$. For each closed term $\vdash A\colon K$, (subst1) rule gives a closed type $\vdash L(A)\colon \mathbf{Type}$, so an open type $x\colon K\vdash L(x)\colon \mathbf{Type}$ defines a family of closed types indexed by the set of closed terms of type K. Then the interpretation $\overline{L(x)}$ is naturally given by a family of sets $(\overline{L}_k)_{k\in\overline{K}}$ indexed by $\overline{K}$ with $\overline{L}_{\overline{A}} = \overline{L(A)}$. Similarly an open term $x\colon K\vdash B(x)\colon L(x)$ is interpreted as a family of elements $(\overline{B}_k)_{k\in\overline{K}}$, where $\overline{B}_k \in \overline{L}_k$ for all $k \in \overline{K}$.

A type with two or more free variables, say $x\colon K, y\colon L(x)\vdash H(x,y)\colon \mathbf{Type}$, is also considered as a family of sets that are interpretations of closed types, but the indexing set is no longer a mere interpretation of a closed type. Instead, one has closed type $H(A,B)$ for each pair of closed terms $\vdash A\colon K$ and $\vdash B\colon L(A)$, i.e., terms consistently assignable to $x\colon K, y\colon L(x)$. Interpreting such pairs, one gets the set $\coprod_{k\in\overline{K}} \overline{L}_k = \{(k,l) | k \in \overline{K}, l \in \overline{L}_k\}$, which is the interpretation $\overline{x\colon K, y\colon L(x)}$ of the context. The interpretation $\overline{H(x,y)}$ is then a family $(\overline{H}_{(k,l)})_{(k,l)\in\overline{x:K,y:L(x)}}$ indexed by this set.

The definition of $\bar{E}$ for a well-formed expression E follows from the the derivation of E. In the following, K_{n+1}, A etc. represent the same expressions occurring in corresponding rules in Definition 5.2.4.

The empty context given by (context1) is interpreted as a singleton set $\overline{()} = \{*\}$.

As for (context2), suppose a sets $\Gamma = \overline{\vec{x}{:}\vec{K}}$ and a family of set $\overline{K_{n+1}} = (\overline{K_{n+1}}\ _{\gamma})_{\gamma\in\Gamma}$ are already defined. $\overline{\vec{x}{:}\vec{K}, x_{n+1}{:}K_{n+1}}$ is defined to be $\coprod_{\gamma\in\Gamma} \overline{K_{n+1}}\ _{\gamma}(= \{(\gamma,k)|\gamma\in\Gamma\ k\in\overline{K}\})$.

$\mathbf{1}$ and $*$ introduced by ($\mathbf{1}$ F) and ($\mathbf{1}$ I) are interpreted by $\overline{\mathbf{1}} = \{*\}$ and $\overline{*} = *$, respectively. For other constants introduced by (constant1) and (constant2), $\overline{F(\vec{x})}$ is a family of sets $(\overline{\mathrm{F}}_\gamma)_{\gamma\in\overline{\vec{x}:\vec{K}}}$ and $\overline{C(\vec{x})}$ a family of elements $(\overline{\mathrm{C}}_\gamma)_{\gamma\in\overline{\vec{x}:\vec{K}}}$ with $\overline{\mathrm{C}}_\gamma \in \overline{H}_\gamma$ so that the axioms are satisfied. It is impossible to define interpretations of all the constants *first* and then check satisfaction of axioms because well-definedness of interpretation depends on satisfaction of equations as remarked in Section 5.2.1.

The interpretation of a variable introduced by (assumption) is $\overline{x_{n+1}} = \chi_{\overline{K_{n+1}}}$.

As for (weakening1), one must distinguish the interpretation of the type H occurring in $\vec{x}{:}\vec{K}\vdash H{:}\,\mathbf{Type}$ from one occurring in $(\vec{x}{:}\vec{K}, x_{n+1}{:}K_{n+1})\vdash H{:}\,\mathbf{Type}$ because they should have different indexing sets. The interpretation of the latter H is given by that of the former, that is, $\overline{H}^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} = \overline{H}^{\vec{x}:\vec{K}}[d_{\overline{K_{n+1}}}]$. Rule (weakening2) is treated similarly. In particular, a variable introduced by successive application of (weakening2) is interpreted by $\overline{x_i}^{\vec{x}:\vec{K}} = \chi_{\overline{K_i}}[d_{K_{i+1}}d_{K_{i+2}}\cdots d_{K_n}]$.

Before giving the interpretation of $\mathrm{F}(\vec{A})$ in (subst1), we show the simplest case. Suppose $w{:}\,G\vdash A[w]{:}\,K$ and $x{:}\,K\vdash \mathrm{F}(x){:}\,\mathbf{Type}$. The type $w{:}\,G\vdash \mathrm{F}(A[w]){:}\,\mathbf{Type}$ is then naturally interpreted by the family of sets $(\overline{\mathrm{F}}_{\overline{A}g})_{g\in\overline{G}}$ indexed by $\overline{G}$. Identifying $\overline{(w{:}\,G)} = \coprod_{*\in\mathbf{1}} \overline{G}$ with $\overline{G}$, one can write $\overline{\mathrm{F}(A)} = \overline{\mathrm{F}}[\prec !, A\succ]$ where ! is the unique map from $\overline{G}$ to a singleton set. For the general case, given

$$\vec{y}{:}\,\vec{G}\vdash A_i{:}\,K_i[x_1,\cdots x_{i-1}/A_1,\cdots A_{i-1}] \quad (1\le i\le n),$$

one obtains a map $\prec\prec\prec\cdots\prec\prec !, \overline{A}_1\succ, \overline{A}_2\succ,\cdots\succ, \overline{A}_{n-1}\succ, \overline{A}_n\succ{:}\,\overline{\vec{y}{:}\,\vec{G}} \longrightarrow \overline{\vec{x}{:}\,\vec{K}}$, sending $(g_1,\cdots,g_m)$ to $(\overline{A_1})_{(g_1,\cdots,g_m)},\cdots,(\overline{A_n})_{(g_1,\cdots,g_m)}$. Then, for constant type $\vec{x}{:}\,\vec{K}\vdash \mathrm{F}(\vec{x}){:}\,\mathbf{Type}$, we put

$$\overline{\mathrm{F}(\vec{A})} = \overline{\mathrm{F}}[\prec\prec\prec\cdots\prec\prec !, \overline{A}_1\succ, \overline{A}_2\succ,\cdots\succ, \overline{A}_{n-1}\succ, \overline{A}_n\succ].$$

The interpretation of $\mathrm{C}(\vec{A})$ in (subst2) is defined similarly.

5.2.3 Dependent product and coproduct types

The type system $\mathcal{G}A$ is strengthened with new kinds of type forming operations. This subsection introduces three of these, *dependent product type* and two kinds of *dependent coproduct type* formations.

Type system λP1

λP1 is a type system obtained from $\mathcal{G}A$ by adjoining *dependent product types*. Intuitively, a dependent product type is a type of functions whose result types depend on their arguments. In terms of Jacobs' classification, this adds to $\mathcal{G}A$ the feature $(*, *)$-products.

Definition 5.2.7. The *type expressions* and *term expressions* of λP1 are defined by

- (clauses given in Definition 5.2.1.)
- for all type expressions K and H, $\Pi x\colon K.H$ is a type expression.
- for all type expressions K and all term expression A, $\lambda x\colon K.(A)$ is a term expression.

Definition 5.2.8. The derivable judgements of a pretheory in λP1 are defined with inference rules (context1) – (congruence2) given in Definition 5.2.4 plus the following rules.

$$\frac{\vec{x}\colon\vec{K}, x_{n+1}\colon K_{n+1}\vdash H\colon \mathbf{Type}}{\vec{x}\colon\vec{K}\vdash\Pi x_{n+1}\colon K_{n+1}.H\colon\mathbf{Type}}\;(\Pi\mathrm{F})$$

$$\frac{\vec{x}\colon\vec{K}, x_{n+1}\colon K_{n+1}\vdash A\colon H}{\vec{x}\colon\vec{K}\vdash\lambda x_{n+1}\colon K_{n+1}.(A)\colon\Pi x_{n+1}\colon K_{n+1}.H}\;(\Pi\mathrm{I})$$

$$\frac{\vec{x}\colon\vec{K}\vdash B\colon\Pi x_{n+1}\colon K_{n+1}.H \quad \vec{x}\colon\vec{K}\vdash C\colon K_{n+1}}{\vec{x}\colon\vec{K}\vdash BC\colon H[x_{n+1}/C]}\;(\Pi\mathrm{E})$$

$$\frac{\vec{x}\colon\vec{K}, x_{n+1}\colon K_{n+1}\vdash A\colon H \quad \vec{x}\colon\vec{K}\vdash C\colon K_{n+1}}{\vec{x}\colon\vec{K}\vdash(\lambda x_{n+1}\colon K_{n+1}.(A))C = A[x_{n+1}/C]\colon H[x_{n+1}/C]}\;(\beta)$$

$$\frac{\vec{x}\colon\vec{K}\vdash B\colon\Pi x_{n+1}\colon K_{n+1}.H}{\vec{x}\colon\vec{K}\vdash\lambda z\colon K_{n+1}.(Bz) = B\colon\Pi x_{n+1}\colon K_{n+1}.H}\;(\eta)$$

$$\frac{\vec{x}\colon \vec{K}\vdash K_{n+1} = K'\colon \mathbf{Type} \quad \vec{x}\colon \vec{K}, x'\colon K'\vdash H' = H[x_{n+1}/x']\colon \mathbf{Type}}{\vec{x}\colon \vec{K}\vdash \Pi x_{n+1}\colon K_{n+1}.H = \Pi x\colon K'.H'\colon \mathbf{Type}} \text{(congruence1)}$$

$$\frac{\begin{pmatrix} \vec{x}\colon \vec{K}\vdash K_{n+1} = K'\colon \mathbf{Type} \\ \vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1}\vdash H\colon \mathbf{Type} \\ \vec{x}\colon \vec{K}, x'\colon K'\vdash A' = A[x_{n+1}/x']\colon H[x_{n+1}/x'] \end{pmatrix}}{\vec{x}\colon \vec{K}\vdash \lambda x_{n+1}\colon K_{n+1}.(A) = \lambda x'\colon K'.(A')\colon \Pi x_{n+1}\colon K_{n+1}.H} \text{(congruence2)}$$

The occurrences of x_{n+1} in the conclusion of (ΠF) and (ΠI) are called *bound* as usual. Instead of complicating the rules with tedious side conditions, we presuppose, at a meta-level, the usual variable conventions such as no variables having bound and free occurrences at the same time and all bound variables being distinct.

If H occurring in a type $\Pi x\colon K.H$ contains no free occurrence of x, the type is written $K \supset H$ because its intuitive meaning agrees with the usual function type from K to H.

Type system $w\mathcal{M}L$

Dependent coproduct types are similar to disjoint sums of sets. A term of a dependent coproduct types is a pair of terms whose second component has varying type depending on its first component. We first add to $\mathcal{T}F\lambda$ dependent coproduct types, yielding the type system $w\mathcal{M}L$. In Jacobs' terms, this is weak $(*,*)$-sum. The added inference rules are as follows.

$$\frac{\vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1}\vdash G\colon \mathbf{Type}}{\vec{x}\colon \vec{K}\vdash \Sigma x_{n+1}\colon K_{n+1}.G\colon \mathbf{Type}} (w\Sigma\text{F})$$

$$\frac{\vec{x}\colon \vec{K}\vdash A_1\colon K_{n+1} \quad \vec{x}\colon \vec{K}\vdash A_2\colon G[x_{n+1}/A_1]}{\vec{x}\colon \vec{K}\vdash \langle A_1, A_2\rangle\colon \Sigma x_{n+1}\colon K_{n+1}.G} (w\Sigma\text{I})$$

$$\frac{\vec{x}\colon \vec{K}\vdash A\colon \Sigma x_{n+1}\colon K_{n+1}.G \quad \vec{x}\colon \vec{K}\vdash H\colon \mathbf{Type} \quad \vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1}, y\colon G\vdash B\colon H}{\vec{x}\colon \vec{K}\vdash \mathbf{let}_{\langle x_{n+1}, y\rangle = A} \mathbf{\ in\ } B\colon H} (w\Sigma\text{E})$$

$$\frac{\begin{pmatrix} \vec{x}\colon \vec{K}\vdash A_1\colon K_{n+1} \quad \vec{x}\colon \vec{K}\vdash A_2\colon G[x_{n+1}/A_1] \\ \vec{x}\colon \vec{K}\vdash H\colon \mathbf{Type} \quad \vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1}, y\colon G\vdash B\colon H \end{pmatrix}}{\vec{x}\colon \vec{K}\vdash \mathbf{let}_{\langle x,y\rangle = \langle A_1, A_2\rangle} \mathbf{\ in\ } B = B[x,y/A_1,A_2]\colon H} (\beta')$$

$$\frac{\vec{x}\colon \vec{K}\vdash A\colon \Sigma x_{n+1}\colon K_{n+1}.G \quad \vec{x}\colon \vec{K}, z\colon \Sigma x_{n+1}\colon K_{n+1}.G\vdash C\colon H}{\vec{x}\colon \vec{K}\vdash \mathbf{let}_{\langle x,y\rangle = A} \mathbf{\ in\ } C[z/\langle x,y\rangle] = C[z/A]\colon H} (\eta')$$

Congruence rules similar to (congruence1,2) in Definition 5.2.8 is omitted.

With this weaker notion of dependent coproduct, one cannot extract the second component A_2 of $\langle A_1, A_2\rangle\colon \Sigma x_{n+1}\colon K_{n+1}.G$, since, in ($w\Sigma$E) rule, the type H of B can not depend on $x_{n+1}\colon K_{n+1}$. This restriction corresponds to a familiar restriction on eigen variables in the elimination rule for $\exists$ of the first-order logic.

$$\dfrac{\begin{array}{c}\vdots\\ \exists x_{n+1}.G[x_{n+1}]\end{array} \qquad \begin{array}{c}[G]\\ \vdots\, B\\ H\end{array}}{H}$$

In the above natural deduction style rule, the proof B must not contain open assumptions containing x_{n+1} freely. Categorically, this restriction gives the pattern of adjointness.

Type system $\mathcal{M}L$

The type system $\mathcal{M}L$ has usual dependent coproduct types with first and second projections. In Jacobs' terms, they are strong $(*,*)$-sums. The inference rules added to those of λP1 are:

$$\dfrac{\vec{x}\colon \vec{K}, x_{n+1}\colon K_{n+1} \vdash G\colon \mathbf{Type}}{\vec{x}\colon \vec{K} \vdash \Sigma x_{n+1}\colon K_{n+1}.G\colon \mathbf{Type}}\ (\Sigma\mathrm{F})$$

$$\dfrac{\vec{x}\colon \vec{K} \vdash A_1\colon K_{n+1} \quad \vec{x}\colon \vec{K} \vdash A_2\colon G[x_{n+1}/A_1]}{\vec{x}\colon \vec{K} \vdash \langle A_1, A_2\rangle\colon \Sigma x_{n+1}\colon K_{n+1}.G}\ (\Sigma\mathrm{I})$$

$$\dfrac{\vec{x}\colon \vec{K} \vdash A\colon \Sigma x_{n+1}\colon K_{n+1}.G}{\vec{x}\colon \vec{K} \vdash \mathbf{fst}(A)\colon K_{n+1}}\ (\Sigma\mathrm{E1}) \qquad \dfrac{\vec{x}\colon \vec{K} \vdash A\colon \Sigma x_{n+1}\colon K_{n+1}.G}{\vec{x}\colon \vec{K} \vdash \mathbf{snd}(A)\colon G[x_{n+1}/\mathbf{fst}(A)]}\ (\Sigma\mathrm{E2})$$

$$\dfrac{\vec{x}\colon \vec{K} \vdash A_1\colon K_{n+1} \quad \vec{x}\colon \vec{K} \vdash A_2\colon G[x_{n+1}/A_1]}{\vec{x}\colon \vec{K} \vdash \mathbf{fst}(\langle A_1, A_2\rangle) = A_1\colon K_{n+1}}\ (\mathbf{fst}(-)\mathrm{eq})$$

$$\dfrac{\vec{x}\colon \vec{K} \vdash A_1\colon K_{n+1} \quad \vec{x}\colon \vec{K} \vdash A_2\colon G[x_{n+1}/A_1]}{\vec{x}\colon \vec{K} \vdash \mathbf{snd}(\langle A_1, A_2\rangle) = A_2\colon G[x_{n+1}/A_1]}\ (\mathbf{snd}(-)\mathrm{eq})$$

$$\dfrac{\vec{x}\colon \vec{K} \vdash A\colon \Sigma x_{n+1}\colon K_{n+1}.G}{\vec{x}\colon \vec{K} \vdash \langle \mathbf{fst}(A), \mathbf{snd}(A)\rangle = A\colon \Sigma x_{n+1}\colon K_{n+1}.G}\ (\langle\rangle\mathrm{eq})$$

One can define the **let** construct of $w\mathcal{M}L$ by

$$\mathbf{let}_{\langle x_{n+1},y\rangle = A}\ \mathbf{in}\ B \equiv B[x_{n+1}/\mathbf{fst}(A), y/\mathbf{snd}(A)],$$

then (β') and (η') rules of $w\mathcal{ML}$ are derivable in $\mathcal{ML}$.

5.2.4 Interpretations between theories

We define a notion of *interpretations* between theories. An interpretation is a map that sends well-formed expressions of one theory to those of the other, preserving equality. Through an interpretation, derivations in one theory can be mapped to those in another. Later, we will form categories of theories with interpretations as morphisms to show the equivalence of those to categories of suitable models.

Definition 5.2.9. Given two $\mathcal{G}A$ theories T_G and T'_G, a map $I(-)$ from the set of well-formed expressions paired with their contexts of T_G to the set of possibly not well-formed expressions of T'_G is a *preinterpretation* from T_G to T'_G if and only if the following conditions are satisfied. The conditions are listed following structural inference rules for expressions of T_G with the same notation in Definition 5.2.4. For brevity, the set of variable symbols of T'_G is supposed to be the same as that of T_G.

$$\begin{array}{ll} \text{(context1)} & I((\ )) = (\ ) \\ \text{(context2)} & I(\vec{x}{:}\,\vec{K}, x_{n+1}{:}\,K_{n+1}) = (I(\vec{x}{:}\,\vec{K}), x_{n+1}{:}\,I(K_{n+1})) \\ \text{(assumption)} & I(x_{n+1})^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} = x_{n+1} \\ \text{(weakening1)} & I(H)^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} = I(H)^{\vec{x}:\vec{K}} \\ \text{(weakening2)} & I(A)^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} = I(A)^{\vec{x}:\vec{K}} \\ \text{(subst1)} & I(\mathrm{F}(\vec{A}))^{\vec{y}:\vec{G}} = I(\mathrm{F}(\vec{x}))^{\vec{x}:\vec{K}}[\vec{x}/I(A_1)^{\vec{y}:\vec{G}}, \cdots, I(A_n)^{\vec{y}:\vec{G}}] \\ \text{(subst2)} & I(\mathrm{C}(\vec{A}))^{\vec{y}:\vec{G}} = I(\mathrm{C}(\vec{x}))^{\vec{x}:\vec{K}}[\vec{x}/I(A_1)^{\vec{y}:\vec{G}}, \cdots, I(A_n)^{\vec{y}:\vec{G}}] \end{array}$$

An *interpretation* is a pretranslation satisfying the following conditions.

$$\begin{array}{ll} \text{(constant1)} & I(\vec{x}{:}\,\vec{K}) \vdash I(\mathrm{F}(\vec{x}))^{\vec{x}:\vec{K}}{:}\,\mathbf{Type} \quad \text{is derivable in } T'_G \\ \text{(constant2)} & I(\vec{x}{:}\,\vec{K}) \vdash I(\mathrm{C}(\vec{x}))^{\vec{x}:\vec{K}}{:}\,I(H)^{\vec{x}:\vec{K}} \quad \text{is derivable in } T'_G \\ \text{(axiom1),(axiom2)} & \text{Axioms of } T_F \text{ are sent to derivable judgements of } T'_G \end{array}$$

A preinterpretation $I(-)$ is determined by $I(\mathrm{F}(\vec{x}))$'s and $I(\mathrm{C}(\vec{x}))$'s under the above conditions. One can show the soundness of interpretations from T_G to T'_G by induction on derivations in T_G.

Lemma 5.2.10. Given an interpretation $I(-)$ from a $\mathcal{G}A$ theory T_G to T'_G, interpretations of the derived judgements of T_G are also derivable in T'_G. ∎

Definition 5.2.11. Given two λP1 theories T_G to T'_G, a *(pre)interpretation* from T_G and T'_G is a map satisfying the conditions given in Definition 5.2.9 with the conditions below.

$$\begin{array}{rl}
(\Pi\mathrm{F}) & I(\Pi x_{n+1}\colon K_{n+1}.H)^{\vec{x}:\vec{K}} = \Pi z\colon I(K_{n+1})^{\vec{x}:\vec{K}}.I(H)^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}[x_{n+1}/z] \\
(\Pi\mathrm{I}) & I(\lambda x_{n+1}\colon K_{n+1}.(A))^{\vec{x}:\vec{K}} = \lambda z\colon I(K_{n+1})^{\vec{x}:\vec{K}}.(I(A)^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}[x_{n+1}/z]) \\
(\Pi\mathrm{E}) & I(BC)^{\vec{x}:\vec{K}} = I(B)^{\vec{x}:\vec{K}} I(C)^{\vec{x}:\vec{K}}
\end{array}$$

where z is a fresh variable.

Definition 5.2.12. Given two $w\mathcal{M}L$ theories T_w and T'_w, a *(pre)interpretation* from T_w and T'_w is a map satisfying the conditions given in Definition 5.2.9 and 5.2.11 plus the following conditions.

$$\begin{array}{rl}
(w\Sigma\mathrm{F}) & I(\Sigma x_{n+1}\colon K_{n+1}.H)^{\vec{x}:\vec{K}} = \Sigma z\colon I(K_{n+1})^{\vec{x}:\vec{K}}.I(H)^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}[x_{n+1}/z] \\
(w\Sigma\mathrm{I}) & I(\langle A_1, A_2\rangle)^{\vec{x}:\vec{K}} = \langle I(A_1)^{\vec{x}:\vec{K}}, I(A_2)^{\vec{x}:\vec{K}}\rangle \\
(w\Sigma\mathrm{E}) & I(\mathbf{let}_{\langle x_{n+1},y\rangle = A}\ \mathbf{in}\ B)^{\vec{x}:\vec{K}} \\
& \quad = \mathbf{let}_{\langle z,w\rangle = I(A)^{\vec{x}:\vec{K}}}\ \mathbf{in}\ I(B)^{\vec{x}:\vec{K},x:K_{n+1},y:G}[x_{n+1}/z, y/w]
\end{array}$$

(z, w are fresh variables.)

Definition 5.2.13. Given two $\mathcal{M}L$ theories T_M and T'_M, a *(pre)interpretation* from T_M to T'_M is a map satisfying the conditions given in Definition 5.2.9 and

5.2.11 plus the following conditions.

$$
\begin{array}{rl}
(\Sigma\mathrm{F}) & I(\Sigma x_{n+1}{:}\, K_{n+1}.H)^{\vec{x}:\vec{K}} = \Sigma z{:}\, I(K_{n+1})^{\vec{x}:\vec{K}}.I(H)^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}[x_{n+1}/z] \\
(\Sigma\mathrm{I}) & I(\langle A_1, A_2\rangle)^{\vec{x}:\vec{K}} = \langle I(A_1)^{\vec{x}:\vec{K}}, I(A_2)^{\vec{x}:\vec{K}}\rangle \\
(\Sigma\mathrm{E1}) & I(\mathbf{fst}(A))^{\vec{x}:\vec{K}} = \mathbf{fst}(I(A)^{\vec{x}:\vec{K}}) \\
(\Sigma\mathrm{E2}) & I(\mathbf{snd}(A))^{\vec{x}:\vec{K}} = \mathbf{snd}(I(A)^{\vec{x}:\vec{K}}) \\
& \quad\text{where } z \text{ is a fresh variable.}
\end{array}
$$

In each case, the notion *interpretations* is defined similarly to Definition 5.2.9. The soundness of these interpretations is also shown by similar induction.

5.3 Type system $\mathcal{TF}$

This section defines the type system $\mathcal{TF}$ and gives intuition behind its inference rules. $\mathcal{TF}$ is a syntactical and equational presentation of fibrewise locally small fibrations with a terminal object, product and coproducts along Hom-projections.

The characteristics of $\mathcal{TF}$ are as follows. First, it has a generalised notion of terms such that a term must be specified with a domain type and a codomain type. This notion of *generalised term* allows $\mathcal{TF}$ to have more general structure than usual type systems. Direct connections with a greater variety of models and the duality between dependent product and coproduct types are also obtained with this notion.

Second, variable names are abandoned in the syntax of $\mathcal{TF}$. Substitution is addressed by *re-indexing* operations along *context mappings* and *generic term* that express indeterminate values. The resulting system is similar to de Bruijn notation.

A fibration $p{:}\,\mathcal{E} \longrightarrow \mathcal{B}$ expressed in $\mathcal{TF}$ has a splitting and satisfies the strict Beck-Chevalley condition, every base object is isomorphic to one of the form $\mathrm{Hom}_I(1, X)$ for some $I \in \mathcal{B}$ and $X \in \mathcal{E}_I$. The strictness condition excludes many natural examples, but this makes it feasible to set up notation and show precise calculations, which are already complex even without coherent isomorphisms.

However, this involves no substantial loss of generality since, given a fibration p, one can construct a split fibration equivalent to p in the 2-category of fibrations (see Proposition 4.5.6).

5.3.1 $\mathcal{T}F$ theories

Expressions of a $\mathcal{T}F$ theory are classified in four kinds: *contexts* $\Gamma, \Delta, \cdots$, *context mappings* $u, v, \cdots$, *types* $K, L, \cdots$, *generalised terms* $A, B, \cdots$. Each of these corresponds to the form of *structural judgements* that assert well-formedness and typing of expressions. Context mappings, types, and generalised terms of a theory in $\mathcal{T}F$ have a notion of equality asserted with *equality judgements*.

Definition 5.3.1. *Judgements* are expressions of the following forms.

- (Structural judgements for contexts.) Γ **Context** (read "Γ is a well-formed context.")
- (Structural judgements for context mappings.) $u\colon \Gamma \to \Delta$ ("u is a well-formed context mapping from a context Γ to Δ.")
- (Structural judgements for types.) $\Gamma \vdash K\colon \mathbf{Type}$ ("K is a well-formed type under a context Γ.")
- (Structural judgements for terms.) $\Gamma \vdash A\colon K \Rightarrow L$ ("A is a well-formed term under a context Γ from a domain type K to a codomain type L.")
- (Equality judgements for context mappings.) $u = u'\colon \Delta \to \Gamma$
- (Equality judgements for types.) $\Gamma \vdash K = K'\colon \mathbf{Type}$
- (Equality judgements for terms.) $\Gamma \vdash A = A'\colon K \Rightarrow L$

The definition of $\mathcal{T}F$ theories exhibits the same difficulty as the one that appeared in Section 5.2.1. The set of expressions, well-formedness and typing of expressions, and well-formedness of introduction rules and axioms are all defined by a simultaneous recursive definition of the derivability of judgements. In contrast to Section 2, we give a large simultaneous definition to which the rest of this section is devoted. However, we will swap between the two possible presentations when it is convenient. For example, we refer to the set of possibly not well-formed expressions without explicit definition, because it should be easily understood.

Definition 5.3.2. Given the data listed below,

- a set of constant type symbols.
- a set of constant term symbols.
- for each constant type symbol F, an associated structural judgement J^T_{F} of the form $\Gamma\vdash\mathrm{F}\colon\mathbf{Type}$ for some Γ.
- for each constant term symbol C, an associated structural judgement J^T_{C} of the form $\Gamma\vdash\mathrm{C}\colon K\Rightarrow L$ for some Γ and K, L.
- a set Ax_T of *axioms* consisting of equality judgements for types or terms.

A $\mathcal{TF}$ *theory* T consists of a class of *derivable* (or *well-formed*) judgements closed under the inference rules given in the rest of this section. The *well-formed expressions* of T are ones subject to derivable judgements of T. Moreover, all the given structural judgements associated with constant symbols and all the axioms of T must be well-formed in T.

In the following, we motivate inference rules by informally showing the correspondence between a type theory T and a fibrewise locally small fibration $p\colon\mathcal{E}\longrightarrow\mathcal{B}$ with a terminal object, products and coproducts along Hom-projections.

Contexts and context mappings

Definition 5.3.2. (Continued.) The following form part of the inference rules of T.

$$\frac{}{()\ \mathbf{Context}}\ (\mathrm{C1}) \qquad \frac{\Gamma\vdash K, L\colon\mathbf{Type}}{(\Gamma, K\Rightarrow L)\ \mathbf{Context}}\ (\mathrm{C2})$$

$$\frac{}{\mathbf{id}_\Gamma\colon\Gamma\to\Gamma}\ (\mathrm{PM1}) \qquad \frac{u\colon\Delta\to\Gamma \quad v\colon\Xi\to\Delta}{u\circ v\colon\Xi\to\Gamma}\ (\mathrm{CM2})$$

$$\frac{u\colon\Delta\to\Gamma}{\mathbf{id}_\Gamma\circ u = u\colon\Delta\to\Gamma}\ (\mathrm{ECM1}) \qquad \frac{u\colon\Delta\to\Gamma}{u\circ\mathbf{id}_\Delta = u\colon\Delta\to\Gamma}\ (\mathrm{ECM2})$$

$$\frac{u\colon\Delta\to\Gamma \quad v\colon\Xi\to\Delta \quad w\colon\Theta\to\Xi}{(u\circ v)\circ w = u\circ(v\circ w)\colon\Theta\to\Gamma}\ (\mathrm{ECM3})$$

$$\frac{\Gamma\ \mathbf{Context}}{!_\Gamma\colon \Gamma \to ()}\ (\text{CM3}) \qquad \frac{w\colon \Gamma \to ()}{w = !_\Gamma\colon \Gamma \to ()}\ (\text{ECM4})$$

$$\frac{\Gamma \vdash K, L\colon \mathbf{Type}}{\mathbf{d}_{K \Rightarrow L}\colon (\Gamma, K \Rightarrow L) \to \Gamma}\ (\text{CM4})$$

$$\frac{\Gamma \vdash K, L\colon \mathbf{Type} \quad u\colon \Delta \to \Gamma \quad \Delta \vdash A\colon K[u] \Rightarrow L[u]}{\prec u, A \succ\colon \Delta \to (\Gamma, K \Rightarrow L)}\ (\text{CM5})$$

$$\frac{\Gamma \vdash K, L\colon \mathbf{Type} \quad u\colon \Delta \to \Gamma \quad \Delta \vdash A\colon K[u] \Rightarrow L[u]}{\mathbf{d}_{K \Rightarrow L} \circ \prec u, A \succ = u\colon \Delta \to \Gamma}\ (\text{ECM5})$$

$$\frac{\Gamma \vdash K, L\colon \mathbf{Type} \quad w\colon \Delta \to (\Gamma, K \Rightarrow L)}{\prec \mathbf{d}_{K \Rightarrow L} \circ w, \chi_{K \Rightarrow L}[w] \succ = w\colon \Delta \to (\Gamma, K \Rightarrow L)}\ (\text{ECM6})$$

Subscripts Γ in $!_\Gamma$ and $K \Rightarrow L$ in $\mathbf{d}_{K \Rightarrow L}$ are occasionally omitted if they are recoverable from the context.

The intuition behind these inference rules is given by the following correspondence between T and p. Contexts correspond to objects of the base category $\mathcal{B}$ and context mappings to morphisms of $\mathcal{B}$.

	T	$p\colon \mathcal{E} \longrightarrow \mathcal{B}$
(C1)	$()$	$1 \in \mathcal{B}$ (the terminal of $\mathcal{B}$)
(C2)	$(\Gamma, K \Rightarrow L)$	$\mathrm{Hom}_\Gamma(K, L) \in \mathcal{B}$
(CM1)	$\mathbf{id}_\Gamma$	$id_\Gamma\colon \Gamma \to \Gamma \in \mathcal{B}$
(CM2)	$u \circ v$	$uv\colon \Delta \to \Xi \in \mathcal{B}$
(CM3)	$!_\Gamma$	$!_\Gamma\colon \Gamma \to 1 \in \mathcal{B}$ (the unique projection)
(CM4)	$\mathbf{d}_{K \Rightarrow L}$	$d_{\Gamma,KL}\colon \mathrm{Hom}_\Gamma(K, L) \to \Gamma \in \mathcal{B}$
(CM5)	$\prec u, A \succ$	$\prec u, A \succ\colon \Delta \to \mathrm{Hom}_\Gamma(K, L) \in \mathcal{B}$

where types and terms are identified with appropriate objects and morphisms of $\mathcal{F}$, as we shall see below.

(ECM5), (ECM6), and (ETM9) given below are the conditions that characterise $(\Gamma, K \Rightarrow L)$ as the object $\mathrm{Hom}_\Gamma(K, L)$ with $\mathbf{d}_{K \Rightarrow L}$ as $d_{\Gamma,KL}$.

Types

Definition 5.3.2. (Continued.) The following are inference rules of T.

$$\frac{}{()\vdash \mathbf{1}\colon \mathbf{Type}}\ (\mathrm{TY1}) \qquad \frac{\Gamma\,\mathbf{Context}}{\Gamma\vdash \mathrm{F}\colon \mathbf{Type}}\ (\mathrm{TY2})$$

for each J^T_{F} of the form $\Gamma\vdash\mathrm{F}\colon\mathbf{Type}$)

$$\frac{u\colon\Delta\to\Gamma \quad \Gamma\vdash K\colon\mathbf{Type}}{\Delta\vdash K[u]\colon\mathbf{Type}}\ (\mathrm{TY3})$$

$$\frac{\Gamma\ \mathbf{Context}}{\Gamma\vdash K[\mathbf{id}_\Gamma]=K\colon\mathbf{Type}}\ (\mathrm{ETY1})$$

$$\frac{\Gamma\vdash K\colon\mathbf{Type} \quad u\colon\Delta\to\Gamma \quad v\colon\Xi\to\Delta}{\Xi\vdash K[u][v]=K[u\circ v]\colon\mathbf{Type}}\ (\mathrm{ETY2})$$

$$\frac{(\Gamma, K\Rightarrow L)\vdash H\colon\mathbf{Type}}{\Gamma\vdash\Pi_{K\Rightarrow L}.H\colon\mathbf{Type}}\ (\mathrm{TY4}) \qquad \frac{(\Gamma, K\Rightarrow L)\vdash G\colon\mathbf{Type}}{\Gamma\vdash\Sigma_{K\Rightarrow L}.G\colon\mathbf{Type}}\ (\mathrm{TY5})$$

$$\frac{\Gamma\vdash K, L\colon\mathbf{Type} \quad (\Gamma, K\Rightarrow L)\vdash H\colon\mathbf{Type} \quad u\colon\Delta\to\Gamma}{\Delta\vdash(\Pi_{K\Rightarrow L}.H)[u]=\Pi_{K[u]\Rightarrow L[u]}.H[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]},\chi_{K[u]\Rightarrow L[u]}\succ]\colon\mathbf{Type}}\ (\mathrm{ETY3})$$

$$\frac{\Gamma\vdash K, L\colon\mathbf{Type} \quad (\Gamma, K\Rightarrow L)\vdash G\colon\mathbf{Type} \quad u\colon\Delta\to\Gamma}{\Delta\vdash(\Sigma_{K\Rightarrow L}.G)[u]=\Sigma_{K[u]\Rightarrow L[u]}.G[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]},\chi_{K[u]\Rightarrow L[u]}\succ]\colon\mathbf{Type}}\ (\mathrm{ETY4})$$

The rule (TY2) is an *introduction rule* for F. (Later, the associated judgement J^T_{F} will also be called introduction rule.) The type $\mathbf{1}[!_\Gamma]$ inferred by (TY3) and (CM3) is often written $\mathbf{1}$.

As for the correspondence between T and p, well-formed types under a certain context are objects of a fibre over the context regarded as an object of the base category $\mathcal{B}$. The correspondence is summarised as follows.

	T	$p\colon\mathcal{E}\longrightarrow\mathcal{B}$
(TY1)	$\mathbf{1}$	$1\in\mathcal{E}_1$ (the terminal of $\mathcal{E}_1$)
(TY3)	$K[u]$	$K[u]\in\mathcal{E}_\Delta$
(TY4)	$\Pi_{K\Rightarrow L}.H$	$\Pi_{KL}(H)\in\mathcal{E}_\Gamma$
(TY5)	$\Sigma_{K\Rightarrow L}.G$	$\Sigma_{KL}(G)\in\mathcal{E}_\Gamma$

Equations inferred by (ETY1) and (ETY2) require that p splits. Type equations (ETY3) and (ETY4) are the strict Beck-Chevalley condition for Π_{KL} and Σ_{KL} (see Definition 4.8.7). These equations show commutativity between the substitution operation and Π,Σ formation operations on types. Informally using the usual notation for substitution, we may show, for example,

$$(\Pi_{x:K(y)\Rightarrow L(y)}.H(x,y))[y/A(z)] = \Pi_{x:K(A(z))\Rightarrow L(A(z))}.H(x,A(z))$$
$$= \Pi_{x:K(y)[y/A(z)]\Rightarrow L(y)[y/A(z)]}.H(x,y)[x/x, y/A(z)].$$

In other words, a substitution $(-)[u]$ for a type $\Pi_{K\Rightarrow L}.H$ gets inside the Π formation and acts on H as $(-)[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]}, \chi_{K[u]\Rightarrow L[u]}\succ]$ which acts in the same way as $(-)[u]$ but leaves the "bound slot" unchanged.

Terms

Definition 5.3.2. (Continued.) The following are inference rules of T.

$$\frac{\Gamma\ \mathbf{Context}}{\Gamma\vdash \mathrm{C}\colon K\Rightarrow L}\ (\mathrm{TM1}) \quad (\text{for each } J^T_{\mathrm{C}} \text{ of the form } \Gamma\vdash\mathrm{C}\colon K\Rightarrow L)$$

$$\frac{\Gamma\vdash K\colon \mathbf{Type}}{\Gamma\vdash \mathbf{id}_K\colon K\Rightarrow K}\ (\mathrm{TM2}) \quad \frac{\Gamma\vdash A\colon K\Rightarrow L \quad \Gamma\vdash B\colon L\Rightarrow G}{\Gamma\vdash B\circ A\colon K\Rightarrow G}\ (\mathrm{TM3})$$

$$\frac{\Gamma\vdash A\colon K\Rightarrow L}{\mathbf{id}_L\circ A = A}\ (\mathrm{ETM1}) \quad \frac{\Gamma\vdash A\colon K\Rightarrow L}{A\circ\mathbf{id}_K = A}\ (\mathrm{ETM2})$$

$$\frac{\Gamma\vdash A\colon K\Rightarrow L \quad \Gamma\vdash B\colon L\Rightarrow G \quad \Gamma\vdash C\colon G\Rightarrow H}{\Gamma\vdash (C\circ B)\circ A = C\circ(B\circ A)\colon K\Rightarrow H}\ (\mathrm{ETM3})$$

$$\frac{\Gamma\vdash K\colon \mathbf{Type}}{\Gamma\vdash !_K\colon K\Rightarrow \mathbf{1}[!_\Gamma]}\ (\mathrm{TM4}) \quad \frac{\Gamma\vdash C\colon K\Rightarrow \mathbf{1}[!_\Gamma]}{\Gamma\vdash C = !_K\colon K\Rightarrow \mathbf{1}[!_\Gamma]}\ (\mathrm{ETM4})$$

$$\frac{u\colon\Delta\to\Gamma \quad \Gamma\vdash A\colon K\Rightarrow L}{\Delta\vdash A[u]\colon K[u]\Rightarrow L[u]}\ (\mathrm{TM5}) \quad \frac{\Gamma\vdash A\colon K\Rightarrow L}{\Gamma\vdash A[\mathbf{id}_\Gamma] = A\colon K\Rightarrow L}\ (\mathrm{ETM5})$$

$$\frac{\Gamma\vdash A\colon K\Rightarrow L \quad u\colon\Delta\to\Gamma \quad v\colon\Xi\to\Delta}{\Xi\vdash A[u][v] = A[u\circ v]\colon K[u\circ v]\Rightarrow L[u\circ v]}\ (\mathrm{ETM6})$$

$$\frac{\Gamma\vdash K\colon\mathbf{Type} \quad u\colon\Delta\to\Gamma}{\Delta\vdash(\mathbf{id}_K)[u] = \mathbf{id}_{K[u]}\colon K[u]\Rightarrow K[u]}\ (\mathrm{ETM7})$$

$$\frac{\Gamma\vdash A\colon K\Rightarrow L \quad \Gamma\vdash B\colon L\Rightarrow G \quad u\colon\Delta\to\Gamma}{\Delta\vdash B[u]\circ A[u] = (B\circ A)[u]\colon K[u]\Rightarrow G[u]}\ (\mathrm{ETM8})$$

$$\frac{\Gamma\vdash K, L\colon\mathbf{Type}}{(\Gamma, K\Rightarrow L)\vdash\chi_{K\Rightarrow L}\colon K[\mathbf{d}_{K\Rightarrow L}]\Rightarrow L[\mathbf{d}_{K\Rightarrow L}]}\ (\mathrm{TM6})$$

$$\frac{\Gamma\vdash K, L\colon\mathbf{Type} \quad u\colon\Delta\to\Gamma \quad \Delta\vdash A\colon K[u]\Rightarrow L[u]}{\Delta\vdash\chi_{K\Rightarrow L}[\prec u, A\succ] = A\colon K[u]\Rightarrow L[u]}\ (\mathrm{ETM9})$$

$$\frac{(\Gamma, K\Rightarrow L)\vdash A\colon G[\mathbf{d}_{K\Rightarrow L}]\Rightarrow H}{\Gamma\vdash\lambda_{K\Rightarrow L}.(A)\colon G\Rightarrow\Pi_{K\Rightarrow L}.H}\ (\mathrm{TM7})$$

$$\frac{(\Gamma, K\Rightarrow L)\vdash H\colon\mathbf{Type}}{(\Gamma, K\Rightarrow L)\vdash\varphi^H_{K\Rightarrow L}\colon(\Pi_{K\Rightarrow L}.H)[\mathbf{d}_{K\Rightarrow L}]\Rightarrow H}\ (\mathrm{TM8})$$

$$\frac{(\Gamma, K \Rightarrow L)\vdash A\colon G[\mathbf{d}_{K\Rightarrow L}] \Rightarrow H}{(\Gamma, K \Rightarrow L)\vdash \varphi^H_{K\Rightarrow L}\circ((\lambda_{K\Rightarrow L}.(A))[\mathbf{d}_{K\Rightarrow L}]) = A\colon G[\mathbf{d}_{K\Rightarrow L}] \Rightarrow H}\ \text{(ETM10)}$$

$$\frac{\Gamma\vdash B\colon G \Rightarrow \Pi_{K\Rightarrow L}.H}{\Gamma\vdash \lambda_{K\Rightarrow L}.(\varphi^H_{K\Rightarrow L}\circ B[\mathbf{d}_{K\Rightarrow L}]) = B\colon G \Rightarrow \Pi_{K\Rightarrow L}.H}\ \text{(ETM11)}$$

$$\frac{\Gamma\vdash G\colon \mathbf{Type} \quad (\Gamma, K \Rightarrow L)\vdash H\colon \mathbf{Type} \quad u\colon \Delta \to \Gamma}{\begin{array}{ll}(\Delta, K[u] \Rightarrow L[u])\vdash & \varphi^H_{K\Rightarrow L}[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]}, \chi_{K[u]\Rightarrow L[u]}\succ] = \\ & \varphi^{H[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]},\chi_{K[u]\Rightarrow L[u]}\succ]}_{K[u]\Rightarrow L[u]} \\ & : (\Pi_{K\Rightarrow L}.H)[u] \Rightarrow H[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]}, \chi_{K[u]\Rightarrow L[u]}\succ]\end{array}}\ \text{(ETM12)}$$

$$\frac{(\Gamma, K \Rightarrow L)\vdash A\colon G \Rightarrow H[\mathbf{d}_{K\Rightarrow L}]}{\Gamma\vdash \nu_{K\Rightarrow L}.(A)\colon \Sigma_{K\Rightarrow L}.G \Rightarrow H}\ \text{(TM9)}$$

$$\frac{(\Gamma, K \Rightarrow L)\vdash G\colon \mathbf{Type}}{(\Gamma, K \Rightarrow L)\vdash \psi^G_{K\Rightarrow L}\colon G \Rightarrow (\Sigma_{K\Rightarrow L}.G)[\mathbf{d}_{K\Rightarrow L}]}\ \text{(TM10)}$$

$$\frac{(\Gamma, K \Rightarrow L)\vdash A\colon G \Rightarrow H[\mathbf{d}_{K\Rightarrow L}]}{(\Gamma, K \Rightarrow L)\vdash (\nu_{K\Rightarrow L}.(A))[\mathbf{d}_{K\Rightarrow L}]\circ\psi^G_{K\Rightarrow L} = A\colon G \Rightarrow H[\mathbf{d}_{K\Rightarrow L}]}\ \text{(ETM13)}$$

$$\frac{\Gamma\vdash B\colon \Sigma_{K\Rightarrow L}.G \Rightarrow H}{\Gamma\vdash \nu_{K\Rightarrow L}.(B[\mathbf{d}_{K\Rightarrow L}]\circ\psi^G_{K\Rightarrow L}) = B\colon \Sigma_{K\Rightarrow L}.G \Rightarrow H}\ \text{(ETM14)}$$

$$\frac{(\Gamma, K \Rightarrow L)\vdash G\colon \mathbf{Type} \quad \Gamma\vdash H\colon \mathbf{Type} \quad u\colon \Gamma \to \Delta}{\begin{array}{ll}(\Delta, K[u] \Rightarrow L[u])\vdash & \psi^G_{K\Rightarrow L}[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]}, \chi_{K[u]\Rightarrow L[u]}\succ] = \\ & \psi^{G[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]},\chi_{K[u]\Rightarrow L[u]}\succ]}_{K[u]\Rightarrow L[u]} \\ & : G[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]}, \chi_{K[u]\Rightarrow L[u]}\succ] \Rightarrow (\Sigma_{K\Rightarrow L}.G)[u]\end{array}}\ \text{(ETM15)}$$

The rule (TM1) is called an *introduction rule* for C. The associated judgement J^T_{C} is also called an introduction rule.

Intuitively, a term $\Gamma\vdash A\colon K \Rightarrow L$ corresponds to a morphisms of $\mathcal{E}_\Gamma$ as follows.

	a $\mathcal{TF}$ theory T	$p\colon \mathcal{E} \longrightarrow \mathcal{B}$
(TM2)	$\mathbf{id}_K$	$id_K\colon K \to K \in \mathcal{E}_\Gamma$
(TM3)	$B\circ A$	$BA\colon K \to G \in \mathcal{E}_\Gamma$
(TM4)	$!_K$	$!_K\colon K \to 1 \in \mathcal{E}_\Gamma$
(TM5)	$A[u]$	$A[u]\colon K[u] \to L[u] \in \mathcal{E}_\Delta$
(TM6)	$\chi_{K\Rightarrow L}$	$\chi_{\Gamma,KL}\colon K[d_{\Gamma,KL}] \to L[d_{\Gamma,KL}] \in \mathcal{E}_{\mathrm{Hom}_\Gamma(K,L)}$
(TM7)$-(TM10)$	See below.	

With (ETM4), each fibre $\mathcal{E}_\Gamma$ has a terminal $\mathbf{1}[!_\Gamma]$. In fact, a terminal in the fibre $\mathcal{E}_1$ is sufficient because it is preserved by $(-)[!_\Gamma]$. (ETM 5) and (ETM6) show that p splits. (ETM 7) and (ETM 8) express the functoriality of $(-)[u]: \mathcal{E}_\Gamma \to \mathcal{E}_\Delta$. (ETM9) characterises $\chi_{K \Rightarrow L}$ as a generic morphism in the fibre $\mathcal{E}_{\mathrm{Hom}_\Gamma(K,L)}$.

$\lambda_{K \Rightarrow L}.(A)$ introduced with (TM7) represents the transposition of a morphism

$$A: G[d_{\Gamma,KL}] \to H \in \mathcal{E}_{\mathrm{Hom}_\Gamma(K,L)}$$

across the adjunction $(-)[d_{\Gamma,KL}] \dashv \Pi_{KL}$. (ETM10) and (ETM11) show the bijection $\lambda_{K \Rightarrow L}.(-)$ with $\psi^H_{K \Rightarrow L}$ being the H-component of the counit. (ETM12) is part of the Beck-Chevalley condition for Π_{KL}.

(TM9)-(ETM15) are the duals of (TM7)-(ETM12). The term $\nu_{K \Rightarrow L}.(A)$ in (TM9) represents the transposition of a morphism A across the adjunction $\Sigma_{KL} \dashv (-)[d_{\Gamma,KL}]$. The term $\psi^G_{K \Rightarrow L}$ corresponds to the unit.

In the special case of $p = \mathrm{p}_{\mathbf{Set}} : \mathbf{Fam}(\mathbf{Set}) \longrightarrow \mathbf{Set}$, this can be explained as follows. Contexts are sets of tuples of functions, context mappings are maps, types are families of sets, and reindexing is that of families of sets. Then, a well-formed term $\Gamma \vdash A: K \Rightarrow L$ is given as a family of functions indexed by Γ. Each component function A_γ $(\gamma \in \Gamma)$ maps the elements of K_γ to those of L_γ.

A *generic term* $\chi_{K \Rightarrow L}$ introduced with (TM6) is interpreted by

$$(\chi_{K \Rightarrow L})_{(\gamma,A)} = A \text{ for all } (\gamma, A) \in (\Gamma, K \Rightarrow L).$$

In (TM7), the last slot $K \Rightarrow L$ is a "dummy" slot for the domain type $G[\mathbf{d}_{K \Rightarrow L}]$ of A. For such a generalised term A, λ abstraction $\lambda_{K \Rightarrow L}.(A)$ is, in the case of $p = \mathrm{p}_{\mathbf{Set}}$, the family of maps $(\lambda_{K \Rightarrow L}.(A))_\gamma: G_\gamma \longrightarrow (\Pi_{K \Rightarrow L}.H)_\gamma$, sending g to $(A_{(\gamma,B)}g)_{(B:K_\gamma \to L_\gamma)}$. This coincides with usual λ abstraction when $G = \mathbf{1}[!_\Gamma]$. The term $\varphi^H_{K \Rightarrow L}$ is the family of evaluation map $(\varphi^H_{K \Rightarrow L})_{(\gamma,A)}: (\Pi_{K \Rightarrow L}.H)_\gamma \longrightarrow H_{(\gamma,A)}$, sending f to f_A. With this interpretation of $\varphi^H_{K \Rightarrow L}$, (ETM 10) and (ETM11) correspond to beta and eta conversion rules, respectively.

In $\mathcal{TF}$, dependent coproduct types are dual to dependent product types. (TM9) introduces its ν abstraction, which is dual to λ abstraction. In the case of $p = p_{\mathbf{Set}}$, it is the family of maps $(\nu_{K \Rightarrow L}.(A))_\gamma: (\Sigma_{K \Rightarrow L}.G)_\gamma \longrightarrow H_\gamma$, sending (B, g) to $A_{(\gamma,B)}(g)$. This is a generalised case expression which selects a map depending on the first component of its argument. The term $\psi^G_{K\Rightarrow L}$ in $(TM10)$ corresponds to the injection $(\psi^G_{K\Rightarrow L})_{(\gamma,A)}\colon G_{(\gamma,A)} \longrightarrow (\Sigma_{K\Rightarrow L}.G)_\gamma$, sending g to (A,g).

Axioms

Definition 5.3.2. The following are inference rules of T.

$$\frac{\Gamma\vdash H\colon \mathbf{Type} \quad \Gamma\vdash H'\colon \mathbf{Type}}{\Gamma\vdash H = H'\colon \mathbf{Type}}\ (\mathrm{ETY6}) \qquad (\Gamma\vdash H = H'\colon \mathbf{Type} \quad \in Ax_T)$$

$$\frac{\Gamma\vdash A\colon K\Rightarrow L \quad \Gamma\vdash A\colon K\Rightarrow L}{\Gamma\vdash A = A'\colon K\Rightarrow L}\ (\mathrm{ETM16}) \qquad (\Gamma\vdash A = A'\colon K\Rightarrow L \quad \in Ax_T)$$

General properties of equalities

Definition 5.3.2. (Continued, sketch.) Besides, there are the inference rules of T that express the following.

- (reflexivity, symmetry, transitivity) rules

  The reflexivity, symmetry, and transitivity of equality.
- (replacement) rules

  Well-formedness and typing of expressions are preserved by the replacement of equal types.
- (congruence) rules

  Type equality is compatible with Π,Σ type formation and λ,ν abstraction.

(End of **Definition** 5.3.2.)

The explicit forms of these rules are omitted as they are easily understood by analogy with the λP1's rules.

We write equations of $\mathcal{T}F$ in the usual style for the sake of readability, omitting the context.

5.3.2 Interpretations between theories in $\mathcal{TF}$

Given two theories T and T' in $\mathcal{TF}$, an interpretation from T to T' corresponds to a fibred 1-cell between two split locally small fibrations that preserves splitting and the choice of $\mathrm{Hom}_{-}(-,-)$.

Definition 5.3.3. Given two theories T and T' in $\mathcal{TF}$, a *preinterpretation* $I(-)$ from T to T' is a map which assigns to each well-formed expression of T a possibly not well-formed expression of T' and satisfying the following conditions. We follow the notation in Section 5.3.1.

(C1) $I(()) = ()$

(C2) $I((\Gamma, K \Rightarrow L)) = (I(\Gamma), I(K) \Rightarrow I(L))$

(CM1) $I(\mathbf{id}_)\Gamma = \mathbf{id}_{I(\Gamma)}$

(CM2) $I(u \circ v) = I(u) \circ I(v)$

(CM3) $I(!_\Gamma) = !_{I(\Gamma)}$

(CM4) $I(\mathbf{d}_{K \Rightarrow L}) = \mathbf{d}_{I(K) \Rightarrow I(L)}$

(CM5) $I(\prec u, A \succ) = \prec I(u), I(A) \succ$

(TY1) $I(\mathbf{1}) = \mathbf{1}$

(TY3) $I(K[u]) = I(K)[I(u)]$

(TY4) $I(\Pi_{K \Rightarrow L}.H) = \Pi_{I(K) \Rightarrow I(L)}.I(H)$

(TY5) $I(\Sigma_{K \Rightarrow L}.G) = \Sigma_{I(K) \Rightarrow I(L)}.I(G)$

(TM2) $I(\mathbf{id}_K) = \mathbf{id}_{I(K)}$

(TM3) $I(B \circ A) = I(B) \circ I(A)$

(TM4) $I(!_K) = !_{I(K)}$

(TM5) $I(A[u]) = I(A)[I(u)]$

(TM6) $I(\chi_{K \Rightarrow L}) = \chi_{I(K) \Rightarrow I(L)}$

(TM7) $I(\lambda_{K \Rightarrow L}.(A)) = \lambda_{I(K) \Rightarrow I(L)}.(I(A))$

(TM8) $I(\varphi^H_{K \Rightarrow L}) = \varphi^{I(H)}_{I(K) \Rightarrow I(L)}$

(TM9) $I(\nu_{K \Rightarrow L}.(B)) = \nu_{I(K) \Rightarrow I(L)}.(I(B))$

(TM10) $I(\psi^G_{K \Rightarrow L}) = \psi^{I(G)}_{I(K) \Rightarrow I(L)}$

An *interpretation* is a preinterpretation that sends every introduction rule and axiom of T to a derivable judgement of T'.

This definition makes sense because the interpretation of any well-formed expression of T is indeed well-formed in T'. Moreover, the following is the case.

Lemma 5.3.4. Let $I(-)$ be an interpretation from a $\mathcal{T}F$ theory T to T'. For every derivable judgement J of T, $I(J)$ is also derivable in T'.

Proof By induction on derivations in T. ∎

Chapter 6

Type theory and fibrations 2

6.1 Introduction

In this chapter, we establish the connection between the four traditional type systems introduced in the previous chapter and our $\mathcal{TF}$. Particularly, we show two variants of $\mathcal{TF}$ are equivalent to $w\mathcal{ML}$ and $\mathcal{ML}$, respectively. These are instances of the correspondence between categorical structures and type systems; in the literature, one can find a long list of other instances [2,3,41,42,43,44,46,51, 55,72,74,83]. Since the four type systems studied here represent, with appropriate constants and axioms, a wide range of type theories, these connection can be used to exploit categorical methods in an application of these type theories.

The connection is studied in terms of translations between these type systems and $\mathcal{TF}$. In one direction, this provides a categorical semantics for these type systems in fibrewise locally small fibrations. Categorical semantics of dependent type systems has been extensively studied. Our contribution in this area is the use of local smallness and the detailed, syntactic exposition of connection in both direction.

Categorical structures used in the literature to model type dependency are: contextual categories / categories with attributes [13,14,82], locally cartesian closed categories [73], categories with display-maps or relatively cartesian closed categories [36,86], $\mathcal{D}$-categories [22], comprehensive categories [61], comprehension categories [37,38], the last one being most standard by now. These are all (though it is implicit in some earlier works) fibrations with extra structure that models the context comprehension rule (context2) in Definition 5.2.4.

Local smallness is yet another such structure on fibrations and there are close connections with the above structures. Its advantage over the others is that it is already a well-established concept in category theory, particularly fundamental from the viewpoint of fibrations as parameterised categories [15,6,8,79,78]. In a locally small fibration, one can specify base objects that play a role of Hom sets of ordinary categories (Example 4.6.4).

This chapter is organised as follows. First, in Section 6.2, we show how type dependency is modelled by locally small fibrations by introducing the notion of translations from $\mathcal{G}A$ theories to $\mathcal{TF}^-$ theories, $\mathcal{TF}^-$ being a subsystem of $\mathcal{TF}$ without irrelevant Π and Σ. Then, Section 6.3 shows $\mathcal{TF}^-$ is a conservative extension of $\mathcal{G}A$. Given a $\mathcal{G}A$-theory T_G, we construct a $\mathcal{TF}^-$-theory $\mathcal{F}(T_G)$ with a faithful translation from T_G to $\mathcal{F}(T_G)$. Section 6.4 considers dependent product types of λP1 and $\mathcal{TF}\lambda$ (a subsystem without Σ). Product types allow a translation of generalised terms, so we have translations in both direction between λP1 and $\mathcal{TF}\lambda$. However the notion of generalised term in $\mathcal{TF}$ and that of functions in λP1 are subtly different. So, in order to establish equivalences, Section 6.5 studies a restricted class of $\mathcal{TF}$ theories where these two notions coincide. The restricting condition is most conveniently expressed with weak dependent coproducts, so we define a system $\mathcal{TF}\iota$ which is $\mathcal{TF}$ with appropriate extra axioms. Given this, Section 6.6 establishes the equivalence between $\mathcal{TF}\iota$ theories and $w\mathcal{ML}$ theories. For the stronger notion of dependent coproduct, one further needs to restrict $\mathcal{TF}\iota$ theories with a condition to establish the equivalence. Section 6.7 considers the condition and proves the equivalence. To keep the chapter concise, some proofs involving a detailed calculation are separately shown in Appendix A.

6.2 Translation from $\mathcal{G}A$ to $\mathcal{T}F$

In this section, we start investigating relations between $\mathcal{T}F$ and type systems with usual notation by introducing the notion of translations between theories of different type systems. In this section we focus on the point that local smallness structure gives the essence of type dependency, studying the translation of $\mathcal{G}A$ theories into $\mathcal{T}F$ theories. For this section, we do not need Π and Σ types of $\mathcal{T}F$ and only consider the subsystem without them, which we call $\mathcal{T}F^{-}$.

We define a *global element* (or simply, *element*) of a type K to be a term from $\mathbf{1}$ to K. A translation $[\![-]\!]_{\mathcal{T}F^{-}}$ associates a well-formed term $A{:}\,K$ of a $\mathcal{G}A$ theory with a global element $[\![A]\!]{:}\,\mathbf{1} \Rightarrow [\![K]\!]$ of a $\mathcal{T}F^{-}$ theory. The subscript $\mathcal{T}F^{-}$ in $[\![-]\!]_{\mathcal{T}F^{-}}$ may be omitted if it is clear from the context.

Definition 6.2.1. A map $[\![-]\!]_{\mathcal{T}F^{-}}$ from the set of well-formed expressions paired with contexts of a $\mathcal{G}A$ theory T_G to the set of possibly not well-formed expressions of a $\mathcal{T}F^{-}$ theory T_F is a *pretranslation* if and only if it satisfies the conditions listed below. The notation is the same as that in Definition 5.2.4 and 5.2.8.

$$
\begin{array}{ll}
\text{(context1)} & [\![(\ )]\!] = (\ ) \\
\text{(context2)} & [\![\vec{x}{:}\,\vec{K}, x_{n+1}{:}\,K_{n+1}]\!] = ([\![\vec{x}{:}\,\vec{K}]\!], \mathbf{1} \Rightarrow [\![K_{n+1}]\!]^{\vec{x}:\vec{K}}) \\
\text{(assumption)} & [\![x_{n+1}]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} = \chi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}} \\
\text{(weakening1)} & [\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} = [\![H]\!]^{\vec{x}:\vec{K}}[\mathbf{d}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}] \\
\text{(weakening2)} & [\![A]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} = [\![A]\!]^{\vec{x}:\vec{K}}[\mathbf{d}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}]
\end{array}
$$

$$
\begin{array}{ll}
\text{(subst1)} & [\![\mathrm{F}(\vec{A})]\!]^{\vec{y}:\vec{G}} = [\![\mathrm{F}(\vec{x})]\!]^{\vec{x}:\vec{K}}[u_n] \\
\text{(subst2)} & [\![\mathrm{C}(\vec{A})]\!]^{\vec{y}:\vec{G}} = [\![\mathrm{C}(\vec{x})]\!]^{\vec{x}:\vec{K}}[u_n] \\
 & (\text{where } u_0 = !_{[\![\vec{y}:\vec{G}]\!]}, \qquad u_i = \prec u_{i-1}, [\![A_i]\!]^{\vec{y}:\vec{G}} \succ \quad (1 \le i \le n).)
\end{array}
$$

A *translation* is a pretranslation that sends every introduction rule and axiom of T_G to a derivable judgement in T_F.

Note that $[\![-]\!]_{\mathcal{T}F^{-}}$ forgets variable names in a way similar to a translation into de Bruijn notation. Generic terms re-indexed along $\mathbf{d}$'s, χ, $\chi[\mathbf{d}]$, $\chi[\mathbf{d}{\circ}\mathbf{d}], \cdots$, play the role of de Bruijn indices 0, 1, 2,$\cdots$. The context mapping u_n in the condition (subst1) and (subst2) can be written explicitly as

$$u_n = \prec\prec\prec\cdots\prec\prec!_{[\![\vec{y}:\vec{G}]\!]}, [\![A_1]\!]^{\vec{y}:\vec{G}}\succ, [\![A_2]\!]^{\vec{y}:\vec{G}}\succ, \cdots\succ, [\![A_{n-1}]\!]^{\vec{y}:\vec{G}}\succ, [\![A_n]\!]^{\vec{y}:\vec{G}}\succ.$$

In other words, a tuple of terms is translated to a context mapping.

A translation preserves well-formedness and equality of expressions. The following theorem shows this soundness property of a translation.

Proposition 6.2.2. Let $[\![-]\!]_{\mathcal{TF}^-}$ be a translation from a $\mathcal{G}A$ theory T_G to a $\mathcal{T}F^-$ theory T_F. If $\vec{x}{:}\,\vec{K}$ **Context**, $\vec{x}{:}\,\vec{K}\vdash H{:}\,\mathbf{Type}$, $\vec{x}{:}\,\vec{K}\vdash A{:}\,H$, $\vec{x}{:}\,\vec{K}\vdash H = H'{:}\,\mathbf{Type}$, $\vec{x}{:}\,\vec{K}\vdash A = A'{:}\,H$ are derivable judgements of T_G, then

$[\![\Gamma]\!]$ **Context**
$[\![\vec{x}{:}\,\vec{K}]\!]\vdash[\![H]\!]^{\vec{x}:\vec{K}}{:}\,\mathbf{Type}$
$[\![\vec{x}{:}\,\vec{K}]\!]\vdash[\![A]\!]^{\vec{x}:\vec{K}}{:}\,\mathbf{1}\Rightarrow[\![H]\!]^{\vec{x}:\vec{K}}$
$[\![\vec{x}{:}\,\vec{K}]\!]\vdash[\![H]\!]^{\vec{x}:\vec{K}} = [\![H']\!]^{\vec{x}:\vec{K}}{:}\,\mathbf{Type}$
$[\![\vec{x}{:}\,\vec{K}]\!]\vdash[\![A]\!]^{\vec{x}:\vec{K}} = [\![A']\!]^{\vec{x}:\vec{K}}{:}\,\mathbf{1}\Rightarrow H$

are also derivable in T_F.

Proof See the proof of Proposition 6.4.2 in Appendix A. ∎

6.3 Construction of $\mathcal{F}(T_G)$

With these translations, $\mathcal{T}F^-$ is shown to be a conservative extension of $\mathcal{G}A$. For any $\mathcal{G}A$ theory T_G, there is a $\mathcal{T}F$ theory $\mathcal{F}(T_G)$ into which T_G can be embedded faithfully, i.e., there is a translation from T_G to $\mathcal{F}(T_G)$ such that if the translations of two terms of T_G are equal in $\mathcal{F}(T_G)$ then they are already equal in T_G. Moreover, $\mathcal{F}(T_G)$ is universal among the $\mathcal{T}F^-$ theories into which T_G can be translated. We fix a theory T_G in the rest of this section.

Definition 6.3.1. The $\mathcal{T}F^-$ theory $\mathcal{F}(T_G)$ and the *canonical translation* $[\![-]\!]_{\mathcal{F}(T_G)}$ from T_G to $\mathcal{F}(T_G)$ are mutually defined by

- for each constant type symbol F of T_G, $\tilde{\mathrm{F}}$ is a constant type symbol of $\mathcal{F}(T_G)$.
- for each constant term symbol F of T_G, $\tilde{\mathrm{C}}$ is a constant term symbol of $\mathcal{F}(T_G)$.
- the *canonical* translation $[\![-]\!]_{\mathcal{F}(T_G)}$ from T_G to $\mathcal{F}(T_G)$ is given by $[\![\mathrm{F}]\!]_{\mathcal{F}(T_G)} = \tilde{\mathrm{F}}$ and $[\![\mathrm{C}]\!]_{\mathcal{F}(T_G)} = \tilde{\mathrm{C}}$.
- the theory $\mathcal{F}(T_G)$ is specified with the introduction rules and axioms
  - for each introduction rule $\vec{x}\colon\vec{K}\vdash\mathrm{F}\colon\mathbf{Type}(\vec{x})$ of T_G,
    $[\![\vec{x}\colon\vec{K}]\!]_{\mathcal{F}(T_G)}\vdash\tilde{\mathrm{F}}\colon\mathbf{Type}$ is an introduction rule of $\mathcal{F}(T_G)$.
  - for each introduction rule $\vec{x}\colon\vec{K}\vdash\mathrm{C}(\vec{x})\colon H$ of T_G,
    $[\![\vec{x}\colon\vec{K}]\!]_{\mathcal{F}(T_G)}\vdash\tilde{\mathrm{C}}\colon\mathbf{1}\Rightarrow[\![H]\!]_{\mathcal{F}(T_G)}$ is an introduction rule of $\mathcal{F}(T_G)$.
  - for each axiom $\vec{x}\colon\vec{K}\vdash H = H'\colon\mathbf{Type}$ or $\vec{x}\colon\vec{K}\vdash A = A'\colon H$ of T_G,

    $$[\![\vec{x}\colon\vec{K}]\!]_{\mathcal{F}(T_G)}\vdash[\![H]\!]_{\mathcal{F}(T_G)} = [\![H']\!]_{\mathcal{F}(T_G)}\colon\mathbf{Type}$$

    or

    $$[\![\vec{x}\colon\vec{K}]\!]_{\mathcal{F}(T_G)}\vdash[\![A]\!]_{\mathcal{F}(T_G)} = [\![A']\!]_{\mathcal{F}(T_G)}\colon\mathbf{1}\Rightarrow[\![H]\!]_{\mathcal{F}(T_G)}$$

    is an axiom of $\mathcal{F}(T_G)$.

It is proved by induction on derivations in T_G that this definition indeed defines a $\mathcal{TF}^-$ theory.

Proposition 6.3.2. The canonical translation $[\![-]\!]_{\mathcal{F}(T_G)}$ from T_G to $\mathcal{F}(T_G)$ is faithful, i.e., given $\vec{x}\colon\vec{K}\vdash A\colon H$ and $\vec{x}\colon\vec{K}\vdash A'\colon H$, if

$$[\![\vec{x}\colon\vec{K}]\!]_{\mathcal{F}(T_G)}\vdash[\![A]\!]_{\mathcal{F}(T_G)} = [\![A']\!]_{\mathcal{F}(T_G)}\colon\mathbf{1}\Rightarrow[\![H]\!]_{\mathcal{F}(T_G)}$$

is derivable in $\mathcal{F}(T_G)$ then $\vec{x}\colon\vec{K}\vdash A = A'\colon H$ is already derivable in T_G.

Proof See the proof of Proposition 6.4.4 in Appendix A. ▮

Theorem 6.3.3. Given any translation $[\![-]\!]_{\mathcal{TF}^-}$ from a $\mathcal{GA}$ theory T_G to a $\mathcal{TF}^-$ theory T_F, there is a unique translation $I(-)$ from $\mathcal{F}(T_G)$ to T_F such that the iterated translation $I([\![-]\!]_{\mathcal{F}(T_G)})$ from T_G to T_F via $\mathcal{F}(T_G)$ agrees with the given $[\![-]\!]_{\mathcal{TF}^-}$.

Proof We define a pretranslation $I(-)$ from $\mathcal{F}(T_G)$ to T_F by specifying its values at the constant symbols of $\mathcal{F}(T_G)$ by

- each constant type symbol $\tilde{\mathrm{F}}$ of $\mathcal{F}(T_G)$ introduced by $[\![\vec{x}\colon \vec{K}]\!]_{\mathcal{F}(T_G)} \vdash \tilde{\mathrm{F}}\colon \mathbf{Type}$ is sent to $[\![\mathrm{F}]\!]_{\mathcal{TF}^-}$.
- each constant term symbol $\tilde{\mathrm{C}}$ of $\mathcal{F}(T_G)$ introduced by $[\![\vec{x}\colon \vec{K}]\!]_{\mathcal{F}(T_G)} \vdash \tilde{\mathrm{C}}\colon \mathbf{1} \Rightarrow [\![H]\!]_{\mathcal{F}(T_G)}$ is sent to $[\![\mathrm{C}]\!]_{\mathcal{TF}^-}$.

By definition of $I(-)$ and $[\![-]\!]_{\mathcal{F}(T_G)}$, this composite evidently agrees with $[\![-]\!]_{\mathcal{TF}^-}$. Since $[\![-]\!]_{\mathcal{TF}^-}$ is a translation,

$$I([\![\vec{x}\colon \vec{K}]\!]_{\mathcal{F}(T_G)}) \vdash I(\tilde{\mathrm{F}})\colon \mathbf{Type}$$
$$I([\![\vec{x}\colon \vec{K}]\!]_{\mathcal{F}(T_G)}) \vdash I(\tilde{\mathrm{C}})\colon \mathbf{1} \Rightarrow I([\![H]\!]_{\mathcal{F}(T_G)})$$

are derivable in T_F. So $I(-)$ is a translation. The uniqueness part is a consequence of the fact that a translation is determined by its values at constant symbols. ∎

The above theorem can be seen as a universal property of T_G with respect to $\mathcal{F}(T_G)$. Then a natural question arises: does every $\mathcal{TF}^-$ theory has such a universal $\mathcal{G}A$ theory? If so, one can conclude that $\mathcal{TF}^-$ and $\mathcal{G}A$ are equivalent type systems, but this is not the case. We will come back to this question in Section 6.6.

6.4 Translations between $\mathcal{TF}\lambda$ and λP1

In this section, we extend $[\![-]\!]_{\mathcal{TF}^-}$ to include dependent product types of λP1. We restrict $\mathcal{TF}$ to a subsystem $\mathcal{TF}\lambda$ without dependent coproducts and denote new translations by $[\![-]\!]_{\mathcal{TF}\lambda}$. With dependent products, one can also define translations in the other direction, *i.e.,* from $\mathcal{TF}\lambda$ theories to λP1 theories.

6.4.1 From λP1 Theories to $\mathcal{TF}\lambda$ Theories

Definition 6.4.1. A map $[\![-]\!]_{\mathcal{T}F\lambda}$ from the set of well-formed expressions of a λP1 theory T_G to the set of possibly not well-formed expressions of a $\mathcal{T}F\lambda$ theory T_F is a *pretranslation* if it satisfies the conditions listed in Definition 6.2.1 plus the following.

$$\begin{array}{ll}
(\Pi\mathrm{F}) & [\![\Pi x_{n+1}\colon K_{n+1}.H]\!]^{\vec{x}:\vec{K}} = \Pi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.[\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} \\
(\Pi\mathrm{I}) & [\![\lambda x_{n+1}\colon K_{n+1}.(A)]\!]^{\vec{x}:\vec{K}} = \lambda_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.([\![A]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}) \\
(\Pi\mathrm{E}) & [\![BC]\!]^{\vec{x}:\vec{K}} = \varphi^{[\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![C]\!]^{\vec{x}:\vec{K}}\succ]\circ[\![B]\!]^{\vec{x}:\vec{K}}
\end{array}$$

A *translation* is a pretranslation that sends every introduction rule and axioms of T_G to a derivable judgement of T_F.

The properties of translations $[\![-]\!]_{\mathcal{T}F^-}$ given in Section 6.3 (soundness, construction of $\mathcal{F}(T_G)$, conservativeness of canonical translations) extend to this definition. The proofs are in Appendix A.

Proposition 6.4.2. Proposition 6.2.2 holds with $\mathcal{T}F^-$, $[\![-]\!]_{\mathcal{T}F^-}$, and $\mathcal{G}A$ replaced by $\mathcal{T}F\lambda$, $[\![-]\!]_{\mathcal{T}F\lambda}$, and λP1, respectively.

Definition 6.4.3. Given a λP1 theory T_G, the $\mathcal{T}F\lambda$ theory $\mathcal{F}(T_G)$ defined in the same way as in Definition 6.3.1.

Proposition 6.4.4. Proposition 6.3.2 holds with $\mathcal{T}F^-$, $[\![-]\!]_{\mathcal{T}F^-}$, and $\mathcal{G}A$ replaced by $\mathcal{T}F\lambda$, $[\![-]\!]_{\mathcal{T}F\lambda}$, and λP1, respectively.

6.4.2 From $\mathcal{T}F\lambda$ Theories to λP1 Theories

Next, we define the notion of translations from $\mathcal{T}F\lambda$ theories to λP1 theories. This cannot be done for $\mathcal{G}A$ theories since the translation of generalised terms requires function types, which is a special case of dependent product types.

Definition 6.4.5. A map $[\![-]\!]_{\lambda\mathrm{P1}}$ from the set of well-formed expressions of a $\mathcal{T}F\lambda$ theory T_F to the set of possibly not well-formed expressions of a λP1 theory

T_G is a *pretranslation* if it satisfies the following conditions. The subscript λP1 may be omitted when it is obvious. We adopt the notation in Definition 5.3.2 with the following addition:

$$\begin{array}{rclrcl} \Gamma &=& (K_1 \Rightarrow L_1, K_2 \Rightarrow L_2, \cdots K_n \Rightarrow L_n), & [\![u]\!] &=& (B_1, \cdots, B_n), \\ \Delta &=& (G_1 \Rightarrow H_1, G_2 \Rightarrow H_2, \cdots, G_m \Rightarrow H_m), & [\![v]\!] &=& (C_1, \cdots, C_m). \end{array}$$

(C1) $[\![(\ )]\!] = (\ )$

(C2) $[\![(\Gamma, K \Rightarrow L)]\!] = ([\![\Gamma]\!], v_{KL}\colon ([\![K]\!] \supset [\![L]\!]))$

(CM1) $[\![\mathbf{id}_\Gamma]\!] = (v_{K_1L_1}, v_{K_2L_2}, \cdots, v_{K_nL_n})$

(CM2) $[\![u \circ v]\!] = (B_1[v_{G_1H_1}/C_1, \cdots, v_{G_mH_m}/C_m], \\ \cdots, B_n[v_{G_1H_1}/C_1, \cdots, v_{G_mH_m}/C_m])$

(CM3) $[\![!_\Gamma]\!] = (\ )$

(CM4) $[\![\mathbf{d}_{K \Rightarrow L}]\!] = (v_{K_1L_1}, \cdots, v_{K_nL_n})$

(CM5) $[\![\prec u, A \succ]\!] = ([\![u]\!], [\![A]\!])$

(TY1) $[\![\mathbf{1}]\!] = \mathbf{1}$

(TY3) $[\![H[u]]\!] = [\![H]\!][v_{K_1L_1}/B_1, \cdots, v_{K_nL_n}/B_n]$

(TY4) $[\![\Pi_{K \Rightarrow L}.H]\!] = \Pi x\colon ([\![K]\!] \supset [\![L]\!]).[\![H]\!][v_{KL}/x]$

(TM2) $[\![\mathbf{id}_K]\!] = \lambda x\colon [\![K]\!].(x)$

(TM3) $[\![B \circ A]\!] = \lambda x\colon [\![K]\!].([\![B]\!]([\![A]\!](x)))$

(TM4) $[\![!_K]\!] = \lambda x\colon [\![K]\!].(*)$

(TM5) $[\![\chi_{K \Rightarrow L}]\!] = v_{KL}$

(TM6) $[\![A[u]]\!] = [\![A]\!][v_{K_1L_1}/B_1, \cdots, v_{K_nL_n}/B_n]$

(TM7) $[\![\lambda_{K \Rightarrow L}.(A)]\!] = \lambda x\colon [\![G]\!].(\lambda z\colon ([\![K]\!] \supset [\![L]\!]).([\![A]\!][v_{KL}/z]x))$

(TM8) $[\![\varphi^H_{K \Rightarrow L}]\!] = \lambda x\colon [\![\Pi_{K \Rightarrow L}.H]\!].((x\ v_{KL}))$,

where v_{KL} is a variable symbol uniquely assigned to the pair (K, L), and x, z are fresh variables other than the v_{KL}'s.

A *translation* is A pretranslation that sends every introduction rule and axiom of T_F to a derivable judgement of T_G.

A pretranslation $[\![-]\!]_{\lambda\mathrm{P1}}$ is determined by its value at constant types and terms of T_F.

A well-formed context of T_F is translated to a well-formed context of T_G replacing each slot $K \Rightarrow L$ with a declaration $v_{KL}\colon [\![K]\!]\supset[\![L]\!]$ of a canonically named variable.

A well-formed context mapping $u\colon \Delta \to \Gamma$ of T_F is translated to a tuple $(B_1, B_2, \cdots, B_n)$ of well-formed terms of T_G. Each term B_i has free variables declared in $[\![\Delta]\!]$ and is of type $[\![K_i]\!]\supset[\![L_i]\!]$. With substitution of closed terms for free variables in the B_i's, $[\![u]\!]$ can be regarded as a map sending a tuple of closed terms of type $[\![G_1]\!]\supset[\![H_1]\!], \cdots, [\![G_m]\!]\supset[\![H_m]\!]$ to a tuple of closed terms of type $[\![K_1]\!]\supset[\![L_1]\!], \cdots, [\![K_n]\!]\supset[\![L_n]\!]$.

Re-indexing operations $(-)[u]$ are translated by the substitutions of terms for free variables. When $u = \mathbf{d}_{K\Rightarrow L}$ for some K and L, translations $[\![H[\mathbf{d}_{K\Rightarrow L}]]\!]$ and $[\![H]\!]$ are the same type expressions in different contexts.

Proposition 6.4.6. Let $[\![-]\!]_{\lambda\mathrm{P1}}$ be a translation from a $\mathcal{TF}\lambda$ theory T_F to a $\mathcal{G}A$ theory T_G. If Γ **Context**, $\Gamma\vdash H\colon \mathbf{Type}$, $\Gamma\vdash A\colon K \Rightarrow L$, $u = u'\colon \Delta \to \Gamma$, $\Gamma\vdash H = H'\colon \mathbf{Type}$, and $\Gamma\vdash A = A'\colon K \Rightarrow L$ are derivable judgements of T_F, then

$$
\begin{aligned}
&[\![\Gamma]\!]\ \mathbf{Context}\\
&[\![\Gamma]\!]\vdash[\![H]\!]\colon \mathbf{Type}\\
&[\![\Gamma]\!]\vdash[\![A]\!]\colon ([\![K]\!] \supset [\![L]\!])\\
&\Delta\vdash B_i = B_i'\colon [\![K_i]\!][v_{K_1L_1}/B_1, \cdots, v_{K_{i-1}L_{i-1}}/B_{i-1}] \quad (1 \le i \le n)\\
&[\![\Gamma]\!]\vdash[\![H]\!] = [\![H']\!]\colon \mathbf{Type}\\
&[\![\Gamma]\!]\vdash[\![A]\!] = [\![A']\!]\colon [\![K]\!]\supset[\![L]\!]
\end{aligned}
$$

are also derivable in T_G, respectively, where $(K_1 \Rightarrow L_1, \cdots, K_n \Rightarrow L_n) = \Gamma$, $(B_1, \cdots, B_n) = [\![u]\!]$, and $(B_1', \cdots, B_1') = [\![u']\!]$.

Proof By induction on derivations. See Appendix A. ▮

6.5 $\mathcal{TF}\iota$ theories

Although we have translations between $\mathcal{TF}\lambda$ theories and λP1 theories in both directions, these are not subtle enough to establish the equivalence of the two type systems. In the previous section, we used functions (terms of dependent types) of λP1 to translate generalised terms of $\mathcal{TF}\lambda$. However, when we regard generalised terms as functions, and composition of global elements as application, a $\mathcal{TF}\lambda$ theory need not satisfy the condition equivalent to the functional completeness of λP1. In any λP1 theory, application $A \mapsto Ax$ gives the bijection $A \leftrightarrow B$ in

$$\frac{\Gamma \vdash A: K \supset L}{\Gamma, x: K \vdash B: L,} \cong$$

while in a $\mathcal{TF}\lambda$ theory the composition $A \mapsto A \circ \chi_{\mathbf{1} \Rightarrow K}$ need not give a bijection $A \leftrightarrow B$ in

$$\frac{\Gamma \vdash A: K \Rightarrow L}{\Gamma, \mathbf{1} \Rightarrow K \vdash B: \mathbf{1} \Rightarrow L[\mathbf{d}_{\mathbf{1} \Rightarrow K}].} \not\cong$$

A counterexample is given by considering a $\mathcal{TF}\lambda$ theory that corresponds to $p_{\mathcal{C}}: \mathbf{Fam}(\mathcal{C}) \longrightarrow \mathbf{Set}$ where $\mathcal{C}(1, -)$ is not fully faithful.

Therefore, to establish the equivalence between a $\mathcal{TF}$-style type system and a λP1-style system in later sections, one needs to restrict attention to $\mathcal{TF}$ theories where this bijection exists. The simplest way to stipulate this condition, without adding a new operation on terms, is to move up to type systems with dependent coproducts and add new constants to express the required axiom.

First, note that in any $w\mathcal{ML}$ theory T, the terms of type K are in one-to-one correspondence with those of the type $\Sigma x: K.\mathbf{1}$ with the bijections $A: K \mapsto \langle A, * \rangle: \Sigma x: K.\mathbf{1}$ and $B: \Sigma x: K.\mathbf{1} \mapsto \mathbf{let}_{\langle x,y \rangle = B}\ \mathbf{in}\ x: K$.

The corresponding property of dependent coproduct types of a $\mathcal{TF}$ theory would be the assertion that, for every type $\Gamma \vdash K: \mathbf{Type}$, the term $\nu_{\mathbf{1} \Rightarrow K}.(\chi_{\mathbf{1} \Rightarrow K})$ of type $\Gamma \vdash \nu_{\mathbf{1} \Rightarrow K}.(\chi_{\mathbf{1} \Rightarrow K}): \Sigma_{\mathbf{1} \Rightarrow K}.\mathbf{1} \Rightarrow K$ has the inverse. However, it is not the case in general. We stipulate $\nu_{\mathbf{1} \Rightarrow K}.(\chi_{\mathbf{1} \Rightarrow K})$ to be an isomorphism introducing a constant ι_K. In other words, we require every type K to be isomorphic to a certain coproduct of $\mathbf{1}$.

Definition 6.5.1. A $\mathcal{TF}\iota$ *theory* is defined as in Definition 5.3.2 with the following additional rules.

$$\frac{\Gamma\vdash K\colon \mathbf{Type}}{\Gamma\vdash \iota_K\colon K\Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.\mathbf{1}}\ \text{(TM11)}$$

$$\frac{\Gamma\vdash K\colon \mathbf{Type}}{\Gamma\vdash \nu_{\mathbf{1}\Rightarrow K}.(\chi_{\mathbf{1}\Rightarrow K})\circ\iota_K = \mathbf{id}_K\colon K\Rightarrow K}\ \text{(ETM16)}$$

$$\frac{\Gamma\vdash K\colon \mathbf{Type}}{\Gamma\vdash \iota_K\circ\nu_{\mathbf{1}\Rightarrow K}.(\chi_{\mathbf{1}\Rightarrow K}) = \mathbf{id}_{\Sigma_{\mathbf{1}\Rightarrow K}.\mathbf{1}}\colon \Sigma_{\mathbf{1}\Rightarrow K}.\mathbf{1}\Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.\mathbf{1}}\ \text{(ETM17)}$$

With the constant ι_K, one has the desired bijection as follows.

Lemma 6.5.2. In a theory T in $\mathcal{TF}\iota$, generalised terms $\Gamma\vdash A\colon K\Rightarrow L$, $(\Gamma,\mathbf{1}\Rightarrow K)\vdash B\colon \mathbf{1}\Rightarrow L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]$, and $\Gamma\vdash C\colon \mathbf{1}\Rightarrow \Pi_{\mathbf{1}\Rightarrow K}.L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]$ are in bijection with each other.

Proof one has the following natural bijection of terms.

$$\cfrac{\cfrac{\Gamma\vdash A\colon K\Rightarrow L}{\Gamma\vdash A'\colon \Sigma_{\mathbf{1}\Rightarrow K}.\mathbf{1}\Rightarrow L}\cong}{\cfrac{(\Gamma,\mathbf{1}\Rightarrow K)\vdash B\colon \mathbf{1}\Rightarrow L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]}{\Gamma\vdash C\colon \mathbf{1}\Rightarrow \Pi_{\mathbf{1}\Rightarrow K}.L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]}\cong}\cong$$

∎

We name the explicit expressions for these bijections as follows. For each K, L, A, B, and C such that $\Gamma\vdash K, L\colon \mathbf{Type}$, $\Gamma\vdash A\colon K\Rightarrow L$, $(\Gamma,\mathbf{1}\Rightarrow K)\vdash B\colon \mathbf{1}\Rightarrow L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]$, and $\Gamma\vdash C\colon \mathbf{1}\Rightarrow \Pi_{\mathbf{1}\Rightarrow K}.L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]$, the following are defined.

$$\begin{array}{l}
\Gamma\vdash K\supset L\equiv \Pi_{\mathbf{1}\Rightarrow K}.L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]\colon \mathbf{Type}\\
(\Gamma,\mathbf{1}\Rightarrow K)\vdash \mathrm{uncurr}_K(A)\equiv A[\mathbf{d}_{\mathbf{1}\Rightarrow K}]\circ\chi_{\mathbf{1}\Rightarrow K}\colon \mathbf{1}\Rightarrow L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]\\
\Gamma\vdash \mathrm{curr}_K(B)\equiv \nu_{\mathbf{1}\Rightarrow K}.(B)\circ\iota_K\colon K\Rightarrow L\\
\Gamma\vdash \mathrm{code}_{K\Rightarrow L}(A)\equiv \lambda_{\mathbf{1}\Rightarrow K}.(\mathrm{uncurr}_K(A))\colon \mathbf{1}\Rightarrow K\supset L\\
(\Gamma,\mathbf{1}\Rightarrow K)\vdash \lambda^{-1}_{\mathbf{1}\Rightarrow K}(C)\equiv \varphi^{L}_{\mathbf{1}\Rightarrow K}\circ C[\mathbf{d}_{\mathbf{1}\Rightarrow K}]\colon \mathbf{1}\Rightarrow L[\mathbf{d}_{\mathbf{1}\Rightarrow K}]\\
\Gamma\vdash \mathrm{decode}_{K\Rightarrow L}(C)\equiv \mathrm{curr}_K(\lambda^{-1}_{\mathbf{1}\Rightarrow K}(C))\colon K\Rightarrow L
\end{array}$$

Iterating application of these bijections, one can replace all expressions of generalised terms with global elements.

Definition 6.5.3. A well-formed expression E of a $\mathcal{TF}\iota$ theory T_F is *hereditarily elementary* (h.e., for short) if a structural judgement for E has a derivation in T_F such that all the slots of all the contexts appearing in the derivation is of the form $\mathbf{1} \Rightarrow K$ for some K.

Note that K must also be h.e.. Iterating $\mathrm{uncurr}_{\_}(-)$, one has

Theorem 6.5.4. (Definition-Theorem) Let T_F be a $\mathcal{TF}\iota$ theory whose constant types and terms are all h.e.. Then, for each well-formed expression E, one can define an h.e. expression E^+ such that the mapping $E \mapsto E^+$ is bijective in the following sense.

- For each well-formed expression E of T_F, its *h.e. form* E^+ has the following structural judgements.
  - For Γ **Context**, Γ^+ **Context**.
  - For $u\colon \Delta \to \Gamma$, $u^+\colon \Delta^+ \to \Gamma^+$.
  - For $\Gamma \vdash H\colon \mathbf{Type}$, $\Gamma^+ \vdash H^+\colon \mathbf{Type}$.
  - For $\Gamma \vdash A\colon K \Rightarrow L$, $\Gamma^+ \vdash A^+\colon \mathbf{1} \Rightarrow K^+ \supset L^+$.
- The map $A \mapsto A^+$ gives a bijection between terms A of type $\Gamma \vdash A\colon K \Rightarrow L$ and B of type $\Gamma \vdash B\colon \mathbf{1} \Rightarrow K^+ \supset L^+$ with inverse

  $$B \mapsto B^- \equiv \delta_L \circ \mathrm{decode}_{K^+ \Rightarrow L^+}(B)[\alpha_\Gamma] \circ \gamma_K.$$

- For each well-formed context Γ, there are context mappings $\alpha_\Gamma\colon \Gamma \to \Gamma^+$ and $\beta_\Gamma\colon \Gamma^+ \to \Gamma$, satisfying $\alpha_\Gamma \circ \beta_\Gamma = \mathbf{id}_{\Gamma^+}\colon \Gamma^+ \to \Gamma^+$ and $\beta_\Gamma \circ \alpha_\Gamma = \mathbf{id}_\Gamma\colon \Gamma \to \Gamma$.
- For each context mapping $u\colon \Delta \to \Gamma$, $u^+ \circ \alpha_\Delta = \alpha_\Gamma \circ u\colon \Delta \to \Gamma^+$ holds.
- For each type $\Gamma \vdash K\colon \mathbf{Type}$, there are terms $\Gamma \vdash \gamma_K\colon K \Rightarrow K^+[\alpha_\Gamma]$ and $\Gamma \vdash \delta_K\colon K^+[\alpha_\Gamma] \Rightarrow K$, satisfying $\Gamma \vdash \gamma_K \circ \delta_K = \mathbf{id}_{K^+[\alpha_\Gamma]}\colon K^+[\alpha_\Gamma] \Rightarrow K^+[\alpha_\Gamma]$ and $\Gamma \vdash \delta_K \circ \gamma_K = \mathbf{id}_K\colon K \Rightarrow K$.

For the definition of E^+, see Appendix A. Then, one has:

Proposition 6.5.5. If all constant types and terms contained in a well-formed expression E are hereditarily elementary then so is E^+. ∎

The bijection $A \leftrightarrow B$ in Lemma 6.5.2 gives another characterisation of $\mathcal{T}F\iota$ theories among all $\mathcal{T}F$ theories.

Proposition 6.5.6. In any $\mathcal{T}F\iota$ theory T_F, there is a natural bijection between terms $\Gamma\vdash A: K \Rightarrow L$ and context mappings $u: (\Gamma, \mathbf{1} \Rightarrow K) \to (\Gamma, \mathbf{1} \Rightarrow L)$ such that $\mathbf{d}_{\mathbf{1}\Rightarrow L} \circ u = \mathbf{d}_{\mathbf{1}\Rightarrow K}$. Conversely, any $\mathcal{T}F$ theory T_F having this natural bijection $A \leftrightarrow u$ also has the inverse of $\nu_{\mathbf{1}\Rightarrow K}.(\chi_{\mathbf{1}\Rightarrow K})$.

Proof See Appendix A. ∎

This connects Jacobs' full comprehension categories ([37] and Section 4.10 of [38]) and $\mathcal{T}F\iota$ theories. As in the remark after Proposition 4.6.7, the functor $\mathrm{Hom}(1, -)$ that sends K to $\mathbf{d}_{\mathbf{1}\Rightarrow K}$ gives rise to a comprehension category. The natural bijection shows fully faithfulness of comprehension categories that correspond to $\mathcal{T}F\iota$ theories.

6.6 Equivalence between $\mathcal{T}F\iota$ and $w\mathcal{M}L$

Having restricted $\mathcal{T}F$ theories to $\mathcal{T}F\iota$ theories, we can now show the equivalence between $\mathcal{T}F\iota$ and $w\mathcal{M}L$. First, the notion of translations given in Section 6.4 is extended to include dependent coproduct types. Then the construction $\mathcal{F}(T_G)(-)$ in Section 6.3 can be extended to the $w\mathcal{M}L$ case and proved to be a part of an adjoint equivalence between the category of $\mathcal{T}F\iota$ theories and that of $w\mathcal{M}L$ theories.

6.6.1 Translations between $\mathcal{T}F\iota$ and $w\mathcal{M}L$

Definition 6.6.1. A *pretranslation* from a $w\mathcal{M}L$ theory T_w to a $\mathcal{T}F\iota$ theory T_F is a map $[\![-]\!]_{\mathcal{T}F\iota}$ from the set of well-formed expressions of T_w to the set of possibly not well-formed expressions of T_F which satisfies the conditions in Definition 6.4.1 and those listed below. We use the notation in Section 5.2.3.

$$(w\Sigma\mathrm{F})\quad [\![\Sigma x_{n+1}\colon K_{n+1}.G]\!]^{\vec{x}:\vec{K}} = \Sigma_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.[\![G]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}$$

$$(w\Sigma\mathrm{I})\quad [\![\langle A_1, A_2\rangle]\!]^{\vec{x}:\vec{K}} = \psi^{G}_{\mathbf{1}\Rightarrow K}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![A_1]\!]^{\vec{x}:\vec{K}}\succ]\circ[\![A_2]\!]^{\vec{x}:\vec{K}}$$

$$(w\Sigma\mathrm{E})\quad [\![\mathbf{let}_{\langle x_{n+1},y\rangle=A}\ \mathbf{in}\ B]\!]^{\vec{x}:\vec{K}}$$
$$= (\nu_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.(\mathrm{curr}_{[\![G]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}}([\![B]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1},y:G})))\circ[\![A]\!]^{\vec{x}:\vec{K}}$$

A *translation* is a pretranslation that sends every introduction rule to a h.e. judgement derivable in T_F.

Note that ι_K constant in $\mathrm{curr}_{-}(-)$ is needed to translate $\mathbf{let}_{\langle x,y\rangle=A}$ **in** B.

The additional condition for constant types and terms to be sent to h.e. types and constants is not so strong in view of Theorem 6.5.4. What we gain is the strictness of translations. The soundness of interpretation is proved as usual but requires complex calculations, relying on the bijection given in Lemma 6.5.2.

Proposition 6.6.2. If $[\![-]\!]_{\mathcal{TF}\iota}$ is a translation from $w\mathcal{M}L$ theory T_w to $\mathcal{TF}\iota$ theory T_F, all derivable judgements of T_w are translated to those of T_F by $[\![-]\!]_{\mathcal{TF}\iota}$. ▮

Proof See Appendix A. ▮

Definition 6.6.3. A *(pre)translation* $[\![-]\!]_{wML}$ from a $\mathcal{TF}\iota$ theory to a $w\mathcal{M}L$ theory is a map that satisfies the conditions (C1)-(TM8) in Definition 6.4.2, plus the following condition.

$$(\mathrm{TY5})\quad [\![\Sigma_{K\Rightarrow L}.G]\!] = \Sigma z\colon ([\![K]\!] \supset [\![L]\!]).[\![G]\!][v_{KL}/z]$$

$$(\mathrm{TM9})\quad [\![\nu_{K\Rightarrow L}.(A)]\!] = \lambda z\colon [\![\Sigma_{K\Rightarrow L}.G]\!].(\mathbf{let}_{\langle x,y\rangle=z}\ \mathbf{in}\ [\![A]\!][v_{KL}/x]y)$$

$$(\mathrm{TM10})\quad [\![\psi^{G}_{K\Rightarrow L}]\!] = \lambda y\colon [\![G]\!].(\langle v_{KL}, y\rangle)$$

$$(\mathrm{TM11})\quad [\![\iota_K]\!] = \lambda z\colon [\![K]\!].(\langle \lambda w\colon \mathbf{1}.(z), *\rangle)$$

Proposition 6.6.4. The soundness lemma 6.4.6 extends to $\mathcal{TF}\iota$ theories. ▮

6.6.2 Construction of $\mathcal{F}(T_w)$

Construction of $\mathcal{F}(T_G)$ for a $\mathcal{G}A$ theory T_G and the canonical translation from $\mathcal{F}(T_G)$ to T_G in Section 6.3 can be extended to the $w\mathcal{M}L$ case. For $w\mathcal{M}L$ theory T_w, the corresponding $w\mathcal{M}L$ theory $\mathcal{F}(T_w)$ and the canonical translation $[\![-]\!]_{\mathcal{F}(T_w)}$ from T_w to $\mathcal{F}(T_w)$ are defined similarly to $\mathcal{F}(T_G)$ and $[\![-]\!]_{\mathcal{F}(T_G)}$, respectively. Proposition 6.4.4 and Theorem 6.3.3 extends to the $w\mathcal{M}L$ case. See Appendix A for the additional calculation.

Proposition 6.6.5. The canonical translation $[\![-]\!]_{\mathcal{F}(T_w)}$ from a $w\mathcal{M}L$ theory T_w to $\mathcal{F}(T_w)$ is faithful. ▮

Theorem 6.6.6. Given any translation $[\![-]\!]_{\mathcal{T}F\iota}$ from a $w\mathcal{M}L$ theory T_w into a $\mathcal{T}F\iota$ theory T_F, there is a unique interpretation $I(-)$ from $\mathcal{F}(T_w)$ to T_F such that the composite translation $I([\![-]\!]_{\mathcal{F}(T_w)})$ from T_w to T_F via $\mathcal{F}(T_w)$ agrees with the given $[\![-]\!]_{\mathcal{T}F\iota}$. ▮

6.6.3 Construction of $\mathcal{G}(T_F)$

The bijection in 6.5.2 between generalised terms and functions of global elements exactly characterises $\mathcal{T}F$ theories of the form $\mathcal{F}(T_w)$. This subsection answers the problem posed in the end of Section 6.3.3 with $\mathcal{T}F\lambda$ replaced by $w\mathcal{M}L$.

Theorem 6.6.7. Given a $\mathcal{T}F\iota$ theory T_F, one can construct a $w\mathcal{M}L$ theory $\mathcal{G}(T_F)$ and a canonical translation $[\![-]\!]_{\mathcal{G}(T_F)}$ from $\mathcal{G}(T_F)$ to T_F with the following universal property. For any $\mathcal{T}F\iota$ theory T'_F and any translation $[\![-]\!]_{\mathcal{T}F\iota}$ from $\mathcal{G}(T_F)$ to T'_F, there is a unique interpretation $I(-)$ from T_F to T'_F such that the composite $I([\![-]\!]_{\mathcal{G}(T_F)})$ agrees with $[\![-]\!]_{\mathcal{T}F\iota}$.

We only give the construction and interpretation, leaving the proof to Appendix A. In the following, a $\mathcal{T}F\iota$ theory T_F is fixed.

Definition 6.6.8. The $w\mathcal{M}L$ theory $\mathcal{G}(T_F)$ is defined by

- for every h.e. type H of T_F, c_H is a constant type symbol of $\mathcal{G}(T_F)$.
- for every h.e. term A of T_F, c_A is a constant term symbol of $\mathcal{G}(T_F)$.
- the *canonical translation* $[\![-]\!]_{\mathcal{G}(T_F)}$ from expressions of $\mathcal{G}(T_F)$ to those of T_F is the translation which sends constant symbols c_H and c_A of $\mathcal{G}(T_F)$ onto H and A, respectively.
- for every h.e. type $\mathbf{1} \Rightarrow K_1, \ldots, \mathbf{1} \Rightarrow K_n \vdash H\colon \mathbf{Type}$,

  $$x_1\colon \mathrm{c}_{K_1}, \ldots, x_n\colon \mathrm{c}_{K_n} \vdash \mathrm{c}_H(\vec{x})\colon \mathbf{Type}$$

  is an introduction rule of $\mathcal{G}(T_F)$.
- for every h.e. term $\mathbf{1} \Rightarrow K_1, \ldots, \mathbf{1} \Rightarrow K_n \vdash A\colon \mathbf{1} \Rightarrow H$,

  $$x_1\colon \mathrm{c}_{K_1}, \ldots, x_n\colon \mathrm{c}_{K_n} \vdash \mathrm{c}_A(\vec{x})\colon \mathrm{c}_H(\vec{x})$$

  is an introduction rule of $\mathcal{G}(T_F)$.
- the axioms of $\mathcal{G}(T_F)$ are the equality judgements of $\mathcal{G}(T_F)$ whose translations by $[\![-]\!]_{\mathcal{G}(T_F)}$ are derivable in T_F.

Definition 6.6.9. A preinterpretation $I(-)_{\mathcal{T}F\iota}$ from $\mathcal{G}(T_F)$ to T'_F is specified relatively to $[\![-]\!]_{\mathcal{T}F\iota}$ by

- for every constant type F introduced as $\Gamma \vdash \mathrm{F}\colon \mathbf{Type}$ of T_F, $I(\mathrm{F})_{\mathcal{T}F\iota} = [\![\mathrm{c}_{\mathrm{F}^+}]\!]_{\mathcal{T}F\iota}[\alpha_{I(\Gamma)_{\mathcal{T}F\iota}}]$.
- for every constant term C introduced as $\Gamma \vdash \mathrm{C}\colon K \Rightarrow L$, $I(\mathrm{C})_{\mathcal{T}F\iota} = ([\![\mathrm{c}_{\mathrm{C}^+}]\!]_{\mathcal{T}F\iota})^{-}$.

By Theorem 6.5.4, $I(-)_{\mathcal{T}F\iota}$ is a translation.

The theory $\mathcal{G}(T_F)$ has another universal property other than that of Theorem 6.6.7.

Theorem 6.6.10. For any $w\mathcal{M}L$ theory T_w and any translation $[\![-]\!]_{\mathcal{T}F\iota}$ from T_w to T_F, there is a unique interpretation $[\![-]\!]_{w\mathcal{M}L}$ from T_w to $\mathcal{G}(T_F)$ such that $[\![[\![-]\!]_{w\mathcal{M}L}]\!]_{\mathcal{G}(T_F)}$ agrees with $[\![-]\!]_{\mathcal{T}F\iota}$.

Proof By definition, the value of $[\![-]\!]_{\mathcal{T}F\iota}$ is h.e.. Define $I(H)_{\mathcal{T}F\iota} = \mathrm{c}_{[\![H]\!]_{\mathcal{T}F\iota}}$ etc. ∎

6.6.4 The equivalence

Through the construction of $\mathcal{G}(T_F)$ and $\mathcal{F}(T_w)$, one can show that $\mathcal{T}F\iota$ and $w\mathcal{M}L$ are equivalent type systems.

Definition 6.6.11. The category $\mathcal{T}F\iota\mathbf{Cat}$ ($w\mathcal{M}L\mathbf{Cat}$) has $\mathcal{T}F\iota$ ($w\mathcal{M}L$) theories as objects and interpretations between them as morphisms.

Lemma 6.6.12. Let T_F and T_w be a TFi theory and a $w\mathcal{M}L$ theory, respectively. Then, there is a bijection between $\mathcal{T}F\iota\mathbf{Cat}(\mathcal{F}(T_w), T_F)$ and $w\mathcal{M}L\mathbf{Cat}(T_w, \mathcal{G}(T_F))$.

Proof With Theorem 6.6.6 and 6.6.10, the following bijections are obtained.

$$\begin{array}{cl} \mathcal{F}(T_w) \to T_F \in w\mathcal{M}L\mathbf{Cat} & \\ \hline \text{The translations from } T_w \text{ to } T_F & \cong \text{(by Th. 6.6.6)} \\ \hline T_w \to \mathcal{G}(T_F) \in \mathcal{T}F\iota\mathbf{Cat} & \cong \text{(by Th. 6.6.10)} \end{array}$$

∎

One can naturally extend the construction $\mathcal{F}(-)$ to a functor from $\mathcal{T}F\iota\mathbf{Cat}$ to $w\mathcal{M}L\mathbf{Cat}$. The bijection of Lemma 6.6.12 exhibits an adjunction $\mathcal{F}(-) \dashv \mathcal{G}(-)$ between $\mathcal{T}F\iota\mathbf{Cat}$ and $w\mathcal{M}L\mathbf{Cat}$.

Proposition 6.6.13. Given a $\mathcal{T}F\iota$ theory T_F, $\mathcal{F}(\mathcal{G}(T_F))$ is isomorphic to T_F, that is, there are interpretations

$$I(-)_{\mathcal{T}F\iota}\colon \mathcal{F}(\mathcal{G}(T_F)) \to T_F \in \mathcal{T}F\iota\mathbf{Cat}$$
$$J(-)\colon T_F \to \mathcal{F}(\mathcal{G}(T_F)) \in \mathcal{T}F\iota\mathbf{Cat}$$

such that the composite interpretations $I(J(-))_{\mathcal{T}F\iota}$ and $J(I(-)_{\mathcal{T}F\iota})$ agree with the identity interpretations on T_F and $\mathcal{F}(\mathcal{G}(T_F))$, respectively. Similarly, for every $w\mathcal{M}L$ theory T_w, $\mathcal{G}(\mathcal{F}(T_w))$ is isomorphic to T_w.

Proof Let $I(-)_{\mathcal{T}F\iota}$ be the counit of the adjunction $\mathcal{F}(-) \dashv \mathcal{G}(-)$. Namely, $I(-)_{\mathcal{T}F\iota}$ is the unique interpretation given by Theorem 6.6.6 where we take $[\![-]\!]_{\mathcal{T}F\iota}$ to be the canonical translation $[\![-]\!]_{\mathcal{G}(T_F)}$ from $\mathcal{G}(T_F)$ to T_F. $J(-)$ is given by Theorem 6.6.7 by replacing $[\![-]\!]_{\mathcal{T}F\iota}$ by the canonical translation $[\![-]\!]_{\mathcal{F}(\mathcal{G}(T_F))}$ from

$\mathcal{G}(T_F)$ to $\mathcal{F}(\mathcal{G}(T_F))$. Then, we have both $I(J(\llbracket - \rrbracket_{\mathcal{G}(T_F)}))_{\mathcal{T}F\iota} = \llbracket - \rrbracket_{\mathcal{G}(T_F)}$ and $\mathrm{Id}(\llbracket - \rrbracket_{\mathcal{G}(T_F)}) = \llbracket - \rrbracket_{\mathcal{G}(T_F)}$ where $\mathrm{Id}(-)$ is the identity interpretation. By the uniqueness asserted by Theorem 6.6.7, $I(J(-))_{\mathcal{T}F\iota} = \mathrm{Id}(-)$. Similarly, $J(I(-)_{\mathcal{T}F\iota}) = \mathrm{Id}(-)$ can be shown using Theorem 6.6.10. ▮

Therefore, we have:

Theorem 6.6.14. $\mathcal{F}(-) \dashv \mathcal{G}(-)$ defines an adjoint equivalence between $\mathcal{T}F\iota\mathbf{Cat}$ and $w\mathcal{M}L\mathbf{Cat}$. ▮

6.7 Equivalence between $\mathcal{T}F\iota\sigma$ and $\mathcal{M}L$

We now turn to the stronger notion of dependent coproduct types in $\mathcal{M}L$ theories. By adding another family of constants to $\mathcal{T}F\iota$ with axioms, one obtains an equivalence similar to that in the previous section.

6.7.1 $\mathcal{T}F\iota\sigma$ theories

To obtain the equivalence, we add to $\mathcal{T}F\iota$ theories new constants σ_{KL} that act as the second projections $\mathbf{snd}(-)$ in $\mathcal{M}L$. First we fix some notation for elements of coproduct types of $\mathcal{T}F$. In view of Lemma 6.5.2, we focus only on global elements.

Definition 6.7.1. For types $\Gamma \vdash K\colon \mathbf{Type}$ and $(\Gamma, \mathbf{1} \Rightarrow K) \vdash L\colon \mathbf{Type}$ in a $\mathcal{T}F\iota$ theory, the following are defined.

$$(\Gamma, \mathbf{1} \Rightarrow K, \mathbf{1} \Rightarrow L) \vdash \mathrm{pair}_{KL} \equiv \mathrm{uncurr}_L(\psi^L_{\mathbf{1}\Rightarrow K})\colon \mathbf{1} \Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.L[\mathbf{d}_{\mathbf{1}\Rightarrow K} \circ \mathbf{d}_{\mathbf{1}\Rightarrow L}]$$
$$\Gamma, \mathbf{1} \Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.L \vdash \mathrm{fst}_{KL} \equiv \mathrm{uncurr}_{\Sigma_{\mathbf{1}\Rightarrow K}.L}(\nu_{\mathbf{1}\Rightarrow K}.(\chi_{\mathbf{1}\Rightarrow K} \circ !_L))\colon \mathbf{1} \Rightarrow K[\mathbf{d}_{\mathbf{1}\Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.L}]$$

For two terms $\Gamma \vdash A_1\colon \mathbf{1} \Rightarrow K$ and $\Gamma \vdash A_2\colon \mathbf{1} \Rightarrow L[\prec \mathbf{id}_\Gamma, A_1 \succ]$ of a $\mathcal{T}F\iota$ theory, define the *pair* of A_1 and A_2 to be the term

$$\Gamma \vdash \mathrm{pair}_{KL}[\prec\prec \mathbf{id}_\Gamma, A_1 \succ, A_2 \succ] (= \varphi^L_{\mathbf{1}\Rightarrow K}[\prec \mathbf{id}_\Gamma, A_1 \succ] \circ A_2)\colon \mathbf{1} \Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.L.$$

The first component of a pair can be extracted by fst_{KL}:

$$\mathrm{fst}_{KL}[\prec\mathbf{id}_\Gamma, \mathrm{pair}_{KL}[\prec\prec\mathbf{id}_\Gamma, A_1\succ, A_2\succ]\succ] = A_1.$$

But, in general, $\mathcal{TF}\iota$ theories support no operations that extract second components of pairs. The constants σ_{KL} are adjoined to $\mathcal{TF}\iota$ theories for this purpose.

Definition 6.7.2. A $\mathcal{TF}\iota\sigma$ theory is given as in Definition 6.5.1 with the following additional inference rules.

$$\frac{(\Gamma, \mathbf{1} \Rightarrow K)\vdash L\colon \mathbf{Type}}{\Gamma, \mathbf{1} \Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.L\vdash\sigma_{KL}\colon \mathbf{1} \Rightarrow L[\prec d_{\mathbf{1}\Sigma_{\mathbf{1}\Rightarrow K}.L}, \mathrm{fst}_{KL}\succ]}\ \text{(TM12)}$$

$$\frac{(\Gamma, \mathbf{1} \Rightarrow K)\vdash L\colon \mathbf{Type}}{\begin{array}{c}\Gamma, \mathbf{1} \Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.L\vdash\ \psi^L_{\mathbf{1}\Rightarrow K}[\prec\mathbf{d}_{\mathbf{1}\Rightarrow\Sigma_{\mathbf{1}\Rightarrow K}.L}, \mathrm{fst}_{KL}\succ]\circ\sigma_{KL} = \chi_{\mathbf{1}\Rightarrow\Sigma_{\mathbf{1}\Rightarrow K}.L}\\ :\mathbf{1} \Rightarrow (\Sigma_{\mathbf{1}\Rightarrow K}.L)[\mathbf{d}_{\mathbf{1}\Rightarrow\Sigma_{\mathbf{1}\Rightarrow K}.L}]\end{array}}\ \text{(ETM18)}$$

$$\frac{(\Gamma, \mathbf{1} \Rightarrow K)\vdash L\colon \mathbf{Type}}{(\Gamma, \mathbf{1} \Rightarrow K, \mathbf{1} \Rightarrow L)\vdash\sigma_{KL}[\prec\mathbf{d}_{\mathbf{1}\Rightarrow K}\circ\mathbf{d}_{\mathbf{1}\Rightarrow L}, \mathrm{pair}_{KL}\succ] = \chi_{\mathbf{1}\Rightarrow L}\colon \mathbf{1}\Rightarrow L[\mathbf{d}_{\mathbf{1}\Rightarrow L}]}\ \text{(ETM19)}$$

The second component of $\mathrm{pair}_{KL}[\prec\prec\mathbf{id}_\Gamma, A_1\succ, A_2\succ]$ can be obtained by using (ETM19) as follows.

$$\begin{aligned}&\sigma_{KL}[\prec\mathbf{id}_\Gamma, \mathrm{pair}_{KL}[\prec\prec\mathbf{id}_\Gamma, A_1\succ, A_2\succ]\succ]\\ &\quad= \sigma_{KL}[\prec\mathbf{d}_{\mathbf{1}\Rightarrow K}\circ\mathbf{d}_{\mathbf{1}\Rightarrow L}, \mathrm{pair}_{KL}\succ][\prec\prec\mathbf{id}_\Gamma, A_1\succ, A_2\succ]\\ &\quad= \chi_{\mathbf{1}\Rightarrow L}[\prec\prec\mathbf{id}_\Gamma, A_1\succ, A_2\succ] = A_2\end{aligned}$$

The rule (ETM18) shows that the pairing is surjective. Also note that any term σ' that satisfies (ETM18) and (ETM19) is necessarily equal to σ_{KL}.

6.7.2 Translation between $\mathcal{TF}\iota\sigma$ and $\mathcal{ML}$

Definition 6.7.3. A *(pre)translation* $[\![-]\!]_{ML}$ from a $\mathcal{TF}\iota\sigma$ theory T_F to a $\mathcal{ML}$ theory T_M is a map that satisfies the conditions given in Definition 6.6.3 and the following.

$$\text{(TM12)}\quad [\![\sigma_{KL}]\!]_{ML} = \lambda z\colon \mathbf{1}.(\mathbf{snd}(v_{\mathbf{1}\Sigma_{\mathbf{1}\Rightarrow K}.L}*))$$

The soundness of interpretations extends to this case without difficulty.

Lemma 6.7.4. If $[\![-]\!]_{ML}$ is a translation from a $\mathcal{TF}\iota\sigma$ theory T_F to a $\mathcal{ML}$ theory T_M, all the derivable judgements of T_F are translated to those of T_M by $[\![-]\!]_{ML}$. ∎

Definition 6.7.5. A *(pre)translation* $[\![-]\!]_{\mathcal{TF}\iota\sigma}$ from a $\mathcal{ML}$ theory T_M to a $\mathcal{TF}\iota\sigma$ theory T_F is a map that satisfies the conditions in Definition 6.4.1, plus the following conditions.

$$(\Sigma\mathrm{F})\quad [\![\Sigma x_{n+1}\colon K_{n+1}.G]\!]^{\vec{x}:\vec{K}} = \Sigma_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.[\![G]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}$$

$$(\Sigma\mathrm{I})\quad [\![\langle A_1, A_2\rangle]\!]^{\vec{x}:\vec{K}} = \psi^{[\![G]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![A_1]\!]^{\vec{x}:\vec{K}}\succ]\circ[\![A_2]\!]^{\vec{x}:\vec{K}}$$

$$(\Sigma\mathrm{E1})\quad [\![\mathbf{fst}(A)]\!]^{\vec{x}:\vec{K}} = \nu_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.(\chi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}\circ !_{[\![G]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}})\circ[\![A]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}$$

$$(\Sigma\mathrm{E2})\quad [\![\mathbf{snd}(A)]\!]^{\vec{x}:\vec{K}} = \sigma_{[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[\![L]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![A]\!]^{\vec{x}:\vec{K}}\succ]$$

Proposition 6.7.6. If $[\![-]\!]_{\mathcal{TF}\iota\sigma}$ is a translation from a $\mathcal{ML}$ theory T_M to a $\mathcal{TF}\iota\sigma$ theory T_F, all the derivable judgements of T_M are translated to those of T_F by $[\![-]\!]_{\mathcal{TF}\iota\sigma}$.

Proof The soundness for the equality inference rules ($\mathbf{fst}(-)$eq) and ($\mathbf{snd}(-)$eq) in Section 5.2.3 is easily checked. As for ($\langle\rangle$eq) one has the following equation. Writing $[\![G]\!]$ for $[\![G]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}$,

$$\begin{aligned}
&[\![\langle\mathbf{fst}(A), \mathbf{snd}(A)\rangle]\!]^{\vec{x}:\vec{K}}\\
&\quad = \psi^{[\![G]\!]}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, \mathrm{fst}_{[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[\![G]\!]}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![A]\!]^{\vec{x}:\vec{K}}\succ]\succ]\\
&\qquad \circ\sigma_{[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[\![G]\!]}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![A]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad = (\psi^{[\![G]\!]}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}[\prec\mathbf{d}_{\mathbf{1}\Rightarrow[\![\Sigma_{\mathbf{1}\Rightarrow K}.L]\!]^{\vec{x}:\vec{K}}}, \mathrm{fst}_{[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[\![G]\!]}\succ]\\
&\qquad \circ\sigma_{[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[\![G]\!]})[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![A]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad = \chi_{\mathbf{1}\Rightarrow[\![\Sigma_{\mathbf{1}\Rightarrow K}.L]\!]^{\vec{x}:\vec{K}}}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![A]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad = [\![A]\!]^{\vec{x}:\vec{K}}.
\end{aligned}$$

∎

6.7.3 The equivalence

With the soundness theorems 6.7.4 and 6.7.6 for $[\![-]\!]_{ML}$'s and $[\![-]\!]_{\mathcal{T}F\iota\sigma}$'s, the equivalence between $\mathcal{T}F\iota$ theories and $w\mathcal{M}L$ theories can be easily extended to one between $\mathcal{T}F\iota\sigma$ theories and $\mathcal{M}L$ theories. All the propositions and theorems given in Section 6.6.2 – 6.6.4 also hold for $\mathcal{M}L$ theories with an obvious modification given in Appendix A.

Writing $\mathcal{T}F\iota\sigma\mathbf{Cat}$ ($\mathcal{M}L\mathbf{Cat}$) for the category of $\mathcal{T}F\iota\sigma$ ($\mathcal{M}L$) theories and interpretations between them, one obtains:

Theorem 6.7.7. The constructions $\mathcal{F}(-)$ and $\mathcal{G}(-)$ form an adjoint equivalence between $\mathcal{T}F\iota\sigma\mathbf{Cat}$ and $\mathcal{M}L\mathbf{Cat}$.

Chapter 7

Analysis of deliverables

7.1 Introduction

The aim of this chapter is to give an example of categorical analysis on concrete objects syntactically defined within an ambient type theory. We chose *deliverables* [52,11] in the LEGO system for our example because of their rich structure and their immediate usefulness in the problem of program correctness.

Categorical structure revealed by such analysis serves two related purposes. First, it allows us to study the semantics of concrete objects with our theory of universal structure in Chapter 2. By abstracting out the detail of the original syntactic definition, clarity is gained. Our computer checked language for categories with universal structure can be used to develop and formalise such a semantic study. More concretely, one can define a formalised category `Del` of syntactically defined deliverables as a term of the type `Cat` in Chapter 3 and instantiate the generic commands, lemmas given there with `Del`. (Note, however, one needs to adjust the universe levels used in Chapter 3 in the particular case of `Del`.)

Second, the categorical structure of concrete objects gives rise to a type theory within a categorical framework, which complements and helps the above semantic study. Here, two levels of type theory are considered; one is the ambient type theory and the other is an *object-level* type theory for the concrete objects.

The formalised semantic study and the object-level type theory retain a direct connection with the syntactic definition, all coexisting within the ambient type theory.

In this chapter, we give a categorical analysis of deliverables that provides a basis for this development. The main technical contribution is our analysis of second-order deliverables as a Kleisli construction over first-order deliverables. This gives a clear understanding of the syntactic "family construction" [52] and provide a basis for integrating the two object-level type theories for first- and second-order deliverables. (Also see Hermida [34] Section 4.3.4.)

In Section 7.2, we give the syntactic definition of first-order and second-order deliverables. Section 7.3 describes the categorical structure of deliverables, emphasising the fibred nature of structure on both first-order and second-order deliverables. To analyse the relationship between first- and second-order deliverables, abstract first- and second-order deliverables are defined in Section 7.4 as mild abstractions, as the term indicates, to give the categorical structure of deliverables. This structure is used in Section 7.5 to show that second-order deliverables are obtained form first-order deliverables by a Kleisli construction. Finally, using the connection between Kleisli constructions and polynomial categories, the family construction of second-order deliverables is analysed categorically.

In his original work [52], McKinna also investigated the categorical structure of deliverables. Using the categorical combinators derived from the structure, he developed an extensive methodology for provably correct programming. We emphasise the fibred aspect of the structure so that, for the second purpose mentioned above, we can use familiar (dependent) type theory rather than categorical combinators. With a different motivation, Hofmann [35] uses deliverables and their structure to study models of extensional type theory within intensional type theory. Also, Mendler [54] studies a construction on indexed categories related with abstract deliverables to give models for his modal logic.

7.2 Basic definitions

We first recall the syntactic definition of deliverables defined in our idealised LEGO; we assume proof irrelevance and extensionality. Following McKinna [52], we work in a fixed context Γ in order to give simpler definitions. In terms of the fibration that models the ambient type theory, we work in the fibre over a fixed base object. However, an abstract treatment of deliverables remains the same if we consider various contexts at once.

First, a specification is formalised simply as a type and proposition on it.

Definition 7.2.1. A *specification* is a pair of terms

$$(\Gamma \vdash s\colon \mathrm{Type}, \qquad \Gamma \vdash S\colon s \to \mathrm{Prop}).$$

As usual, we omit contexts and/or types in writing the pair of terms (s, S). Also we use the letters $\mathcal{S}$, $\mathcal{T}$, $\cdots$ for specifications (s, S), (t, T), $\cdots$, respectively.

Given two specifications $\mathcal{S}$ and $\mathcal{T}$, a first-order deliverable is a function with a proof that it produces from an input satisfying S an output satisfying T.

Definition 7.2.2. A *first-order deliverable* from $\mathcal{S}$ to $\mathcal{T}$ is a pair

$$(\Gamma \vdash f\colon s \to t, \qquad \Gamma \vdash p\colon \forall x\colon s.Sx \Rightarrow T(fx)).$$

In [52], a Σ type is used to make this pair a term of the ambient type theory.

First-order deliverables can not specify input-output relations. This lead to the definitions of relativised specifications and second-order deliverables (see Section 3.2.1 of McKinna's thesis [52]).

Definition 7.2.3. A *relativised specification* over $\mathcal{S}$ is a pair of terms

$$(\Gamma \vdash t\colon \mathrm{Type}, \qquad \Gamma \vdash R\colon s \to t \to \mathrm{Prop}).$$

Definition 7.2.4. Given two relativised specifications (t, Q) and (u, R) over $\mathcal{S}$, a *second-order deliverable* over $\mathcal{S}$ from (t, Q) to (u, R) is a pair of terms

$$(\Gamma \vdash f\colon s \to t \to u, \qquad \Gamma \vdash p\colon \forall x\colon s.\forall y\colon t.S(x) \Rightarrow Q(x, y) \Rightarrow R(x, (fxy))).$$

A second-order deliverable may be defined between relativised specifications over two different specifications; a morphism in $\mathbf{Del}_2$ defined in Definition 7.3.4 accounts for this.

7.3 Universal structures for deliverables

In this section we examine the universal structure that arises from the definitions in the previous section. These structures are also examined in Chapter 3 of [52] using semi-adjunction [33] to cater for proof relevance of LEGO. Here we give a simpler version by restricting our attention to our idealised version of LEGO with extensionality and proof irrelevance in order to obtain familiar categorical structures rather than exotic semi- counterparts of them that appear in [52]. This restriction is justified by observing that the idea of deliverables is to separate functions and proofs, so that we need not look for computational contents of the proof.

In Chapter 3 of [52], it is shown that both first-order deliverables and second-order deliverables over a given specification form a semi-cartesian closed category. We recall the fact in our simpler setting. We also show that both categories are fibred categories that are locally small.

7.3.1 Fibred category $\mathbf{Del}_1$

Definition 7.3.1.

- The category $\mathbf{Del}_1$ has specifications as objects and first-order deliverables as morphisms. The composition of two deliverables

$$(f\colon s_0 \to s_1, p\colon \forall x\colon s_0.S_0x \Rightarrow S_1(fx))\colon \mathcal{S}_0 \longrightarrow \mathcal{S}_1,$$
$$(g\colon s_1 \to s_2, q\colon \forall x\colon s_1.S_1x \Rightarrow S_2(gx))\colon \mathcal{S}_1 \longrightarrow \mathcal{S}_2$$

is defined by

$$(g,q)\circ(f,p) = (\Gamma \vdash \lambda x{:}\, s_0.g(fx){:}\, s_0 \to s_2,\\ \Gamma \vdash \lambda x{:}\, s_0.\lambda h{:}\, S_0x.q(fx)(pxh){:}\, \forall x{:}\, s_0.S_0x \Rightarrow S_2(g(fx)))\\ :\mathcal{S}_0 \longrightarrow \mathcal{S}_2$$

- The category **Typ** has types and functions under the context Γ as objects and morphisms, with the obvious composition defined by LEGO.
- The functor $\mathrm{p}_{\mathbf{Del}_1}{:}\, \mathbf{Del}_1 \longrightarrow \mathbf{Typ}$ is given by projections of the first components for both specifications and first-order deliverables.

The identity on $\mathcal{S} \in \mathbf{Del}_1$ is given by

$$\mathrm{id}_{\mathcal{S}} = (\lambda x{:}\, s.x{:}\, s \to s, \lambda x{:}\, s.\lambda h{:}\, Sx.h{:}\, \forall x{:}\, s.Sx \Rightarrow Sx).$$

Note that, without extensionality, even the definition of **Typ** requires a modified identification of terms to force the associativity of composition [52].

Proposition 7.3.2. The functor $\mathrm{p}_{\mathbf{Del}_1}$ is a fibration.

Proof The canonical cartesian morphism with codomain $\mathcal{S}$ over a function $f{:}\, t \to s$ is defined by

$$(f{:}\, t \to s, \lambda x{:}\, t.\lambda h{:}\, S(fx).h){:}\, (t, \lambda x{:}\, t.S(fx)) \longrightarrow (s, S),$$

which we write $\hat{f}\mathcal{S}{:}\, (t, S[f]) \longrightarrow \mathcal{S}$. To see $\hat{f}\mathcal{S}$ is cartesian, consider a function $g{:}\, u \to t$ and a specification (u, U). The composition with $\hat{f}\mathcal{S}$

$$\hat{f}\mathcal{S} \circ (-){:}\, \mathbf{Del}_{1g}((u,U),(t,S[f])) \to \mathbf{Del}_{1fg}((u,U),(s,S))$$

is trivially bijective and natural in (u, U) with the inverse

$$\begin{array}{ccc} \mathbf{Del}_{1fg}((u,U),(s,S)) & \to & \mathbf{Del}_{1g}((u,U),(t,S[f])) \\ (fg, p{:}\, \forall x{:}\, u.Ux \Rightarrow S(f(gx))) & \mapsto & (g, p{:}\, \forall x{:}\, u.Ux \Rightarrow S[f](gx)). \end{array}$$

It is also easy to check that $\widehat{(-)}(-)$ does give a splitting on $\mathrm{p}_{\mathbf{Del}_1}$. ∎

The fibre $\mathbf{Del}_{1s}$ over $s \in \mathbf{Typ}$ is the preorder of s-parameterised propositions $S{:}\, s \to \mathrm{Prop}$ where a morphism from S_0 to S_1 exists if there is a proof $p{:}\, \forall x{:}\, sS_0x \Rightarrow$

$S_1 x$. This is clearly isomorphic to the preorder of propositions under the context $(\Gamma, x{:}\, s)$.

Proposition 7.3.3. The fibration $\mathrm{p}_{\mathbf{Del}_1}$ is locally small.

Proof Given $R, S \in \mathrm{p}_{\mathbf{Del}_1} s$, the projection $d_{s,RS}\colon \mathrm{Hom}_s(R, S) \longrightarrow s$ in **Typ** (Definition 4.6.5) is given by the first projection $\pi_0\colon (\Sigma x{:}\, s.\forall x{:}\, s.Sx \Rightarrow Rx) \longrightarrow s$. This being a Hom-object depends on proof irrelevance. ▮

With proof irrelevance and extensionality, it is also easy to show that $\mathrm{p}_{\mathbf{Del}_1}$ has products and coproducts along $d_{s,RS}$.

7.3.2 Cartesian closed structure on $\mathbf{Del}_1$

Next we observe that $\mathrm{p}_{\mathbf{Del}_1}$ is a fibred cartesian closed category. This gives an instance of logical predicates for simply typed λ calculus (see Section 4.2 of Hermida [34]).

Terminal object The terminal object of $\mathbf{Del}_1$ is given by $(\mathrm{Unit}, \lambda x{:}\, \mathrm{Unit}.\mathrm{True})$, which is evidently preserved by $\mathrm{p}_{\mathbf{Del}_1}$. The unique morphism from (s, S) is $(\lambda x{:}\, s.*,\ \lambda x{:}\, s.\lambda h{:}\, Sx.\mathrm{triv})$, where $*$ and triv are the unique elements of Unit and True, respectively.

Binary products Since each fibre $\mathbf{Del}_{1s}$ has binary product preserved by substitutions $(s, S_0) \times_s (s, S_1) \equiv (s, \lambda x{:}\, s.S_0 x \wedge S_1 x)$, $\mathbf{Del}_1$ has fibred binary products. Spelling out, the product of two specifications $\mathcal{S}$ and $\mathcal{T}$ is given by

$$\mathcal{S} \times \mathcal{T} = (s \times t, \lambda x{:}\, s \times t.S(\pi_0 x) \wedge T(\pi_1 x))$$

The adjunction is

$$\frac{(u, U) \xrightarrow{(f,p)} (s, S) \qquad (u, U) \xrightarrow{(f,p)} (t, T)}{(u, U) \xrightarrow[(\lambda x:u.\langle fx, gx\rangle, \lambda x:u.\lambda h:Ux.\langle pxh, qxh\rangle)]{} (s \times t, \lambda x{:}\, s \times t.S(\pi_0 x) \wedge T(\pi_1 x))}$$

$$\frac{(u,U) \xrightarrow{(f,p)} (s \times t, \lambda x{:}\, s \times t.S(\pi_0 x) \wedge T(\pi_1 x))}{(u,U) \xrightarrow{(\pi_0 f, \lambda x:u.\lambda h:Ux.\pi_0(pxh))} (s,S) \qquad (u,U) \xrightarrow{(\pi_0 f, \lambda x:u.\lambda h:Ux.\pi_1(pxh))} (t.T).}$$

Exponentials Each fibre $\mathbf{Del}_{1s}$ ($s \in \mathbf{Typ}$) is a cartesian closed category with exponentials given by pointwise implications. For $R, S{:}\, s \to \mathrm{Prop}$,

$$S^R \equiv \lambda x{:}\, s.Rx \Rightarrow Sx.$$

The adjunction is given by

$$Q \to S^R \equiv \forall x{:}\, s.Qx \Rightarrow Rx \Rightarrow Sx$$
$$\text{if and only if } \forall x{:}\, s.Qx \wedge Rx \Rightarrow Sx.$$

Given $f{:}\, t \to s$, the substitution $(-)[f]$ preserves the implication as $S^R[f] = \lambda x{:}\, t.R(fx) \Rightarrow S(fx) = S[f]^{R[f]}$.

7.3.3 Fibred category $\mathbf{Del}_2$

Definition 7.3.4. • The category $\mathbf{Del}_2$ has as objects relativised specifications. A morphism from a relativised specification $\mathcal{R} \equiv (r, R)$ over a specification $\mathcal{S} \equiv (s, S)$ to $\mathcal{V} \equiv (v, V)$ over $\mathcal{T} \equiv (t, T)$ is a quadruple (g, p, f, q) where $(g, p) \in \mathbf{Del}_1(\mathcal{S}, \mathcal{T})$ and (f, q) is a second-order deliverable from $\mathcal{R}$ to $(v, \lambda x{:}\, s.\lambda y{:}\, v.V(gx, y))$ over $\mathcal{S}$, *i.e.*,

$$\begin{aligned}
(&\Gamma \vdash g{:}\, s \to t,\\
&\Gamma \vdash p{:}\, \forall x{:}\, s.Sx \Rightarrow T(gx),\\
&\Gamma \vdash f{:}\, s \to r \to v,\\
&\Gamma \vdash q{:}\, \forall x{:}\, s \forall y{:}\, r.Sx \Rightarrow R(x,y) \Rightarrow V(gx, fxy)).
\end{aligned}$$

Given $\mathcal{W} \equiv (w, W)$ over $\mathcal{U} \equiv (u, U)$ and a morphism $(k, a, h, b){:}\, \mathcal{V} \longrightarrow \mathcal{W}$, the composition $(k, a, h, b) \circ (g, p, f, q){:}\, \mathcal{R} \longrightarrow \mathcal{W}$ is

$$\begin{aligned}
(&k \circ g,\\
&\lambda x{:}\, s.\lambda c{:}\, Sx.a(gx)(pxc),\\
&\lambda x{:}\, s.\lambda y{:}\, r.h(gx)(fxy),
\end{aligned}$$

$$\lambda x\colon s.\lambda y\colon r.\lambda c\colon Sx.\lambda d\colon R(x,y).b(gx)(fxy)(pxc)(qxycd)).$$

- The functor $\mathrm{p}_{\mathbf{Del}_2}\colon \mathbf{Del}_2 \longrightarrow \mathbf{Del}_1$ sends relativised specifications to their underlying first-order specifications, and morphisms to first-order deliverables comprising their first two components.

Proposition 7.3.5. $\mathrm{p}_{\mathbf{Del}_2}$ is a split fibration.

Proof Given $(r,R) \in \mathbf{Del}_{2(s,S)}$ and $(t,T) \xrightarrow{(g,p)} (s,S)$ in $\mathbf{Del}_1$, define $\widehat{(g,p)}(r,R)$ by

$$(r,R)[(g,p)] \equiv (r, \lambda x\colon t\lambda y\colon r.R(gx,y)),$$
$$\widehat{(g,p)}(r,R)\colon (r,R)[(g,p)] \xrightarrow{(g,p,\lambda x:t\lambda y:r.y,p')} (r,R)$$

where

$$p' \equiv \lambda x\colon t\lambda y\colon r.\lambda a\colon Tx.\lambda b\colon R(gx,y).b$$
$$\colon \forall x\colon t.\forall y\colon r.Tx \Rightarrow R(gx,y) \Rightarrow R(gx,y).$$

Then, given $(u,U) \xrightarrow{(k,q)} (t,T)$ in $\mathbf{Del}_1$ and $(v,V) \in \mathbf{Del}_{2(u,U)}$, the composition with $\widehat{(g,p)}(r,R)$ is a trivial bijection

$$\begin{array}{rcl} \widehat{(g,p)}(r,R) \circ (-)\colon \mathbf{Del}_{2(k,q)}((v,V),(r,R)[(g,p)]) & \longrightarrow & \mathbf{Del}_{2(g,p)\circ(k,q)}((v,V),(r,R)) \\ (k,q,f,c) & \mapsto & (g \circ k, \\ & & \lambda x\colon u.\lambda y\colon Ux.p(kx)(qxy), \\ & & f, \\ & & c) \end{array}$$

It is also easy to check that this gives a splitting on $\mathrm{p}_{\mathbf{Del}_2}$. ∎

A morphisms in the fibre $\mathbf{Del}_{2\mathcal{S}}$ over a specification $\mathcal{S}$ is precisely a second-order deliverables over $\mathcal{S}$.

7.3.4 Cartesian closed structure on $\mathbf{Del}_2$

Terminal object The terminal object of $\mathbf{Del}_2$ is given by the relativised specification $1 \equiv (\mathrm{Unit}, \lambda x\colon s.\lambda u\colon \mathrm{Unit}.True)$ over the terminal specification. This is a fibred terminal object, since $\mathrm{p}_{\mathbf{Del}_2}$ obviously preserves it. The unique morphism from (r, R) over (s, S) is given by the unique first-order deliverable $!_{(s,S)}$ from (s, S) to $(\mathrm{Unit}, \mathrm{True})$ and the second-order deliverable

$$\begin{aligned}&(\lambda x\colon s.\lambda y\colon r.*,\ \lambda x\colon s.\lambda y\colon r.\lambda p\colon Sx.\lambda q\colon R(x, y).\mathrm{triv})\\ &\quad : (r, R) \longrightarrow 1[!_{(s,S)}].\end{aligned}$$

Binary products Given two relativised specifications (r, R) and (t, T) over (s, S), the binary product of (r, R) and (t, T) in the fibre $\mathbf{Del}_{2(s,S)}$ is given by $(r, R) \times (t, T) \equiv (r \times t, \lambda x\colon s.\lambda y\colon r \times t.\lambda p\colon Sx.R(x, \pi_0 y) \wedge T(x, \pi_1 y))$. Given a first-order deliverable $(f, p)\colon (u, U) \longrightarrow (s, S)$, the substitution functor

$$(-)[(f, p)]\colon \mathbf{Del}_{2(s,S)} \longrightarrow \mathbf{Del}_{2(u,U)}$$

preserves the product as

$$\begin{aligned}((r, R) \times (t, T))[(f, p)] &= (r \times t, \lambda x\colon u.\lambda y\colon r \times t.\lambda p\colon Ux.R(gx, \pi_0 y) \wedge T(gx, \pi_1 y))\\ &= (r, R)[(f, p)] \times (t, T)[(f, p)].\end{aligned}$$

So, $\mathbf{Del}_2$ has fibred binary products.

Exponentials The fibration $\mathrm{p}_{\mathbf{Del}_2}$ has fibred exponentials as follows. First, each fibre $\mathbf{Del}_{2(s,S)}$ is cartesian closed. The exponential object of (r, R) and (t, T) is given by

$$(t, T)^{(r,\, R)} \equiv (r \to t, \lambda x\colon s.\lambda y\colon r \to t.\forall z\colon r.R(x, z) \Rightarrow T(x, yz)).$$

The transposition of a morphism $(f, q)\colon (v, V) \times (r, R) \longrightarrow (t, T)$ in $\mathbf{Del}_{2(s,S)}$ is

$$\begin{aligned}&(\lambda x\colon s.\lambda y\colon v.(\lambda w\colon r.fx\langle y, w\rangle),\\ &\quad \lambda x\colon s.\lambda y\colon v.\lambda p\colon Sx.\lambda a\colon V(x, y).\lambda z\colon r.\lambda b\colon R(x, z).qx\langle y, z\rangle p\langle a, b\rangle)\\ &\qquad : (v, V) \longrightarrow (t, T)^{(r,\, R)}.\end{aligned}$$

For the inverse direction, given $(f, q)\colon (v, V) \longrightarrow (t, T)^{(r,\, R)}$ in $\mathbf{Del}_{2(s,S)}$, the transposition is

$$\begin{aligned}&(\lambda x\colon s.\lambda y\colon v \times r.fx(\pi_0 y)(\pi_1 y),\\ &\quad \lambda x\colon s.\lambda y\colon v \times r.\lambda p\colon Sx.\lambda a\colon V(x, \pi_0 y) \wedge R(x, \pi_1 y).qx(\pi_0 y)p(\pi_0 a)(\pi_1 y)(\pi_1 a))\end{aligned}$$

$$: (v, V) \times (r, R) \longrightarrow (t, T).$$

Second, to see this is preserved by substitution, let $(f, p)\colon (u, U) \longrightarrow (s, S)$ be a first-order deliverable. Then in fact,

$$\begin{aligned}(t, T)^{(r,\, R)}[(f, p)] &= (r \to t, \lambda x\colon u.\lambda y\colon r \to t.\forall z\colon r.R(fx, z) \Rightarrow T(fx, yz)) \\ &= (t, T)[(f, p)]^{(r,\, R)[(f, p)]}.\end{aligned}$$

7.3.5 Local smallness

Proposition 7.3.6. The fibration $\mathrm{p}_{\mathbf{Del}_2}$ is locally small.

Given two relativised specifications (r, R) and (t, T) over (s, S), the Hom-projection

$$d_{(s,S),(r,R)(t,T)}\colon \mathrm{Hom}_{(s,S)}((r, R), (t, T)) \longrightarrow (s, S)$$

is given by

$$\begin{tikzcd}
(s \times (r \to t), \lambda x\colon s.\lambda g\colon r \to t.Sx \wedge \forall y\colon r.Sx \wedge R(x, y) \Rightarrow T(x, gy)) \arrow[d, "{(\pi_0, \lambda(x, g)\colon s \times (r \to t).\lambda p.\pi_0 p(x, g))}"] \\
(s, S)
\end{tikzcd}$$

∎

Once Hom-projections are given, it is straightforward to show that $\mathrm{p}_{\mathbf{Del}_2}$ also has products and coproducts along $d_{(s,S),(r,R)(t,T)}$. Therefore, $\mathrm{p}_{\mathbf{Del}_2}$ supplies a model for our categorical framework in Chapter 5.

7.4 Abstract deliverables

In this section, we rephrase the above syntactic definition in terms of the fibration. The level of abstraction is chosen to make clear the construction of second-order deliverables out of first-order ones.

7.4.1 First-order deliverables

The fist order deliverables are treated simply as a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ with fibred finite products. Finite products are needed for the construction of the second order deliverables in terms of p.

Definition 7.4.1. (McKinna [52])

Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ with fibred finite products, the objects and the morphisms of $\mathcal{E}$ are called *abstract* specification and *abstract* first-order deliverables.

The categories $\mathcal{B}$ and $\mathcal{E}$ correspond to **Typ** and $\mathbf{Del}_1$, respectively. In more detail, we informally identify them as follows and use objects of $\mathcal{B}$ *etc.* in place of types *etc.*.

- An object $I \in \mathcal{B}$ corresponds to a type I under the context Γ, and a morphism $u: J \longrightarrow I$ in $\mathcal{B}$ to a function $u: J \to I$, regarding J and I as types in the latter.
- An object $A \in \mathcal{E}_I$ corresponds to a specification $A \equiv (I, A: I \to \text{Prop})$. A morphism $f: B \underset{(u)}{\longrightarrow} A$ in $\mathcal{E}$ over u corresponds to a first-order deliverable $f \equiv (u, f: \forall x: J.Bx \Rightarrow A(ux))$.

Particularly, if the above f in $\mathcal{E}$ is cartesian, the corresponding deliverable is isomorphic to $\hat{u}A = (u, \lambda x: J.\lambda h: A(ux).h)$, and B is identified with the specification $(J, \lambda x: J.A(ux))$. However, we do not assume a particular cleavage nor a splitting on p.

7.4.2 Second-order deliverables

Now we identify a relativised specification with a diagram in $\mathcal{E}$. First, note that a relativised specification $\mathcal{R} \equiv (J, R: I \times J \to \text{Prop})$ over a specification $A \equiv (I, A: I \to \text{Prop})$ can be specified by A, J, and the specification $R \equiv (I \times J, R: I \times J \to \text{Prop})$, with the obvious identification of the two types $I \to J \to \text{Prop}$

and $I \times J \to \mathrm{Prop}$. Therefore, one may identify a relativised specification $\mathcal{R}$ with a tuple $(A \in \mathcal{E}_I, J \in \mathcal{B}, R \in \mathcal{E}_{I\times J})$. Next, we examine what corresponds to a morphism of $\mathbf{Del}_2$ in terms of p. Consider another relativized specification $\mathcal{S} \equiv (L, S{:}\, K \times L \to \mathrm{Prop})$ over a specification $B \equiv (K, B{:}\, K \to \mathrm{Prop})$ and a morphism $(u, f, g, p){:}\, \mathcal{S} \longrightarrow \mathcal{R}$, *i.e,*

$$\begin{aligned}
&(u{:}\, K \to I,\\
&\; f{:}\, \forall x{:}\, J.Bx \Rightarrow A(ux),\\
&\; g{:}\, K \to L \to J,\\
&\; h{:}\, \forall x{:}\, K.\forall y{:}\, L.Bx \Rightarrow S(x,y) \Rightarrow R(ux, gxy)).
\end{aligned}$$

Then, observe the following.

- the first-order deliverable given by $(u, f){:}\, (K, B) \longrightarrow (I, A)$ corresponds to a morphism $f{:}\, A \underset{(u)}{\longrightarrow} B$ in $\mathcal{E}$ over u:.
- the function $g{:}\, K \to L \to J$ corresponds to a morphism $g{:}\, K \times L \longrightarrow J$ in $\mathcal{B}$.
- to see what should correspond to h, first note that its type can be rewritten

  $$h{:}\, \forall z{:}\, K \times L.B(\pi_{0,KL}z) \wedge Sz \Rightarrow R(\langle u\pi_{0,KL}, g\rangle z).$$

  Then, h, combined with the function $\langle u\pi_{0,KL}, g\rangle{:}\, K \times L \to I \times J$, forms a first-order deliverables

  $$(\langle u\pi_{0,KL}, g\rangle, h){:}\, (K \times L, \lambda z{:}\, K \times L.B(\pi_{0,KL}z) \wedge Sz) \longrightarrow (I \times J, R).$$

  The domain of $(\langle u\pi_{0,KL}, g\rangle, h)$ is equal to a product in $\mathcal{E}_{K\times L}$ of two specifications $(K \times L, \lambda z{:}\, K \times L.B(\pi_{0,KL}z))$ and $(K \times L, S)$. The former corresponds to an object $B' \in \mathcal{E}_{K\times L}$ with a cartesian morphism $B' \underset{(\pi_{0,KL})}{\longrightarrow\!\!\!\!\!\!\!+\;\;} B$. Therefore, h corresponds to the morphism $h{:}\, B' \times_{K\times L} S \longrightarrow R$ of $\mathcal{E}$ over the base morphism $\langle u\pi_{0,KL}, g\rangle{:}\, K \times L \longrightarrow I \times J$.

  $$\begin{tikzcd}
  B & B' \arrow[l, "|" marking] & B' \times S \arrow[l, "\pi_{0,B'S}"'] \arrow[r, "\pi_{1,B'S}"] \arrow[d, "p"] & S \\
  K & K \times L \arrow[l, "\pi_{0,KL}"'] \arrow[r, equal] & K \times L \arrow[r, equal] & K \times L
  \end{tikzcd}$$

In order to express the composition of second-order deliverables, it is convenient to regard this construction $(B, S) \mapsto (B', S) \mapsto B' \times_{K \times L} S$ as a primitive construction.

Definition 7.4.2. Let $p: \mathcal{E} \longrightarrow \mathcal{B}$ be a fibration with $\mathcal{B}$ having binary products. Given $A \in \mathcal{E}_I$, $J \in \mathcal{B}$, and $R \in \mathcal{E}_{I \times J}$, define an object $A \cdot R \in \mathcal{E}_{I \times J}$ to be a representing object such that, for each $(X, u: K \longrightarrow I \times J)$ with $X \in \mathcal{E}_K$, there is a natural bijection

$$\mathcal{E}_u(X, A \cdot R) \cong \mathcal{E}_{\pi_{0,IJ} \circ u}(X, A) \times \mathcal{E}_u(X, R).$$

The chosen counit is written $(\pi_{0,AR}: A \cdot R \xrightarrow[(\pi_{0,IJ})]{} A,\ \pi_{1,AR}: A \cdot R \xrightarrow[(\mathrm{id}_{I \times J})]{} R)$ and has the following universal property.

$$\forall f: X \xrightarrow[(\pi_{0,IJ} u)]{} A. \forall g: X \xrightarrow[(u)]{} Y. \exists! h: X \xrightarrow[(u)]{} A \cdot R. \pi_{0,AR} h = f \wedge \pi_{1,AR} h = g$$

$$\begin{tikzcd}
 & X \arrow[dl, "f"'] \arrow[d, dashed, "h"] \arrow[dr, "g"] & \\
A & A \cdot R \arrow[l, "\pi_{0,AR}"'] \arrow[r, "\pi_{1,AR}"] & R
\end{tikzcd}
\quad \text{over} \quad
\begin{tikzcd}
 & K \arrow[dl, "\pi_{0,IJ} \circ u"'] \arrow[d, "u"] \arrow[dr, "u"] & \\
I & I \times J \arrow[l, "\pi_{0,IJ}"'] \arrow[r, "\mathrm{id}_{I \times J}"] & I \times J
\end{tikzcd}$$

We write the morphism h given by f and g as $\langle f, g \rangle$, and $\langle k\pi_{0,AR}, l\pi_{1,AR} \rangle$ as $k \cdot l$.

Now $B \cdot S$ is isomorphic to $B' \times_{K \times L} S$ in the above paragraph since we have

$$\begin{aligned}
\mathcal{E}_u(X, B' \times_{K \times L} S) &\cong \mathcal{E}_u(X, B') \times \mathcal{E}_u(X, S) \\
&\cong \mathcal{E}_{\pi_{0,KL} u}(X, B) \times \mathcal{E}_u(X, S) \\
&\cong \mathcal{E}_u(X, B \cdot S).
\end{aligned}$$

With this definition, we define the abstract version of second order deliverables as follows.

Definition 7.4.3. Given a fibration $p: \mathcal{E} \longrightarrow \mathcal{B}$ with $(-) \cdot (-)$ of Definition 7.4.2,

- the category $\mathcal{L} \equiv \mathcal{L}(\mathcal{E})$ of the *abstract* relativised specification has as objects the tuples $(A \in \mathcal{E}_I, J \in \mathcal{B}, R \in \mathcal{E}_{I \times J})$. A morphism of $\mathcal{L}$ from (A, J, R) to

(B, L, S) is a morphism $(f, g)\colon \pi_{0,AR} \longrightarrow \pi_{0,BS}$ of $\mathcal{E}^{\mathbf{2}}$, *i.e.*, a pair (f, g) that makes the following diagram commute.

$$\begin{tikzcd}
A \cdot R \arrow[r, "g"] \arrow[d, "\pi_{0,AR}"'] & B \cdot S \arrow[d, "\pi_{0,BS}"] \\
A \arrow[r, "f"] & B
\end{tikzcd}$$

The composition of morphisms is given by $(f, g) \circ (k, l) = (fk, gl)$.

- the functor $q\colon \mathcal{L} \longrightarrow \mathcal{E}$ sends an object (A, J, R) of $\mathcal{L}$ to A and a morphism (f, g) to f.
- a morphism in the fibre $\mathcal{L}_A$ is called an *abstract* second-order deliverable over the abstract specification A.

Lemma 7.4.4. The functor $q\colon \mathcal{L} \longrightarrow \mathcal{E}$ is a fibration.

Proof Given an object $(A \in \mathcal{E}_I, J \in \mathcal{B}, R \in \mathcal{E}_{I\times J})$ in $\mathcal{L}_A$ and a morphism $f\colon B \xrightarrow[(u)]{} A$ in $\mathcal{E}$ over $u\colon K \longrightarrow I$ in $\mathcal{B}$, let Q be an object in $\mathcal{E}_{K\times J}$ with a p-cartesian morphism $c\colon Q \xrightarrow[(u\times \mathrm{id}_J)]{+} R$ over $u \times \mathrm{id}_J\colon K \times J \longrightarrow I \times J$. Then, $(f, h)\colon (B, K, Q) \longrightarrow (A, J, R)$ in the diagram below is a q-cartesian morphism over f.

$$\begin{tikzcd}
Q \arrow[r, "c", "\shortmid"{marking}] & R \\
B \cdot Q \arrow[u, "\pi_{1,BQ}"] \arrow[r, dashed, "h"] \arrow[d, "\pi_{0,BQ}"'] & A \cdot R \arrow[u, "\pi_{1,AR}"'] \arrow[d, "\pi_{0,AR}"] \\
B \arrow[r, "f"] & A
\end{tikzcd}$$

$$\begin{tikzcd}
{} \arrow[d, mapsto, "p"] \\
{}
\end{tikzcd}$$

$$\begin{tikzcd}
I \times J \arrow[r, "u \times \mathrm{id}"] \arrow[d, equal] & K \times J \arrow[d, equal] \\
I \times J \arrow[r, "u \times \mathrm{id}"] \arrow[d, "\pi_{0,IJ}"'] & K \times J \arrow[d, "\pi_{0,KJ}"] \\
I \arrow[r, "u"] & K
\end{tikzcd}$$

In fact, given $(C \in \mathcal{E}_L, M \in \mathcal{B}, P \in \mathcal{E}_{L\times M}) \in \mathcal{L}_C$ and $C \xrightarrow[(v)]{g} B$ in $\mathcal{E}$, the inverse of $\mathcal{L}_f((C, M, P), (f, h))\colon \mathcal{L}_g((C, M, P), (B, K, Q)) \longrightarrow \mathcal{L}_{fg}((C, M, P), (A, J, R))$ is given by the following bijection ($k \mapsto \bar{k}$) of dotted morphisms (over specified $\mathcal{B}$ morphisms) that make each diagram commute: (writing w for $\langle v\pi_{0,LM}, \pi_{1,KJ}\circ pk\rangle$,)

$$
\begin{array}{c}
\begin{array}{c}
\begin{tikzcd}
C \cdot P \arrow[rr, dotted, "k", "((u \times \mathrm{id})w)"'] \arrow[d, "\pi_0"'] & & A \cdot R \arrow[d, "\pi_0"] \\
C \arrow[r, "g", "(v)"'] & B \arrow[r, "f", "(u)"'] & A
\end{tikzcd} \\ \hline
\begin{tikzcd} C \cdot P \arrow[r, dotted] & R \end{tikzcd} \\
((u \times \mathrm{id})w) \\ \hline
\begin{tikzcd} C \cdot P \arrow[r, dotted] & Q \end{tikzcd} \\
(w)
\end{array} \cong \\ \hline
\begin{tikzcd}
C \cdot P \arrow[r, dotted, "\bar{k}", "(w)"'] \arrow[d, "\pi_0"'] & B \cdot Q \arrow[d, "\pi_0"] \\
C \arrow[r, "g", "(v)"'] & B
\end{tikzcd}
\end{array} \cong
$$

∎

7.5 Kleisli construction of second-order deliverables

In this section, we study $\mathcal{L}$ in terms of Kleisli construction. The result is used in the next section to show a second-order deliverable as a first-order deliverable in an extended context. First, recall the following.

Lemma 7.5.1. Given an adjunction $\mathcal{A} \underset{G}{\overset{F}{\underset{\longrightarrow}{\overset{\longleftarrow}{\perp}}}} \mathcal{B}$, $\mathcal{B}$ is isomorphic to the Kleisli category $\mathcal{A}_{FG}$ for the comonad $FG\colon \mathcal{A} \longrightarrow \mathcal{A}$ if and only if the object part of G is a bijection.

Proof The necessity is clear from the usual construction of Kleisli categories described in *e.g.* Mac Lane [49]. To see that it is sufficient, we show that $\mathcal{B}$ has the same universal property as the Kleisli category $\mathcal{A}_{FG}$; namely, for any adjunction $\mathcal{A} \underset{L}{\overset{K}{\underset{\longrightarrow}{\overset{\longleftarrow}{\perp}}}} \mathcal{X}$ with $KL = FG$, there is a unique functor $H\colon \mathcal{B} \longrightarrow \mathcal{X}$ such that

$$\begin{tikzcd} \mathcal{B} \arrow[rr, dashed, "H"] \arrow[d, shift right=2, "F"'] \arrow[d, phantom, "\dashv"] & & \mathcal{X} \arrow[d, shift right=2, "K"'] \arrow[d, phantom, "\dashv"] \\ \mathcal{A} \arrow[u, shift right=2, "G"'] \arrow[rr, equal] & & \mathcal{A} \arrow[u, shift right=2, "L"'] \end{tikzcd}$$

is a morphism of adjunctions. First, the object part of G is determined by $HG = L$ as, for $A \in \mathcal{B}$, $HA = LG^{-1}A$. Second, the morphism part of H is determined by that of $KH = F$. To see this, let $\eta\colon \mathrm{id}_{\mathcal{X}} \longrightarrow LK$ and $\varepsilon\colon KL \longrightarrow \mathrm{id}_{\mathcal{A}}$ be the unit and the counit of the adjunction $K \dashv L$. Then, the following commutes, showing that $Hf = L\varepsilon_{G^{-1}B} \circ LFf \circ \eta_{HA}$ for any $f\colon HA \longrightarrow HB = LG^{-1}B$ in $\mathcal{B}$.

$$\begin{tikzcd} HA \arrow[r, "Hf"] \arrow[d, "\eta_{HA}"'] & LG^{-1}B \arrow[r, equal] \arrow[d, "\eta_{LG^{-1}B}"'] & LG^{-1}B \\ LKHA \arrow[r, "LKHf"', "= LFf"] & LKHB \arrow[ur, "L\varepsilon_{G^{-1}B}"'] & \end{tikzcd}$$

It is easy to check the functoriality of H, $KHB = FB$, and $HGf = Lf$. ∎

Let s(B) be the simple of the category $\mathcal{B}$ [34]. An object of s(B) is a pair of objects of $\mathcal{B}$. A morphism from (I, J) to (K, L) is a pair of morphisms

$$(f\colon I \longrightarrow K, g\colon I \times J \longrightarrow K \times L)$$

with $\pi_{0,KL} \circ g = f\pi_{0,IJ}$. Consider the following limit diagram in **Cat**.

$$\begin{tikzcd} & \mathcal{F} \arrow[dl, "q"'] \arrow[d] \arrow[dr] & \\ \mathcal{E} \arrow[d, "p"'] & \mathrm{s}(\mathcal{B}) \arrow[d, "U"] & \mathcal{E} \arrow[d, "p"] \\ \mathcal{B} & \mathcal{B} \times \mathcal{B} \arrow[l, "\pi_0"'] \arrow[r, "\pi_1"] & \mathcal{B} \end{tikzcd}$$

where the functor $U\colon \mathrm{s}(\mathcal{B}) \longrightarrow \mathcal{B} \times \mathcal{B}$ sends (I, J) to $(I, I \times J)$ and (f, g) to (f, g).

The category $\mathcal{F}$ has the same objects as $\mathcal{L}$ and a morphism from $(A \in \mathcal{E}_I, J \in \mathcal{B}, R \in \mathcal{E}_{I\times J})$ to $(B \in \mathcal{E}_K, L \in \mathcal{B}, S \in \mathcal{E}_{K\times L})$ is a pair $(f\colon A \longrightarrow B, g\colon R \longrightarrow S)$ of morphisms of $\mathcal{E}$ such that the following commutes.

$$\begin{tikzcd} I \times J \arrow[r, "pg"] \arrow[d, "\pi_0"'] & K \times L \arrow[d, "\pi_0"] \\ I \arrow[r, "pf"] & J \end{tikzcd}$$

The composition of morphisms is given componentwise. The functor q sends objects and morphisms to their first components.

Proposition 7.5.2. The functor $q: \mathcal{F} \longrightarrow \mathcal{E}$ is a fibration.

Proof It is immediate from a construction of the above limit diagram by pullbacks and the fact that a pullback preserves fibrations. More concretely, a cartesian morphism over $f: B \longrightarrow A$ with a codomain (A, J, R) is given by $(B \xrightarrow{f} A, R[pf \times \mathrm{id}_J] \longrightarrow\!\!\!\!\!\!\!+\;\; R)$. ∎

The fibre $\mathcal{F}_A$ over $A \in \mathcal{E}_I$ has as an object a pair $(J \in \mathcal{B}, R \in \mathcal{E}_{I\times J})$ and as a morphism from (K, Q) to (J, R) a morphism in $\mathcal{E}$ $f: Q \longrightarrow R$ with $\pi_{0,IJ} \circ pf = \pi_{0,IK}$. The composition and the identity are inherited from $\mathcal{E}$.

Define the functor $T: \mathcal{F} \longrightarrow \mathcal{F}$ by

$$\begin{array}{llcl} T: & \mathcal{F} & \longrightarrow & \mathcal{F} \\ & (A, J, R) & & (A, J, A \cdot R) \\ & \;\;\downarrow (f, g) & \mapsto & \;\;\downarrow (f, f \cdot g) \\ & (B, L, S) & & (B, L, B \cdot S). \end{array}$$

The morphism $f \cdot g$ is well-defined by the condition on a morphism in $\mathcal{F}$, and $(f, f \cdot g)$ is a morphism of $\mathcal{F}$ because $p(f \cdot g) = pg$.

The functor T is a part of the comonad (T, μ, δ) on $\mathcal{F}$. The counit $\mu: T \longrightarrow \mathrm{id}_{\mathcal{F}}$ is given by, for each (A, J, R) in $\mathcal{F}$,

$$\mu_{(A,J,R)} \equiv (\mathrm{id}_A, \pi_{1,AR}): (A, J, A \cdot R) \longrightarrow (A, J, R).$$

The comultiplication $\delta: T \longrightarrow TT$ is given by, for each (A, J, R),

$$\delta_{(A,J,R)} \equiv (\mathrm{id}_A, \langle \pi_{0,AR}, \mathrm{id}_{A \cdot R} \rangle \rangle): (A, J, A \cdot R) \longrightarrow (A, J, A \cdot (A \cdot R)).$$

The proof of the comonad laws can be easily checked.

Proposition 7.5.3. The category $\mathcal{L}$ is the Kleisli category for the comonad T. Moreover, each fibre $\mathcal{L}_A$ is the Kleisli category for the comonad T_A, the restriction of T to $\mathcal{F}_A$.

Proof By Lemma 7.5.1, it is enough to show that T can be decomposed as $\mathcal{F} \xrightarrow{G} \mathcal{L} \xrightarrow{F} \mathcal{F}$ with $F \dashv G$ and G being bijective on objects.

By definition, a morphism $(f, g)\colon (A, J, R) \longrightarrow (B, L, S)$ in $\mathcal{L}$ is also a morphism $(f, g)\colon (A, J, A \cdot R) \longrightarrow (B, L, B \cdot S)$ in $\mathcal{F}$. This gives the morphism part of the functor $F\colon \mathcal{L} \longrightarrow \mathcal{F}$ whose object part is that of T. In the converse direction, given a morphism $(f, g)\colon (A, J, R) \longrightarrow (B, L, S)$ in $\mathcal{F}$, the morphism $T(f, g) = (f, f \cdot g)\colon (A, J, R) \longrightarrow (B, L, S)$ is also a morphism in $\mathcal{L}$ with the same domain and codomain. Thus, the morphism part of T with the identity object part defines the functor $G\colon \mathcal{F} \longrightarrow \mathcal{L}$. Since obviously $T = FG$ and G is bijective on objects, it remains to show that $F \dashv G$. The bijection between g and $\bar{g}$ below gives the desired adjunction $\mathcal{F}(F(A, J, R), (B, L, S)) \cong \mathcal{L}((A, J, R), G(B, L, S))$.

$$\begin{tikzcd}
 & S \\
A \cdot R \arrow[ur, "g"] \arrow[r, "\bar{g}"] \arrow[d, "\pi_0"'] & B \cdot S \arrow[u, "\pi_1"'] \arrow[d, "\pi_0"] \\
A \arrow[r, "f"] & B
\end{tikzcd}
\quad \text{such that} \quad
\begin{tikzcd}
pA \times J \arrow[r, "pg = p\bar{g}"] \arrow[d, "\pi_0"'] & pB \times L \arrow[d, "\pi_0"] \\
pA \arrow[r, "pf"] & pB
\end{tikzcd}$$

It is also clear that the unit

$$\eta_{(A,J,R)}\colon \begin{tikzcd}
A \cdot R \arrow[r, "\langle \pi_0, \mathrm{id} \rangle"] \arrow[d, "\pi_0"'] & A \cdot (A \cdot R) \arrow[d, "\pi_0"] \\
A \arrow[r, equal] & A
\end{tikzcd}$$

in $\mathcal{F}$ is over id_A and F and G restrict to an adjunction between $\mathcal{F}_A$ and $\mathcal{L}_A$. ∎

Note, however, F and G do not form a fibred adjunction (Definition 1.2.12 of Hermida's thesis [34]), since G does not preserve cartesian morphisms. (*cf.* Section 6.2.7 of *ibid.*, the adjunction described there was not fibred with the same reason; he has since proved a precise improved statement.)

7.6 The family construction of second-order deliverables

In this section, we exploit the result of the previous section to categorically analize the view that a second-order deliverable is a first-order deliverable in an extended context. McKinna [52] gave a syntactic construction, called *family construction*, of second-order deliverables from such first-order deliverables.

Proposition 7.6.1. (McKinna [52])

Suppose we have $\Gamma \vdash s{:}\,\mathrm{Type}$, $\Gamma \vdash S{:}\, s \to \mathit{Prop}$, and two relativized specifications (t, T) and (u, U) over (s, S). Then, to give a second-order deliverable over (s, S) from (t, T) to (u, U) is equivalent to to give $\Gamma, x{:}\, s \vdash f_x{:}\, t \to u$ and a first-order deliverable under the context $\Gamma, x{:}\, s, h{:}\, Sx$ from a specification (t, Tx) to (u, Ux) with its first component being f_x.

We show that this correspondence is a consequence of the analysis given in the previous section. This is to expand the last remark in Section 4.3.4 of Hermida's thesis [34].

7.6.1 Polynomial categories

In Chapter 6, we saw an extension of context can be modelled by construction of a Hom-object in the base of a locally small fibration. An alternative, more external interpretation is possible with the concept of polynomial categories [44,34]. Given a category $\mathcal{A}$ with finite products, a *polynomial* category category $\mathcal{A}[x{:}\, A]$ with $A \in \mathcal{A}$ is obtained by freely adjoining a new morphism ('indeterminate') $x{:}\, 1 \longrightarrow A$ to $\mathcal{A}$ [44]. The inclusion of $\mathcal{A}$ into $\mathcal{A}[x{:}\, A]$, hence $\mathcal{A}[x{:}\, A]$, can be characterised by the following universal property.

Proposition 7.6.2. (Prop. 5.1 of Lambek & Scott [44])

Let $\mathcal{A}$ be a category with finite products, $A \in \mathcal{A}$, and $H_x: \mathcal{A} \longrightarrow \mathcal{A}[x: A]$ the inclusion into the polynomial category. For any product preserving functor $F: \mathcal{A} \longrightarrow \mathcal{B}$ and any morphism $h: 1 \longrightarrow FA$ in $\mathcal{B}$, there exists a unique product preserving functor $\langle F, h \rangle: \mathcal{A}[x: A] \longrightarrow \mathcal{B}$ with $\langle F, h \rangle H_x = F$ and $Fx = h$.

Proof See [44]. ▮

The following is also well known.

Proposition 7.6.3. The polynomial category $A[x: A]$ is isomorphic to the Kleisli category $\mathcal{A}_{A\times(-)}$ for the comonad $A \times (-): \mathcal{A} \longrightarrow \mathcal{A}$ with the counit $\varepsilon_B = \pi_1: A \times B \longrightarrow B$ and the comultiplication $\delta_B: \langle \pi_0, \mathrm{id}_{A\times B} \rangle: A \times B \longrightarrow A \times (A \times B)$.

The proof found in [44] shows that $\mathcal{A}_{A\times(-)}$ has the universal property of $\mathcal{A}[x: A]$ in Proposition 7.6.2. The proof is based on a concrete description of $\mathcal{A}_{A\times(-)}$.

The category $\mathcal{A}_{A\times(-)}$ can be described as follows. The objects are the same as $\mathcal{A}$, a morphism f from X to Y is a that of $\mathcal{A}$ from $A \times X$ to $A \times Y$ with $\pi_{0,AY} f = \pi_{0,AX}$; the composition and the identity are inherited from $\mathcal{A}$. This description differs a little from the one given in [44], but the equivalence is immediate. The Kleisli resolution $\mathcal{A} \underset{G}{\overset{F}{\leftrightarrows}} \mathcal{A}[x: A]$ (with $F \dashv G$) is given by $FX = A \times X$, $Ff = f$, $GX = X$, and $Gf = A \times f$.

We show a different, conceptually clearer proof of Proposition 7.6.3 in order to apply this to the construction of abstract second-order deliverables. This is a special case of Hermida's general 2-categorical treatment [34]. The proof is given in two parts.

Proposition 7.6.4. Let $\mathcal{A} \underset{G}{\overset{F}{\leftrightarrows}} \mathcal{B}$ be an adjunction with G being bijective on objects, η the unit, and $H: \mathcal{A} \longrightarrow \mathcal{C}$, $K, K': \mathcal{B} \longrightarrow \mathcal{C}$ functors with $KG = H$ and $K'G = H$. Then, $K'\eta G = K\eta G$ implies $K = K'$.

Proof Let ε be the counit of the adjunction. The morphism parts of K and K' agree because the image of ηG by K (K') determines the rest of the morphism part $K_{GX,GY}$ as follows.

$$
\begin{tikzcd}
 & \mathcal{B}(GX,GY) \arrow[dl, "F_{GX,GY}"'] \arrow[dd, "\cong"'] \arrow[rr, "K_{GX,GY}"] & & \mathcal{C}(HX,HY) \\
\mathcal{A}(FGX,FGY) \arrow[dr, "\varepsilon_Y \circ (-)"'] & & \mathcal{B}(GFGX,GY) \arrow[ul, "(-)\circ\eta_{GX}"'] \arrow[r, "K_{GFGX,GY}"] & \mathcal{C}(HFGX,HY) \arrow[u, "(-)\circ K\eta_{GX}"'] \\
 & \mathcal{A}(FGX,Y) \arrow[ur, "G_{FGX,Y}"] \arrow[urr, "H_{FGX,Y}"'] & &
\end{tikzcd}
$$

∎

This is a special case of Street's 2-categorical analysis ([76], also see [34]). In his term, $\mathcal{B}$ classifies those co-op-algebras (oplax cocones in [34]) for the comonad FG and ηG is the universal such.

Corollary 7.6.5. In the above situation, further suppose $\mathcal{A}$ has finite products preserved by H, and (F,G) is the Kleisli resolution of the $A\times(-)$ comonad. Then, $K\eta_{G1} = K'\eta_{G1}$ implies $K = K'$.

Proof This follows from that $K\eta_{GX}$ $(K'\eta_{GX})$ is determined by $K\eta_{G1}(= K'\eta_{G1})$.

$$
\begin{tikzcd}
 & & KGX \\
KGX \arrow[r, "K\eta_{GX}"'] \arrow[urr, "\mathrm{id}"] \arrow[d, "KG!_X"'] & KGFGX \arrow[r, equal] & H(A\times X) \arrow[u, "H\pi_1 = KG\pi_1"'] \arrow[d, "H(A\times !_X)"'] \\
KG1 \arrow[r, "K\eta_{G1}"'] & KGFG1 \arrow[r, equal] & H(A\times 1)
\end{tikzcd}
$$

The triangle commutes because $\pi_1\colon A\times(-) \longrightarrow \mathrm{id}_{\mathcal{A}}$ is the counit of the resolution. This determines $K\eta_{GX}$ since the right column is a product cone. ∎

Proof (of Proposition 7.6.3) Corollary 7.6.5 shows the uniqueness part of Proposition 7.6.2. The same two proofs of Corollary 7.6.5 and Proposition 7.6.4 also show the existence; a morphism $1 \longrightarrow A$ gives rise to a natural transformation $KG \longrightarrow H(A\times(-))$, which in tern gives rise to a morphism part of a functor from $\mathcal{A}$ to $\mathcal{C}$ with its object part being $X \mapsto HG^{-1}X$. It is also easy to check that this functor preserves products in $\mathcal{A}[x\colon A]$. ▮

7.6.2 Family construction

Now we turn to our abstract second-order deliverables with the Kleisli resolution $\mathcal{F}_A \underset{G_A}{\overset{F_A}{\rightleftarrows}} \mathcal{L}_A$ (with $\perp$). First note that this is the same as $\mathcal{A} \rightleftarrows \mathcal{A}[x\colon A]$, but with the difference between $A\cdot(-)$ and $A \times (-)$. However, Corollary 7.6.5 remains valid when one replaces $A \times (-)$ by $A\cdot(-)$. To see this, consider $A \in \mathcal{E}_I$ and $(J, R \in \mathcal{E}_{I\times J}) \in \mathcal{F}_A$. The terminal object of $\mathcal{F}_A$ is given by $(1 \in \mathcal{B}, 1 \in \mathcal{E}_{I\times 1})$. The diagram $(1, A\cdot 1) \xleftarrow{A\cdot !} (J, A\cdot R) \xrightarrow{\pi_1} (J, R)$ is a product cone in $\mathcal{F}_A$. So the proof of Corollary 7.6.5 still applies.

So, we have the following modified version of Proposition 7.6.3.

Proposition 7.6.6. Given $A \in \mathcal{E}_I$, $\mathcal{L}_A$ is isomorphic to the polynomial category $\mathcal{F}_A[x\colon (1, A\cdot 1)]$. ▮

Using the description of F_A, one can spell this out as follows: to give a morphism in $\mathcal{L}_A$ from (J, Q) to (K, R) is equivalent to to give, with an indeterminate $x\colon 1 \longrightarrow A$, a morphism $f\colon Q \longrightarrow R$ in $\mathcal{E}$ such that $\pi_{0,IK}pf = \pi_{0,IJ}$. This precisely rephrases Proposition 7.6.1; the correspondence of letters is: $I \mapsto s$, $J \mapsto t$, $K \mapsto u$, $A \mapsto (s, S)$, $x \mapsto (x, h)$, $R \mapsto (t, T)$, and $Q \mapsto (u, U)$.

One can further continue this analysis in a fibred setting with varying A, after the analysis given in Hermida's thesis [34]. One needs some modification to it to account for the condition $\pi_{0,IK} \circ pf = \pi_{0,IJ}$ that is not found in his analysis.

Chapter 8

Conclusions and further work

In this thesis, we achieved two objectives. First, we provided category theorists with a computer checked language to describe arrows of categories of their choice with extra structure. That was done in order to support the proofs of coherence theorems in particular, and of any category theoretic proof of a syntactic nature in general. Then we provided what we called a categorical framework for type theory, our computer checked language providing formalisation of that framework on a computer.

The first objective extends the work of Rydeheard and Burstall, who formalised category theoretic constructions in ML, in that we provided a more uniform treatment, giving a general recipe for how to formalise universal constructions. That is of interest not only to category theorists, but also to those computer scientists who may attempt to build programming languages based upon categorical constructions, such as the Categorical Abstract Machines (Curien *et. al.* [18], Ritter [70]), Categorical Programming Language (Hagino [30]), and CHARITY (Cockett and Spencer [17,75]).

A common approach to this involves the classification of structure on categories as equational structure, or more generally, as essentially algebraic structure. We explained why that is somewhat awkward, both conceptually and computationally. We therefore developed an alternative approach, namely the classification of categorical structure as universal structure, universality being the single most important category theoretic concept. This incorporated very many examples of category theoretic concepts, in particular those of greatest interest to computer scientists, such as cartesian closed structure, fibrations with extra structure, limits, colimits, and natural numbers objects.

We gave a formal definition of universal structure, developed the concept abstractly, and gave many examples. We further used it as the basis for a computer checked language in the proof system LEGO. We needed a proof system as our specific aim was to support category theoretic proof. We chose LEGO as it was both familiar to us and convenient.

We then proceeded to investigate the use of category theory in computer science. One of the greatest uses, perhaps the deepest one, is the use of category theory as a semantics for type theory. Many instances of this appear in the literature, the best known being the relationship between the simply typed lambda calculus and cartesian closed categories.

Type theory has been widely used as a foundation for computer science. Programming languages typically have an underlying type theory, and logics typically have a corresponding type theory, given by the Curry-Howard isomorphism. So, for both the study of programming languages and for the study of logics, in particular logics for reasoning about programming languages, one often passes to type theory. For instance, the underlying type theory of the proof system LEGO is the Extended Calculus of Constructions. Categories with structure provide a semantics for type theory.

We therefore gave a unified treatment of the relationship between a general and subtle class of type theories and categories with extra structure. The type system $\mathcal{T}F\iota$ was devised to describe locally small fibrations with particular structure, and its equivalence to the type system $w\mathcal{M}L$ with weak dependent coproduct types were shown in detail. The equivalence between the type system $\mathcal{M}L$ with strong coproducts and $\mathcal{T}F\iota\sigma$ was similarly shown. That extra structure needed was indeed universal structure, so our computer checked language can provide a formalisation of our categorical framework on a computer. Finally, we illustrated this framework and the corresponding universal structure by a study of the fibration of deliverables. Our analysis gave a categorical understanding of the syntactic "family construction".

We now turn to further work. We seek to develop the work of this thesis in several ways. First, we should like to extend our abstract study of universal structure so that we study not only universal structures on categories, but more generally, universal structures on fibrations. As we have illustrated, fibrations are a fundamental concept in category theory, in particular for providing the semantics of type theory, especially for modelling dependent types. They are also used to model parametricity, and are in fact equivalent to a study of logical relations. Bénabou goes so far in his abstract development of fibrations in analogy with category theory as to actually use the word "category" for fibrations [8]. Categories in fact may be seen as special fibrations in several ways.

We had hoped to achieve that aim for this thesis but it proved to be too ambitious. However, having developed the concept of universal structure for ordinary categories, it is now much clearer how to extend it to fibrations. What we need is a Yoneda structure on the 2-category of fibrations. The 2-category of fibrations is studied extensively in Hermida's thesis [34]; Yoneda structures are an established category theoretic notion that allows an abstract account of representability [81, 80,77]. It seems likely that the 2-category of fibrations does posses a Yoneda structure and hence allows us to give an account of representability, and we hope to investigate that further. The formulation of fibrations as bicategory enriched categories [9,26,79] should also be investigated, since this seems to provide a natural and concise way to treat universal structure across different fibres. For example, locally small fibrations with $\mathcal{B}$-products and $\mathcal{B}$-coproducts correspond to such enriched categories with cotensors and tensors, respectively.

Another direction is to investigate consequences of our analysis in terms of Lawvere's [45] generalised logic. From this view point, for example, operations given by universal structure have a strong connection with the description operator.

We hope to present our proposed computer checked language for categories with structure to category theorists working on coherence problems to see their reaction. There are many difficult, highly technical coherence problems being addressed by category theorists nowadays, for instance in the study of higher order categories, such as tricategories, quadricategories, and beyond [66]. The very definition of tricategory requires several pages and the higher order categories are so complex that they have not yet had a complete formal definition. It seems obvious that computer checking could provide a valuable aid to these endeavours. In fact, that has already been achieved to a certain extent by use of programs to calculate the cocycle equations for higher order categories and by use of PROLOG in connection with n-categories and pasting problems [10]. We hope that our investigation may help in this regard, and we hope to further develop our language after consultation with category theorists about their specific needs. We also hope to develop this language based upon the needs of computer scientists working with category theory, continuing to develop the programme outlined by Rydeheard and Burstall [71].

To achieve this goal, we need to account for monoidal structure. In general, monoidal structure is not universal structure, but many instances of monoidal structure are defined by universal properties, typically by colimits. We hope to capture common aspects of such instances in our abstract framework.

Finally, we seek to develop our categorical framework for type theory. There are several issues that we have not yet addressed. For instance, our framework only addresses those functors that strictly preserve structure, such as functors that send an assigned terminal object to an assigned terminal object as explained in [65,66]. These are too restrictive for category theorists even in modelling logic. Moreover, the abstract framework for our computer checked language can already account for the more general functors that are much more common in category theory, such as those functors that preserve a terminal object in the usual sense, meaning up to coherent isomorphism. In "Why Tricategories?" [65,66]there is an informal account not only of the importance of this endeavour, but also how one might address it. We plan to provide the details, thus extending the relationship we have provided between type theory and categories with structure. In contrast to "Why Tricategories?", our classification is in terms of universal structure, rather than in terms of operations and equations, and we believe that this will make the relationship easier to describe.

Less directly arising from this thesis, we have gained much deeper understanding of the relationship between categories with structure and type theories, and how to formalise the proofs of theorems about categories with structure. We hope to bring this knowledge to bear on a treatment of a semantics of particular feature of programming languages and their underlying type theories. Specifically, we plan to analyse the semantics of local variables using the structures we have developed. The study of local variables is intimately linked to the study of parametricity, and hence to the study of fibrations [88,89,69,34,37]. We plan to do this based upon the work of O'Hearn and Tennent [87,88,57,58,89,56] and in collaboration with them.

Appendix A

Proofs of Chapter 6

A.1.1 Proof of Proposition 6.4.2

To prove soundness of a translation $[\![-]\!]_{\mathcal{T}F\lambda}$, the following *substitution lemma* is needed.

Lemma A.1.7. Let $[\![-]\!]_{\mathcal{T}F\lambda}$ be a translation from a λP1 theory T_G to a $\mathcal{T}F\lambda$ theory T_F. Then the equality judgements

$$[\![\vec{y}\colon \vec{G}]\!] \vdash [\![H[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}} = [\![H]\!]^{\vec{x}:\vec{K}}[u_n]\colon \mathbf{Type}$$

and

$$[\![\vec{y}\colon \vec{G}]\!] \vdash [\![C[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}} = [\![C]\!]^{\vec{x}:\vec{K}}[u_n]\colon \mathbf{1} \Rightarrow [\![H]\!]^{\vec{x}:\vec{K}}[u_n]$$

are derivable in T_F, where u_n is the context mapping defined in Definition 6.2.1 and $\vec{x}\colon \vec{K} \vdash H\colon \mathbf{Type}$, $\vec{x}\colon \vec{K} \vdash C\colon H$, and $\vec{y}\colon \vec{G} \vdash A_i\colon K_i[x_1, \cdots x_{i-1}/A_1, \cdots A_{i-1}]$ with $(1 \leq i \leq n)$ are derivable in T_G.

Proof By induction on derivations in T_G. The case $H \equiv \Pi x_{n+1}\colon K_{n+1}.H'[x_{n+1}]$, for example, is checked as follows.

Let v_{n+1} be the context mapping defined by

$$\begin{array}{lcl} v_0 & = & !_{[\![\vec{y}:\vec{G}, y_{n+1}:K_{n+1}[\vec{x}/\vec{A}]]\!]} \\ v_i & = & \prec v_{i-1}, [\![A_i]\!]^{\vec{y}:\vec{G}, y_{n+1}:K_{n+1}[\vec{x}/\vec{A}]} \succ \\ & & (1 \leq i \leq n) \\ v_{n+1} & = & \prec v_n, \chi_{\mathbf{1} \Rightarrow [\![K_{n+1}[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}, y_{n+1}:K_{n+1}[\vec{x}/\vec{A}]}} \succ . \end{array}$$

From the equations

$$[\![A_i]\!]^{\vec{y}:\vec{G}, y_{n+1}:K_{n+1}[\vec{x}/\vec{A}]} = [\![A_i]\!]^{\vec{y}:\vec{G}}[\mathbf{d}_{\mathbf{1} \Rightarrow [\![K_{n+1}[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}}}],$$

one gets

$$v_{n+1} = \prec u_n \circ \mathbf{d}_{\mathbf{1} \Rightarrow [\![K_{n+1}[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}}}, \chi_{\mathbf{1} \Rightarrow [\![K_{n+1}[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}, y_{n+1}:K_{n+1}[\vec{x}/\vec{A}]}} \succ .$$

Then it follows that

$$\begin{array}{l} [\![(\Pi x_{n+1}\colon K_{n+1}.H'[x_{n+1}])[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}} \\ \quad = \quad [\![\Pi y_{n+1}\colon K[\vec{x}/\vec{A}].H'[\vec{x}/\vec{A}, x_{n+1}/y_{n+1}]]\!]^{\vec{y}:\vec{G}} \\ \quad = \quad \Pi_{\mathbf{1} \Rightarrow [\![K[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}}}.[\![H'[\vec{x}/\vec{A}, x_{n+1}/y_{n+1}]]\!]^{\vec{y}:\vec{G}, y_{n+1}:K[\vec{x}/\vec{A}]} \\ \quad = \quad \Pi_{\mathbf{1} \Rightarrow [\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[u_n]}.[\![H']\!]^{\vec{x}:\vec{K}, x_{n+1}:K_{n+1}}[v_{n+1}] \\ \quad = \quad \Pi_{\mathbf{1} \Rightarrow [\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[u_n]}.([\![H']\!]^{\vec{x}:\vec{K}, x_{n+1}:K_{n+1}} \\ \qquad\qquad [\prec u_n \circ \mathbf{d}_{\mathbf{1} \Rightarrow [\![K_{n+1}[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}}}, \chi_{\mathbf{1} \Rightarrow [\![K_{n+1}[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}}} \succ]) \\ \quad = \quad (\Pi_{\mathbf{1} \Rightarrow [\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.[\![H']\!]^{\vec{x}:\vec{K}, x_{n+1}:K_{n+1}})[u_n] \\ \quad = \quad [\![\Pi x_{n+1}\colon K_{n+1}.H'[x_{n+1}]]\!][u_n]. \end{array}$$

Other cases can be checked in a similar way. ∎

Proof (of Proposition 6.4.2)
By induction on derivations in T_G.

If the last rule used is one of the structural inference rules or one of the equality inference rules (reflexivity), (symmetry), (transitivity), (replacement) and (compatibility), then an easy induction step gives the desired judgement in T_F. As for (congruence1) and (congruence2), induction steps follow from Lemma A.1.7.

What remains is to check the equality inference rules (β) and (η). For (β), as a special case of Lemma A.1.7, one has

$$[\![A[x_{n+1}/B']]\!]^{\vec{x}:\vec{K}} = [\![A]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![B']\!]^{\vec{x}:\vec{K}}\succ].$$

Then, the inference rule (ETM10) gives

$$\begin{aligned}
&[\![(\lambda x_{n+1}\colon K_{n+1}.(A))B']\!] \\
&= (\varphi^{[\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}})[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![B']\!]^{\vec{x}:\vec{K}}\succ]\circ\lambda_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.([\![A]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}) \\
&= (\varphi^{[\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}\circ(\lambda_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.([\![A]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}))[\mathbf{d}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}]) \\
&\qquad [\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![B']\!]^{\vec{x}:\vec{K}}\succ] \\
&= [\![A]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![B']\!]^{\vec{x}:\vec{K}}\succ] \\
&= [\![A[x_{n+1}/B']]\!]^{\vec{x}:\vec{K}}
\end{aligned}$$

as desired. To prove that (η) is soundly translated, the Beck-Chevalley condition (ETY3) and (ETY12) are required as well as (ETM11). With (ETY3), one gets

$$\begin{aligned}
&[\![\Pi x_{n+1}\colon K_{n+1}.H]\!]^{\vec{x}:\vec{K},z:K_{n+1}} = [\![\Pi x_{n+1}\colon K_{n+1}.H]\!]^{\vec{x}:\vec{K}}[d_1] \\
&\quad = (\Pi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.[\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}})[d_1] \\
&\quad = \Pi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[d_1]}.[\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}[\prec d_1\circ d_2, \chi_2\succ],
\end{aligned}$$

where $d_1 = \mathbf{d}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}$, $d_2 = \mathbf{d}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[d_1]}$, and $\chi_2 = \chi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[d_1]}$. The rule (η) is checked as follows.

$$\begin{aligned}
&[\![\lambda z\colon K_{n+1}.(B)z]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}} \\
&\quad = \lambda_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[d_1]}.(\phi_1[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, [\![z]\!]^{\vec{x}:\vec{K},z:K_{n+1}}\succ]\circ[\![B]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}) \\
&\quad = \lambda_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[d_1]}.(\phi_2[\prec d_1\circ d_2, \chi\succ][\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]}, \chi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}\succ] \\
&\qquad \circ[\![B]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}) \\
&\quad = \lambda_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[d_1]}.(\phi_2[\prec d_1, \chi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}\succ]\circ[\![B]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}) \\
&\quad = \lambda_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[d_1]}.(\phi_2\circ[\![B]\!]^{\vec{x}:\vec{K}}[d_1]) \\
&\quad = [\![B]\!]^{\vec{x}:\vec{K}}
\end{aligned}$$

where $\phi_1 = \varphi^{[\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}[\prec d_1\circ d_2,\chi\succ]}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[d_1]}$ and $\phi_2 = \varphi^{[\![H]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}$. ∎

A.1.2 Proof of Proposition 6.4.4

To show the faithfulness of $[\![-]\!]_{\mathcal{F}(T_G)}$, some preparation is needed. The technical difficulty is that if we naively translate a term $A\colon H$ of T_G to that of $\mathcal{F}(T_G)$ by $[\![-]\!]_{\mathcal{F}(T_G)}$ and then translate it back to T_G, the resulting term has the type $\mathbf{1} \supset [\![[\![H]\!]]\!]$ rather than $[\![[\![H]\!]]\!]$.

Proposition A.1.8. (Definition-Proposition)
For each well-formed type $\vec{x}\colon \vec{K} \vdash H\colon \mathbf{Type}$ of T_G, the following well-formed type and terms are defined.

- $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash \hat{H}\colon \mathbf{Type}$, where $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{} = z_1\colon \mathbf{1}\supset(K_1)\hat{}, \cdots, z_n\colon \supset K_n$.
- $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash \hat{A}\colon \mathbf{1}\supset(H)\hat{}$, where $\vec{x}\colon \vec{K}\vdash A\colon H$.
- $\vec{x}\colon \vec{K}\vdash \check{B}\colon H$, where $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash B\colon \mathbf{1}\supset(H)\hat{}$.

These satisfy the condition that $\vec{x}\colon \vec{K}\vdash \hat{A}\check{} = A\colon H$ and $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash \check{B}\hat{} = B\colon \mathbf{1}\supset(H)\hat{}$ are derivable in T_G, that is, the maps $A \mapsto \hat{A}$, $B \mapsto \check{B}$ are bijective.

Proof The definition is given by induction on the derivation of $\vec{x}\colon \vec{K}\vdash H\colon \mathbf{Type}$. We use the notation in the corresponding inference rules of λP1.

- (constant1)
  - For $\vec{x}\colon \vec{K}\vdash \mathrm{F}\colon \mathbf{Type}(\vec{x})$, $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash \hat{\mathrm{F}}[\vec{z}] \equiv \mathrm{F}((\vec{z})\check{})\colon \mathbf{Type}$ where $(\vec{z})\check{} = \check{z}_1, \check{z}_2, \cdots, \check{z}_n$.
  - For $\vec{x}\colon \vec{K}\vdash A[\vec{x}]\colon \mathrm{F}(\vec{x})$, $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash \hat{A}[\vec{z}] \equiv \lambda v\colon \mathbf{1}.(A[(\vec{z})\check{}])\colon \mathbf{1}\supset\widehat{\mathrm{F}}$.
  - For $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash B[\vec{z}]\colon \mathbf{1}\supset\widehat{\mathrm{F}}$, $\vec{x}\colon \vec{K}\vdash \check{B}[\vec{x}] \equiv B[\hat{\vec{x}}](*)\colon (\vec{x})$
- (subst1)
  - For $\vec{y}\colon \vec{G}\vdash \mathrm{F}(\vec{A})\colon \mathbf{Type}$, $\vec{w}\colon \mathbf{1}\supset(\vec{G})\hat{}\vdash (\mathrm{F}(\vec{A}))\hat{} \equiv \mathrm{F}(\vec{A}[\vec{y}/(\vec{w})\check{}])\colon \mathbf{Type}$.
  - For $\vec{y}\colon \vec{G}\vdash C\colon \mathrm{F}(\vec{A})$, $\vec{w}\colon \mathbf{1}\supset(\vec{G})\hat{}\vdash \hat{C}[\vec{w}] \equiv \lambda v\colon \mathbf{1}.(C[\vec{y}/(\vec{w})\check{}])\colon \mathbf{1}\supset(\mathrm{F}(\vec{A}))\hat{}$.
  - For $\vec{w}\colon \mathbf{1}\supset(\vec{G})\hat{}\vdash B\colon \mathbf{1}\supset(\mathrm{F}(\vec{A}))\hat{}$,
  $$\vec{y}\colon \vec{G}\vdash \check{B}[\vec{y}] \equiv (B[\vec{w}/\hat{\vec{y}}])(*)\colon \mathrm{F}(\vec{A}).$$
- (Π F)

– For $\vec{x}\colon \vec{K} \vdash \Pi x_{n+1}\colon K_{n+1}.H\colon \mathbf{Type}$,

$$\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash(\Pi x_{n+1}\colon K_{n+1}.H)\hat{}\equiv \Pi z_{n+1}\colon \mathbf{1}\supset\widehat{K_{n+1}}.\hat{H}[\vec{z}, z_{n+1}]\colon \mathbf{Type}$$

– For $\vec{x}\colon \vec{K} \vdash A\colon \Pi x_{n+1}\colon K_{n+1}.H$,

$$\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{} \vdash \hat{A}[\vec{z}] \equiv \lambda v\colon \mathbf{1}.(\lambda z_{n+1}\colon \mathbf{1}\supset\widehat{K_{n+1}}.((A[\vec{x}/(\vec{z})\check{}](\check{z_{n+1}}))\hat{}(*)))$$
$$\colon \mathbf{1}\supset(\Pi x_{n+1}\colon K_{n+1}.H)\hat{}.$$

– For $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash B\colon \mathbf{1}\supset(\Pi x_{n+1}\colon K_{n+1}.H)\hat{}$,

$$\vec{x}\colon \vec{K} \vdash \check{B}[\vec{x}]\equiv \lambda x_{n+1}\colon K_{n+1}.((\lambda v\colon \mathbf{1}.(B[\vec{z}/\hat{\vec{x}}](*)(\hat{x_{n+1}})))\check{})$$
$$\colon \Pi x_{n+1}\colon K_{n+1}.H.$$

- For other type forming inference rules such as (weakening), the definition is obvious.

▮

Definition A.1.9. $[\![-]\!]_G$ is a pretranslation from $\mathcal{F}(T_G)$ to T_G given by $[\![\tilde{\mathrm{F}}]\!]_G = \hat{\mathrm{F}}$ and $[\![\tilde{\mathrm{C}}]\!]_G = \hat{\mathrm{C}}$ for constants.

To show that $[\![-]\!]_G$ is indeed a translation, the next proposition suffices.

Proposition A.1.10. The iterated translation $[\![[\![-]\!]_{\mathcal{F}(T_G)}]\!]_G$ from T_G to itself coincides with the maps $A \mapsto \hat{A}$ on terms up to equality in T_G. More precisely, for every $\vec{x}\colon \vec{K}\vdash A\colon H$ derivable in T_G, the equality judgement

$$z_1\colon \mathbf{1}\supset\widehat{K_1}, \cdots, z_n\colon \mathbf{1}\supset\widehat{K_n}\vdash[\![[\![A]\!]_{\mathcal{F}(T_G)}]\!]_G = \hat{A}\colon \mathbf{1}\supset\widehat{H}$$

is derivable in T_G. ▮

Proof (of Proposition 6.4.4)
Applying $[\![-]\!]_G$, one gets a derivable judgement

$$z_1\colon [\![[\![K_1]\!]_{\mathcal{F}(T_G)}]\!]_G, \cdots, z_n\colon [\![[\![K_n]\!]_{\mathcal{F}(T_G)}]\!]_G\vdash [\![[\![A]\!]_{\mathcal{F}(T_G)}]\!]_G = [\![[\![A']\!]_{\mathcal{F}(T_G)}]\!]_G$$
$$\colon \mathbf{1}\supset[\![[\![H]\!]_{\mathcal{F}(T_G)}]\!]_G$$

of T_G. By the previous proposition, this is equivalent to

$$z_1 \colon \mathbf{1} \supset \widehat{K_1}, \cdots, z_n \colon \mathbf{1} \supset \widehat{K_n} \vdash \hat{A} = \widehat{A'} \colon \mathbf{1} \supset \widehat{H}.$$

So, $A = A'$. ▮

A.1.3 Proof of Proposition 6.4.6

Proof By induction on derivations in $\mathcal{T}F\lambda$.
For structural judgements, easy induction steps show that $[\![-]\!]_{\lambda\mathrm{P1}}$ results well-formed expressions of T_G with desired typing.

For an equality judgement, if the last rule used in the derivation is one of (symmetry), (reflexivity), (transitivity), (replacement), and (compatibility) rules of $\mathcal{T}F\lambda$, then corresponding rule of λP1 combined with induction hypotheses gives the desired equality judgement of T_G.

Hence, it is sufficient to show that, for each equality inference rule

$$\frac{J_1 \quad J_2 \quad \cdots \quad J_n}{J}$$

of $\mathcal{T}F$, the translation of J is derivable from those of J_i's in T_G.

$$(ECM1) \quad \begin{aligned}[\![\mathbf{id}_\Gamma \circ u]\!] &= (v_{K_1L_1}[v_{K_1L_1}/B_1, \cdots, v_{K_nL_n}/B_n], \cdots, \\ &\qquad v_{K_nL_n}[v_{K_1L_1}/B_1, \cdots, v_{K_nL_n}/B_n]) \\ &= (B_1, \cdots, B_n) = [\![u]\!]\end{aligned}$$

$(ECM2)$ (similar to $(ECM1)$)

$$
\begin{array}{lll}
(ECM3) & [\![(u\circ v)\circ w]\!] & \\
 & = & (B_1[v_{G_1H_1}/C_1,\cdots,v_{G_mH_m}/C_m][v_{I_1J_1}/D_1,\cdots,v_{I_lJ_l}/D_l],\cdots, \\
 & & \quad B_n[v_{G_1H_1}/C_1,\cdots,v_{G_mH_m}/C_m][v_{I_1J_1}/D_1,\cdots,v_{I_lJ_l}/D_l]) \\
 & = & (B_1[v_{G_1H_1}/C_1[v_{I_1J_1}/D_1,\cdots,v_{I_lJ_l}/D_l],\cdots, \\
 & & \qquad v_{G_mH_m}/C_m[v_{I_1J_1}/D_1,\cdots,v_{I_lJ_l}/D_l]],\cdots, \\
 & & \quad B_n[v_{G_1H_1}/C_1[v_{I_1J_1}/D_1,\cdots,v_{I_lJ_l}/D_l],\cdots, \\
 & & \qquad v_{G_mH_m}/C_m[v_{I_1J_1}/D_1,\cdots,v_{I_lJ_l}/D_l]]) \\
 & = & [\![u\circ(v\circ w)]\!] \\
 & & \text{where} \\
 & \Xi = & (I_1\Rightarrow J_1,\cdots,I_l\Rightarrow J_l) \\
 & [\![w]\!] = & (D_1,\cdots,D_l)
\end{array}
$$

$$
\begin{array}{lll}
(ECM4) & [\![w]\!] = & (\ ) = [\![!_\Gamma]\!] \\
(ECM5) & [\![\mathbf{d}_{K\Rightarrow L}\circ\prec u, A\succ]\!] & \\
 & = & (v_{K_1L_1}[v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n,v_{KL}/[\![A]\!]],\cdots, \\
 & & \quad v_{K_nL_n}[v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n,v_{KL}/[\![A]\!]]) \\
 & = & (B_1,\cdots,B_n) = [\![u]\!] \\
(ECM6) & [\![\prec\mathbf{d}_{K\Rightarrow L}\circ w,\chi_{K\Rightarrow L}[w]\succ]\!] & \\
 & = & (v_{K_1L_1}[v_{K_1L_1}/D_1,\cdots,v_{K_nL_n}/D_n,v_{KL}/D_{n+1}],\cdots, \\
 & & \quad v_{K_nL_n}[v_{K_1L_1}/D_1,\cdots,v_{K_nL_n}/D_n,v_{KL}/D_{n+1}], \\
 & & \quad v_{KL}[v_{K_1L_1}/D_1,\cdots,v_{K_nL_n}/D_n,v_{KL}/D_{n+1}]) \\
 & = & (D_1,\cdots,D_{n+1}) \\
 & & \text{where} \\
 & [\![w]\!] = & (D_1,\cdots,D_{n+1})
\end{array}
$$

$$
\begin{array}{lll}
(ETY1) & [\![K[\mathbf{id}_\Gamma]]\!] = & [\![K]\!][v_{K_1L_1}/v_{K_1L_1},\cdots,v_{K_nL_n}/v_{K_nL_n}] \\
 & = & [\![K]\!] \\
(ETY2) & [\![K[u][v]]\!] = & [\![K]\!][v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n][v_{G_1H_1}/C_1,\cdots,v_{G_mH_m}/C_m] \\
 & = & [\![K]\!][v_{K_1L_1}/B_1[v_{G_1H_1}/C_1,\cdots,v_{G_mH_m}/C_m],\cdots, \\
 & & \quad v_{K_nL_n}/B_1[v_{G_1H_1}/C_1,\cdots,v_{G_mH_m}/C_m]] \\
 & = & [\![K[u\circ v]]\!]
\end{array}
$$

$$
\begin{aligned}
\text{(ETY3)}\quad & [\![\Pi_{K\Rightarrow L}.H[u]]\!] \\
&= (\Pi z\colon ([\![K]\!] \supset [\![L]\!]).[\![H]\!][v_{KL}/z])[v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n] \\
&= \Pi z'\colon [\![K]\!][v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n]\supset[\![L]\!][v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n]. \\
&\qquad [\![H]\!][v_{KL}/z][v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n][z/z'] \\
&= \Pi z'\colon [\![K[u]]\!]\supset[\![L[u]]\!]. \\
&\qquad [\![H[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]},\chi_{K[u]\Rightarrow L[u]}\succ]]\!][v_{KL}/z'] \\
&= [\![\Pi_{K[u]\Rightarrow L[u]}.H[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]},\chi_{K[u]\Rightarrow L[u]}\succ]]\!]
\end{aligned}
$$

$$
\begin{aligned}
(ETM1)\quad [\![\mathbf{id}_L\circ A]\!] &= \lambda x\colon [\![K]\!].((\lambda z\colon [\![L]\!].(z))([\![A]\!]x)) \\
&= \lambda x\colon [\![K]\!].([\![A]\!]x) \\
&= [\![A]\!]
\end{aligned}
$$

$(ETM2)$ (similar to $(ETM1)$)

$$
\begin{aligned}
(ETM3)\quad [\![(C\circ B)\circ A]\!] &= \lambda x\colon [\![K]\!].((\lambda x'\colon [\![L]\!].([\![C]\!]([\![B]\!]x')))([\![A]\!]x)) \\
&= \lambda x\colon [\![K]\!].([\![C]\!]([\![B]\!]([\![A]\!]x))) \\
&= \lambda x\colon [\![K]\!].([\![C]\!]((\lambda x'\colon [\![G]\!].([\![B]\!]([\![A]\!]x')))x)) \\
&= [\![C\circ(B\circ A)]\!]
\end{aligned}
$$

$$
\begin{aligned}
(ETM4)\quad [\![C]\!] &= \lambda z\colon \mathbf{1}.([\![C]\!]z) \\
&= \lambda z\colon \mathbf{1}.(*) \\
&= [\![!_K]\!]
\end{aligned}
$$

$$
\begin{aligned}
(ETM5)\quad [\![A[\mathbf{id}_\Gamma]]\!] &= [\![A]\!][v_{K_1L_1}/v_{K_1L_1},\cdots,v_{K_nL_n}/v_{K_nL_n}] \\
&= [\![A]\!]
\end{aligned}
$$

$(ETM6)$ (similar to $(ETY2)$)

$$
\begin{aligned}
(ETM7)\quad [\![\mathbf{id}_K[u]]\!] &= \lambda x\colon [\![K]\!].(x)[v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n] \\
&= \lambda x\colon [\![K[u]]\!].(x) = [\![\mathbf{id}_{K[u]}]\!]
\end{aligned}
$$

$$
\begin{aligned}
(ETM8)\quad [\![B[u]\circ A[u]]\!] &= \lambda x\colon [\![K[u]]\!].([\![B]\!][v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n] \\
&\qquad ([\![A]\!][v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n]x)) \\
&= (\lambda x\colon [\![K]\!].([\![B]\!]([\![A]\!]x)))[v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n] \\
&= [\![(B\circ A)[u]]\!]
\end{aligned}
$$

$$
\begin{aligned}
\text{(ETM9)}\quad [\![\chi_{K\Rightarrow L}[\prec u,A\succ]]\!] &= v_{KL}[v_{K_1L_1}/B_1,\cdots,v_{K_nL_n}/B_n,v_{KL}/[\![A]\!]] \\
&= [\![A]\!]
\end{aligned}
$$

$$
\begin{aligned}
(ETM10)\quad & [\![\varphi^{H}_{K\Rightarrow L}\circ(\lambda_{K\Rightarrow L}.(A)[\mathbf{d}_{K\Rightarrow L}])]\!] \\
&= \lambda x{:}\,[\![G]\!].((\lambda x'{:}\,[\![\Pi_{K\Rightarrow L}.H]\!].(x' v_{KL})) \\
&\qquad ((\lambda z{:}\,[\![G]\!].\lambda z'{:}\,([\![K]\!] \supset [\![L]\!]).[\![A]\!][v_{KL}/z']z)x)) \\
&= \lambda x{:}\,[\![G]\!].([\![A]\!]x) \\
&= [\![A]\!]
\end{aligned}
$$

$$
\begin{aligned}
(ETM11)\quad & [\![\lambda_{K\Rightarrow L}.(\varphi^{H}_{K\Rightarrow L}\circ B[\mathbf{d}_{K\Rightarrow L}])]\!] \\
&= \lambda x{:}\,[\![G]\!].\lambda z{:}\,([\![K]\!] \supset [\![L]\!]).((\lambda x'{:}\,[\![G]\!]. \\
&\qquad ((\lambda z'{:}\,[\![\Pi_{K\Rightarrow L}.H]\!].(z' v_{KL}))([\![B]\!]x')))[v_{KL}/z])x \\
&= \lambda x{:}\,[\![G]\!].\lambda z{:}\,([\![K]\!] \supset [\![L]\!]).\lambda x'{:}\,[\![G]\!].([\![B]\!]x'zx) \\
&= [\![B]\!]
\end{aligned}
$$

$$
\begin{aligned}
(ETM12)\quad & [\![\varphi^{H}_{K\Rightarrow L}[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]}, \chi_{K[u]\Rightarrow L[u]}\succ]]\!] \\
&= (\lambda x{:}\,[\![\Pi_{K\Rightarrow L}.H]\!].(x\ v_{KL}))[v_{K_1L_1}/B_1, \cdots, v_{K_nL_n}/B_n, v_{KL}/v_{K[u]L[u]}] \\
&= \lambda x{:}\,[\![\Pi_{K\Rightarrow L}.H[u]]\!].(x\ v_{K[u]L[u]}) \\
&= [\![\varphi^{H[\prec u\circ\mathbf{d}_{K[u]\Rightarrow L[u]}, \chi_{K[u]\Rightarrow L[u]}\succ]}_{K[u]\Rightarrow L[u]}]\!]
\end{aligned}
$$

▮

A.1.4 Proof of Theorem 6.5.4

First, observe:

Lemma A.1.11. In any $\mathcal{TF}\iota$ theory T_F, a well-formed context $(\Gamma, K \Rightarrow L)$ is isomorphic to $(\Gamma, \mathbf{1} \Rightarrow K{\supset}L)$.

Proof In T_F, the following two terms can be derived.

$$
\frac{(\Gamma, K \Rightarrow L)\vdash \chi_{K\Rightarrow L}{:}\, K[\mathbf{d}_{K\Rightarrow L}] \Rightarrow L[\mathbf{d}_{K\Rightarrow L}]}{(\Gamma, K \Rightarrow L)\vdash \mathrm{P}_{KL} \equiv \mathrm{code}_{K[\mathbf{d}_{K\Rightarrow L}]\Rightarrow L[\mathbf{d}_{K\Rightarrow L}]}(\chi_{K\Rightarrow L}){:}\, \mathbf{1} \Rightarrow K[\mathbf{d}_{K\Rightarrow L}]{\supset}L[\mathbf{d}_{K\Rightarrow L}]}
$$

$$
\frac{\Gamma, \mathbf{1} \Rightarrow K{\supset}L\vdash \chi_{\mathbf{1}\Rightarrow K\supset L}{:}\, \mathbf{1} \Rightarrow K{\supset}L}{\Gamma, \mathbf{1}\Rightarrow K{\supset}L\vdash \mathrm{Q}_{KL}\equiv\mathrm{decode}_{K[\mathbf{d}_{K\Rightarrow L}]\Rightarrow L[\mathbf{d}_{K\Rightarrow L}]}(\chi_{\mathbf{1}\Rightarrow K\supset L}){:}\, K[\mathbf{d}_{\mathbf{1}\Rightarrow K\supset L}]\Rightarrow L[\mathbf{d}_{\mathbf{1}\Rightarrow K\supset L}]}
$$

Then, context mappings $\prec\mathbf{d}_{K\Rightarrow L}, \mathrm{P}_{KL}\succ$ and $\prec\mathbf{d}_{\mathbf{1}\Rightarrow K\supset L}, \mathrm{Q}_{KL}\succ$ give a desired isomorphism. ▮

The full version of Theorem 6.5.4 is as follows.

Theorem A.1.12. (Definition-Theorem) Given a $\mathcal{TF}\iota$ theory T_F, the following are defined and satisfy the conditions below.

- For each well-formed expression E of T_F, its *h.e. form* E^+ is defined and satisfies a well-formedness condition.
  - For Γ **Context**, Γ^+ **Context**.
  - For $u\colon \Delta \to \Gamma$, $u^+\colon \Delta^+ \to \Gamma^+$.
  - For $\Gamma \vdash H\colon \mathbf{Type}$, $\Gamma^+ \vdash H^+\colon \mathbf{Type}$.
  - For $\Gamma \vdash A\colon K \Rightarrow L$, $\Gamma^+ \vdash A^+\colon \mathbf{1} \Rightarrow K^+ \supset L^+$.
- For each well-formed context Γ, there are two context mappings $\alpha_\Gamma\colon \Gamma \to \Gamma^+$ and $\beta_\Gamma\colon \Gamma^+ \to \Gamma$ satisfying $\alpha_\Gamma \circ \beta_\Gamma = \mathbf{id}_{\Gamma^+}\colon \Gamma^+ \to \Gamma^+$ and $\beta_\Gamma \circ \alpha_\Gamma = \mathbf{id}_\Gamma\colon \Gamma \to \Gamma$.
- For each context mapping $u\colon \Delta \to \Gamma$, $u^+ \circ \alpha_\Delta = \alpha_\Gamma \circ u\colon \Delta \to \Gamma^+$ holds.
- For each type $\Gamma \vdash K\colon \mathbf{Type}$, there are terms $\Gamma \vdash \gamma_K\colon K \Rightarrow K^+[\alpha_\Gamma]$ and $\Gamma \vdash \delta_K\colon K^+[\alpha_\Gamma] \Rightarrow K$ satisfying $\Gamma \vdash \gamma_K \circ \delta_K = \mathbf{id}_{K^+[\alpha_\Gamma]}\colon K^+[\alpha_\Gamma] \Rightarrow K^+[\alpha_\Gamma]$ $\Gamma \vdash \delta_K \circ \gamma_K = \mathbf{id}_K\colon K \Rightarrow K$.
- The map $A \mapsto A^+$ is bijective and the inverse is

$$B \mapsto B^- \equiv \delta_L \circ \mathrm{decode}_{K^+ \Rightarrow L^+}(B)[\alpha_\Gamma] \circ \gamma_K.$$

These are all mutually defined by induction on derivations in T_F.

(C1) $()^+ = ()$

$$\alpha_{()} = \beta_{()} = \mathbf{id}_{()}$$

(C2) $(\Gamma, K \Rightarrow L)^+ = (\Gamma^+, \mathbf{1} \Rightarrow K^+ \supset L^+)$

$$\begin{aligned}\alpha_{(\Gamma, K \Rightarrow L)} &= \prec \mathbf{d}_{K^+ \Rightarrow L^+}, \mathrm{P}_{K^+ L^+} \succ \\ &\quad \circ \prec \alpha_\Gamma \circ \mathbf{d}_{K \Rightarrow L}, \gamma_L[\mathbf{d}_{K \Rightarrow L}] \circ \chi_{K \Rightarrow L} \circ \delta_K[\mathbf{d}_{K \Rightarrow L}] \succ\end{aligned}$$

$$\begin{aligned}&\beta_{(\Gamma, K \Rightarrow L)} \\ &\quad = \prec \beta_\Gamma \circ \mathrm{Q}_{K^+ L^+}, \delta_L[\beta_\Gamma \circ \mathbf{d}_{K^+ \Rightarrow L^+}] \circ \mathrm{Q}_{K^+ L^+} \\ &\qquad \circ \gamma_L[\beta_\Gamma \circ \mathbf{d}_{K^+ \Rightarrow L^+}] \succ \circ \\ &\qquad \prec \mathbf{d}_{K^+ \Rightarrow L^+}, \mathrm{Q}_{K^+ L^+} \succ\end{aligned}$$

(CM1) $(\mathbf{id}_\Gamma)^+ = \mathbf{id}_{\Gamma^+}$

(CM2) $(u\circ v)^+ = u^+\circ v^+$

(CM3) $(!_\Gamma)^+ = !_{\Gamma^+}$

(CM4) $(\mathbf{d}_{K\Rightarrow L})^+ = \mathbf{d}_{\mathbf{1}\Rightarrow K^+\supset L^+}$

(CM5) $(\prec u, A\succ)^+ = \prec u^+, A^+\succ$

(TY1) $\mathbf{1}^+ = \mathbf{1}$

(TY2) $\mathrm{F}^+ = \mathrm{F}[\beta_\Gamma]$

$\gamma_\mathrm{F} = \delta_\mathrm{F} = \mathbf{id}_\mathrm{F}$

(TY3) $(H[u])^+ = H^+[u^+] \qquad \gamma_{H[u]} = \gamma_H[u] \qquad \delta_{H[u]} = \delta_H[u]$

(TY4) $(\Pi_{K\Rightarrow L}.H)^+ = \Pi_{\mathbf{1}\Rightarrow K^+\supset L^+}.H^+$

$\gamma_{\Pi_{K\Rightarrow L}.H} = \lambda_{K\Rightarrow L}.((\delta_H\circ\varphi^{H^+}_{\mathbf{1}\Rightarrow K^+\supset L^+})[\alpha_{(\Gamma,K\Rightarrow L)}])$

$\delta_{\Pi_{K\Rightarrow L}.H} = \lambda_{\mathbf{1}\Rightarrow K^+\supset L^+}.((\gamma_H\circ\varphi^H_{K\Rightarrow L})[\beta_{(\Gamma,K\Rightarrow L)}])[\alpha_\Gamma]$

(TY5) $(\Sigma_{K\Rightarrow L}.G)^+ = \Sigma_{\mathbf{1}\Rightarrow K^+\supset L^+}.G^+$

$\gamma_{\Sigma_{K\Rightarrow L}.H} = \nu_{\mathbf{1}\Rightarrow K^+\supset L^+}.((\psi^G_{K\Rightarrow L}\circ\delta_G)[\beta_{(\Gamma,K\Rightarrow L)}])[\alpha_\Gamma]$

$\delta_{\Sigma_{K\Rightarrow L}.H} = \nu_{K\Rightarrow L}.((\psi^{G^+}_{\mathbf{1}\Rightarrow K^+\supset L^+}\circ\gamma_G)[\alpha_{(\Gamma,K\Rightarrow L)}])$

(TM1) $\mathrm{C}^+ = \mathrm{C}[\beta_\Gamma]$

(TM2) ${\mathbf{id}_K}^+ = \mathbf{id}_{K^+}$

(TM3) $(B\circ A)^+ = \mathrm{code}_{K^+\Rightarrow G^+}(\mathrm{decode}_{L^+\Rightarrow G^+}(B^+)\circ\mathrm{decode}_{K^+\Rightarrow L^+}(A^+))$

(TM4) ${!_K}^+ = !_{K^+}$

(TM5) ${\chi_{K\Rightarrow L}}^+ = \chi_{\mathbf{1}\Rightarrow K^+\supset L^+}$

(TM6) $(A[u])^+ = A^+[u^+]$

(TM7) $(\lambda_{K\Rightarrow L}.(A))^+$

$= \mathrm{code}_{G^+\Rightarrow(\Pi_{K\Rightarrow L}.H)^+}(\lambda_{\mathbf{1}\Rightarrow K^+\supset L^+}.(\mathrm{decode}_{G^+[\mathbf{d}]\Rightarrow H^+}(A^+)))$

(TM8) $(\varphi^H_{K\Rightarrow L})^+ = \varphi^{H^+}_{\mathbf{1}\Rightarrow K^+\supset L^+}$

(TM9) $(\nu_{K\Rightarrow L}.(A))^+$

$= \mathrm{code}_{(\Sigma_{K\Rightarrow L}.G)^+\Rightarrow H^+}(\nu_{\mathbf{1}\Rightarrow K^+\supset L^+}.(\mathrm{decode}_{G^+\Rightarrow H^+[\mathbf{d}]}(A^+)))$

(TM10) $(\psi^G_{K\Rightarrow L})^+ = \psi^{G^+}_{\mathbf{1}\Rightarrow K^+\supset L^+}$

(TM11) ${\iota_K}^+ = \mathrm{code}_{K^+\Rightarrow(\Sigma_{\mathbf{1}\Rightarrow K}.\mathbf{1})^+}(\iota_{K^+})$

Proof Required conditions are checked by straightforward calculation. ∎

Note that A^+ is equal to $\mathrm{code}_{K^+\Rightarrow L^+}((\gamma_L \circ A \circ \delta_K)[\beta_\Gamma])$, but this *expression* is not h.e..

A.1.5 Proof of Proposition 6.5.6

Proof In the forward direction, one can show that two maps $u \mapsto \mathrm{curr}_K(\chi_{\mathbf{1}\Rightarrow L}[u])$ and $A \mapsto \prec \mathbf{d}_{\mathbf{1}\Rightarrow K}, \mathrm{uncurr}_K(A) \succ$ are reciprocal to each other. In the reverse direction, if the bijection $A \leftrightarrow u$ is given, the inverse of $\nu_{\mathbf{1}\Rightarrow K}.(\chi_{\mathbf{1}\Rightarrow K})$ is obtained as the term $\Gamma \vdash A \colon K \Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.\mathbf{1}$ that corresponds to

$$\prec \mathbf{d}_{\mathbf{1}\Rightarrow K}, \psi^{\mathbf{1}}_{\mathbf{1}\Rightarrow K} \succ \colon (\Gamma, \mathbf{1} \Rightarrow K) \to (\Gamma, \mathbf{1} \Rightarrow \Sigma_{\mathbf{1}\Rightarrow K}.\mathbf{1}).$$

∎

A.1.6 Proof of Proposition 6.6.2

Lemma A.1.13. The substitution lemma A.1.7 for $[\![-]\!]_{\mathcal{TF}\lambda}$ also holds for any translation $[\![-]\!]_{\mathcal{TF}\iota}$ from a $w\mathcal{ML}$ theory T_w to a $\mathcal{TF}\iota$ theory T_F.

Proof We only give the proof for the case of $\mathbf{let}_{\langle x_{n+1}, w\rangle = B}$ **in** C, i.e., if

$$\vec{x} \colon \vec{K} \vdash H \colon \mathbf{Type},$$

$$\vec{x} \colon \vec{K}, x_{n+1} \colon K_{n+1}, w \colon G \vdash C \colon H,$$

$$\vec{x} \colon \vec{K} \vdash B \colon \Sigma x_{n+1} \colon K_{n+1}.G, \text{ and}$$

$$\vec{y} \colon \vec{G} \vdash A_i \colon K_i[x_1, \cdots x_{i-1}/A_1, \cdots A_{i-1}] (1 \le i \le n)$$

are derivable in T_w, one has the following equality judgement in T_F.

$$[\![\vec{y} \colon \vec{G}]\!] \vdash [\![(\mathbf{let}_{\langle x_{n+1}, w\rangle = B} \ \mathbf{in} \ C)[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}} = [\![\mathbf{let}_{\langle x_{n+1}, w\rangle = B} \ \mathbf{in} \ C]\!]^{\vec{x}:\vec{K}}[u_n]$$
$$: \mathbf{1} \Rightarrow [\![H]\!]^{\vec{x}:\vec{K}}[u_n]$$

where u_n is the context mapping defined in the clause for (subst1) of Definition 6.4.1.

$$[\![(\mathbf{let}_{\langle x_{n+1}, w\rangle = B} \ \mathbf{in} \ C)[\vec{x}/\vec{A}]]\!]_{\mathcal{TF}\iota}^{\vec{x}:\vec{K}} = [\![\mathbf{let}_{\langle y_{n+1}, w'\rangle = B[\vec{x}/\vec{A}]} \ \mathbf{in} \ C']\!]^{\vec{x}:\vec{K}}$$

$$
\begin{aligned}
&= \nu_{\mathbf{1}\Rightarrow[\![K_{n+1}{}']\!]^{\vec{y}:\vec{G}}}.(\nu_{\mathbf{1}\Rightarrow G'}.(C')\circ\iota_{G'})\circ[\![B[\vec{x}/\vec{A}]]\!]^{\vec{y}:\vec{G}} \\
&= \nu_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[u_n]}.(\nu_{\mathbf{1}\Rightarrow G_1[\prec u_n\circ d_1,\chi_1\succ]}.(C_1[\prec\prec u_n\circ d_1\circ d_2,\chi_1[d_2]\succ,\chi_2\succ])\circ \\
&\quad \iota_{G_1[\prec\prec u_n\circ d_1\circ d_2,\chi_1[d_2]\succ,\chi_2\succ]})\circ[\![B]\!]^{\vec{x}:\vec{K}}[u_n] \\
&= \nu_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[u_n]}.(\nu_{\mathbf{1}\Rightarrow G_1[\prec u_n\circ d_1,\chi_1\succ]}.(\\
&\quad (C_1\circ\iota_{G_1})[\prec\prec u_n\circ d_1\circ d_2,\chi_1[d_2]\succ,\chi_2\succ]))\circ[\![B]\!]^{\vec{x}:\vec{K}}[u_n] \\
&= (\nu_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}}.(\nu_{\mathbf{1}\Rightarrow G_1}.(C_1)\circ\iota_{G_1})\circ[\![B]\!]^{\vec{x}:\vec{K}}[u_n])[u_n] \\
&= [\![\mathbf{let}_{\langle x_{n+1},w\rangle=B}\ \mathbf{in}\ C]\!]^{\vec{x}:\vec{K}}
\end{aligned}
$$

where

$$
\begin{aligned}
&K_{n+1}{}' = K_{n+1}[\vec{x}/\vec{A}], \\
&G_1 = [\![G]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}, \\
&C_1 = [\![C]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1},w:G}, \\
&G' = [\![G[\vec{x}/\vec{A},x_{n+1}/y_{n+1}]]\!]^{\vec{y}:\vec{G},y_{n+1}:K_{n+1}{}'}, \\
&C' = [\![C[\vec{x}/\vec{A},x_{n+1}/y_{n+1},w/w']]\!]^{\vec{y}:\vec{G},y_{n+1}:K_{n+1}{}',w':G'}, \\
&d_1 = \mathbf{d}_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[u_n]}, \\
&\chi_1 = \chi_{\mathbf{1}\Rightarrow[\![K_{n+1}]\!]^{\vec{x}:\vec{K}}[u_n]}, \\
&d_2 = \mathbf{d}_{\mathbf{1}\Rightarrow G_1[\prec u_n\circ d_1,\chi_1\succ]},\ \text{and} \\
&\chi_1 = \chi_{\mathbf{1}\Rightarrow G_1[\prec u_n\circ d_1,\chi_1\succ]}.
\end{aligned}
$$

▮

Proof (of Proposition 6.6.2)

With the above lemma, it is sufficient to check (β') and (η') rules in Section 5.2.3 are soundly translated. The notation is the same as the one used in Section 5.2.3. With abbreviation

$$
\begin{aligned}
&K_1 = [\![K_{n+1}]\!]^{\vec{x}:\vec{K}}, \\
&G_1 = [\![G]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1}}, \\
&B_1 = [\![B]\!]^{\vec{x}:\vec{K},x_{n+1}:K_{n+1},y:G}, \\
&C_1 = [\![C]\!]^{\vec{x}:\vec{K},z:\Sigma x_{n+1}:K_{n+1}.G},\ \text{and}
\end{aligned}
$$

$$S_1 = [\![\Sigma x_{n+1}\colon K_{n+1}.G]\!]^{\vec{x}:\vec{K}},$$

we have the following equations.

$$
\begin{aligned}
&[\![\mathbf{let}_{\langle x_{n+1},y\rangle=\langle A_1,A_2\rangle}\ \mathbf{in}\ B]\!]^{\vec{x}:\vec{K}}\\
&\quad= \nu_{\mathbf{1}\Rightarrow K_1}.\mathrm{curr}_{G_1}(B_1)\circ\psi^{G_1}_{\mathbf{1}\Rightarrow K_1}[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A_1]\!]^{\vec{x}:\vec{K}}\succ]\circ[\![A_2]\!]^{\vec{x}:\vec{K}}\\
&\quad= (\nu_{\mathbf{1}\Rightarrow K_1}.\mathrm{curr}_{G_1}(B_1)[\mathbf{d}_{\mathbf{1}\Rightarrow K_1}]\circ\psi^{G_1}_{\mathbf{1}\Rightarrow K_1})[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A_1]\!]^{\vec{x}:\vec{K}}\succ]\circ[\![A_2]\!]^{\vec{x}:\vec{K}}\\
&\quad= (\nu_{\mathbf{1}\Rightarrow G_1}.(B_1)\circ\iota_{G_1})[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A_1]\!]^{\vec{x}:\vec{K}}\succ]\circ[\![A_2]\!]^{\vec{x}:\vec{K}}\\
&\quad= ((\nu_{\mathbf{1}\Rightarrow G_1}.(B_1)\circ\iota_{G_1})[\mathbf{d}_{\mathbf{1}\Rightarrow S_1}]\circ\chi_{\mathbf{1}\Rightarrow S_1})[\prec\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A_1]\!]^{\vec{x}:\vec{K}}\succ,[\![A_2]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad= \mathrm{uncurr}_{S_1}(\mathrm{curr}_{G_1}(B_1))[\prec\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A_1]\!]^{\vec{x}:\vec{K}}\succ,[\![A_2]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad= B_1[\prec\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A_1]\!]^{\vec{x}:\vec{K}}\succ,[\![A_2]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad= [\![B[x_{n+1}/A_1,y/A_2]]\!]^{\vec{x}:\vec{K}}
\end{aligned}
$$

$$
\begin{aligned}
&[\![\mathbf{let}_{\langle x,y\rangle=A}\ \mathbf{in}\ C[z/\langle x,y\rangle]]\!]^{\vec{x}:\vec{K}}\\
&\quad= \nu_{\mathbf{1}\Rightarrow K_1}.(\mathrm{curr}_{G_1}(C_1[\prec\mathbf{d}_{\mathbf{1}\Rightarrow K_1}\circ\mathbf{d}_{\mathbf{1}\Rightarrow G_1},\mathrm{pair}_{K_1G_1}\succ]))\circ[\![A]\!]^{\vec{x}:\vec{K}}\\
&\quad= \nu_{\mathbf{1}\Rightarrow K_1}.(\mathrm{curr}_{G_1}(\mathrm{uncurr}_{S_1}(\mathrm{curr}_{S_1}(C_1))[\prec\mathbf{d}_{\mathbf{1}\Rightarrow K_1}\circ\mathbf{d}_{\mathbf{1}\Rightarrow G_1},\mathrm{pair}_{K_1G_1}\succ]))\circ[\![A]\!]^{\vec{x}:\vec{K}}\\
&\quad= \nu_{\mathbf{1}\Rightarrow K_1}.(\mathrm{curr}_{G_1}(\mathrm{curr}_{S_1}(C_1)[\mathbf{d}_{\mathbf{1}\Rightarrow K_1}\circ\mathbf{d}_{\mathbf{1}\Rightarrow G_1}]\circ\psi^{G_1}_{\mathbf{1}\Rightarrow K_1}[\mathbf{d}_{\mathbf{1}\Rightarrow G_1}]\circ\chi_{\mathbf{1}\Rightarrow G_1}))\circ[\![A]\!]^{\vec{x}:\vec{K}}\\
&\quad= \nu_{\mathbf{1}\Rightarrow K_1}.(\mathrm{curr}_{G_1}(\mathrm{uncurr}_{G_1}((\mathrm{curr}_{S_1}(C_1))[\mathbf{d}_{\mathbf{1}\Rightarrow K_1}]\circ\psi^{G_1}_{\mathbf{1}\Rightarrow K_1})))\circ[\![A]\!]^{\vec{x}:\vec{K}}\\
&\quad= \nu_{\mathbf{1}\Rightarrow K_1}.((\mathrm{curr}_{S_1}(C_1))[\mathbf{d}_{\mathbf{1}\Rightarrow K_1}]\circ\psi^{G_1}_{\mathbf{1}\Rightarrow K_1})\circ[\![A]\!]^{\vec{x}:\vec{K}}\\
&\quad= \mathrm{curr}_{S_1}(C_1)\circ[\![A]\!]^{\vec{x}:\vec{K}}\\
&\quad= (\mathrm{curr}_{S_1}(C_1)[\mathbf{d}_{\mathbf{1}\Rightarrow S_1}]\circ\chi_{\mathbf{1}\Rightarrow S_1})[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad= \mathrm{uncurr}_{S_1}(\mathrm{curr}_{S_1}(C_1))[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad= C_1[\prec\mathbf{id}_{[\![\vec{x}:\vec{K}]\!]},[\![A]\!]^{\vec{x}:\vec{K}}\succ]\\
&\quad= [\![C[z/A]]\!]^{\vec{x}:\vec{K}}
\end{aligned}
$$

▮

A.1.7 Proof of Proposition 6.6.5

The proposition is proved similarly to Proposition 6.4.4. We only give extensions of the maps $\check{()}$ and $\hat{()}$ for dependent sum types.

- $(\Sigma\mathrm{F})$
  - For $\vec{x}\colon \vec{K} \vdash \Sigma x_{n+1}\colon K_{n+1}.G\colon \mathbf{Type}$,

    $$\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash(\Sigma x_{n+1}\colon K_{n+1}.)\hat{} \equiv \Sigma z_{n+1}\colon \mathbf{1}\supset(K_{n+1})\hat{}.\hat{G}[\vec{z}, z_{n+1}]\colon \mathbf{Type}.$$

  - For $\vec{x}\colon \vec{K}\vdash A\colon \Sigma x_{n+1}\colon K_{n+1}.G$,

    $$\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash \hat{A}[\vec{z}] \equiv \lambda v\colon \mathbf{1}.(\mathbf{let}_{\langle u,w\rangle=A}\ \mathbf{in}\ \langle \hat{u}, \hat{w}(*)\rangle)\colon \mathbf{1}\supset(\Sigma x_{n+1}\colon K_{n+1}.G)\hat{}.$$

  - For $\vec{z}\colon \mathbf{1}\supset(\vec{K})\hat{}\vdash B\colon \mathbf{1}\supset(\Sigma x_{n+1}\colon K_{n+1}.G)\hat{}$,

    $$\vec{x}\colon \vec{K}\vdash \check{B}[\vec{x}] \equiv \mathbf{let}_{\langle v,w\rangle=B(*)}\ \mathbf{in}\ \langle \check{v}, (\lambda v'\colon \mathbf{1}.(w))\check{}\rangle\colon \Sigma x_{n+1}\colon K_{n+1}.G.$$

∎

A.1.8 Proof of Theorem 6.6.7

With evident induction, one can prove that all introduction rules and axioms are well-formed and that $[\![-]\!]_{\mathcal{G}(T_F)}$ defines a translation.

Lemma A.1.14. The canonical translation $[\![-]\!]_{\mathcal{G}(T_F)}$ sends every well-formed type and term expression of $\mathcal{G}(T_F)$ to an h.e. type and an h.e. term of T_F, respectively.
∎

Lemma A.1.15. In $\mathcal{G}(T_F)$, every well-formed type or term is equal to a constant type or a constant term.

Proof Suppose $\vec{x}\colon \vec{K}\vdash H\colon \mathbf{Type}$ in $\mathcal{G}(T_F)$. By the above lemma, $\mathrm{c}_{[\![H]\!]_{\mathcal{G}(T_F)}}$ is a constant symbol of $\mathcal{G}(T_F)$. The judgement

$$\vec{x}\colon \vec{K}\vdash \mathrm{c}_{[\![H]\!]_{\mathcal{G}(T_F)}} = H\colon \mathbf{Type}$$

is an axiom of $\mathcal{G}(T_F)$ since $[\![-]\!]_{\mathcal{G}(T_F)}$ sends this to

$$[\![\vec{x}\colon \vec{K}]\!]_{\mathcal{G}(T_F)}\vdash [\![H]\!]_{\mathcal{G}(T_F)} = [\![H]\!]_{\mathcal{G}(T_F)}\colon \mathbf{Type},$$

which is derivable in the theory T_F by definition. The same argument serves for term expressions. ∎

Lemma A.1.16. For every well-formed type H of T_F, $(I(H)_{\mathcal{T}F_\iota})^+ = [\![\mathrm{c}_{H^+}]\!]_{\mathcal{T}F_\iota}$ holds.

Proof By induction on the derivation of $\Gamma\vdash H\colon\mathbf{Type}$ and $\Gamma\vdash A\colon K \Rightarrow L$ in T_F. For constant types, the lemma is obvious. As for $\Pi_{K\Rightarrow L}.H$, for example, one has

$$\begin{aligned}(I(\Pi_{K\Rightarrow L}.H)_{\mathcal{T}F_\iota})^+ &= (\Pi_{I(K)_{\mathcal{T}F_\iota}\Rightarrow I(L)_{\mathcal{T}F_\iota}}.I(H)_{\mathcal{T}F_\iota})^+\\ &= \Pi_{\mathbf{1}\Rightarrow I(K)_{\mathcal{T}F_\iota}{}^+\supset I(L)_{\mathcal{T}F_\iota}{}^+}.I(H)_{\mathcal{T}F_\iota}{}^+\\ &= \Pi_{\mathbf{1}\Rightarrow [\![\mathrm{c}_{K^+}]\!]_{\mathcal{T}F_\iota}\supset [\![\mathrm{c}_{L^+}]\!]_{\mathcal{T}F_\iota}}.[\![\mathrm{c}_{H^+}]\!]_{\mathcal{T}F_\iota}\\ &= [\![\Pi x\colon(\Pi y\colon \mathrm{c}_{K^+}.\mathrm{c}_{L^+}).\mathrm{c}_{H^+}]\!]_{\mathcal{T}F_\iota}\\ &= [\![\mathrm{c}_{\Pi_{\mathbf{1}\Rightarrow K^+\supset L^+}.H^+}]\!]_{\mathcal{T}F_\iota}\\ &= [\![\mathrm{c}_{(\Pi_{K\Rightarrow L}.H)^+}]\!]_{\mathcal{T}F_\iota}.\end{aligned}$$

∎

Proposition A.1.17. $I(-)_{\mathcal{T}F_\iota}$ is an interpretation.

Proof Consider an introduction rule $\Gamma\vdash\mathrm{F}\colon\mathbf{Type}$ of T_F. Let Γ be the context $(K_1 \Rightarrow L_1, \ldots, K_n \Rightarrow L_n)$. By Theorem A.1.12, $\Gamma^+\vdash\mathrm{F}[\beta_\Gamma]\colon\mathbf{Type}$ is derivable in T_F. Since $\Gamma^+ = (\ldots, \mathbf{1} \Rightarrow K^+\supset L^+, \ldots)$, $(\ldots, x_i\colon \mathrm{c}_{K^+\supset L^+}, \ldots)\vdash\mathrm{c}_{\mathrm{F}[\beta_\Gamma]}\colon\mathbf{Type}$ is an introduction rule of $\mathcal{G}(T_F)$ and

$$(\ldots, \mathbf{1} \Rightarrow [\![\mathrm{c}_{K^+\supset L^+}]\!]_{\mathcal{T}F_\iota}, \ldots)\vdash[\![\mathrm{c}_{\mathrm{F}[\beta_\Gamma]}]\!]_{\mathcal{T}F_\iota}\colon\mathbf{Type}$$

is derivable in T'_F. With the previous lemma, one has $I(K)_{\mathcal{T}F_\iota}{}^+\supset I(L)_{\mathcal{T}F_\iota}{}^+ = [\![\mathrm{c}_{K^+\supset L^+}]\!]_{\mathcal{T}F_\iota}$ and $(I(\Gamma)_{\mathcal{T}F_\iota})^+ = (\ldots, \mathbf{1} \Rightarrow [\![\mathrm{c}_{\Pi_{\mathbf{1}\Rightarrow K^+}.L^+[\mathbf{d}]}]\!]_{\mathcal{T}F_\iota}, \ldots)$. Therefore,

$$I(\Gamma)_{\mathcal{T}F_\iota}\vdash[\![\mathrm{c}_{\mathrm{F}[\beta_\Gamma]}]\!]_{\mathcal{T}F_\iota}[\alpha_{I(\Gamma)_{\mathcal{T}F_\iota}}]\colon\mathbf{Type}$$

is shown to be derivable in T'_F. The translations of introduction rules for terms and axioms of T_F are shown to be well-formed in a similar way. ∎

Lemma A.1.18. For an h.e. type H and an h.e. term A, $([\![\mathrm{c}_H]\!]_{\mathcal{T}F_\iota})^+ = [\![\mathrm{c}_{H^+}]\!]_{\mathcal{T}F_\iota}$ and $([\![\mathrm{c}_A]\!]_{\mathcal{T}F_\iota})^+ = [\![\mathrm{c}_{A^+}]\!]_{\mathcal{T}F_\iota}$. ∎

Proof (of Theorem 6.6.7)

It is sufficient to note that, on constant types and terms of T_w, the values of the iterated translation $I([\![-]\!]_{\mathcal{G}(T_F)})_{\mathcal{T}F_\iota}$ and $[\![-]\!]_{\mathcal{T}F_\iota}$ are equal by the previous lemma. ∎

A.1.9 Proof for Theorem 6.7.7

We only give the necessarily modification for Theorem A.1.12. The h.e. form of σ_{KL} is given by

(TM12)

$$\sigma^+_{KL} = \mathrm{code}_{\mathbf{1}\Rightarrow(L[\prec\mathbf{d},\mathrm{fst}_{KL}\succ])^+}(\sigma_{\mathbf{1}\supset K^+L^+}[\prec\mathbf{d}_{\mathbf{1}\Rightarrow\mathbf{1}\supset(\Sigma_{\mathbf{1}\Rightarrow K}.L)^+}, \mathrm{Q}_{\mathbf{1}(\Sigma_{\mathbf{1}\Rightarrow K}.L)^+}\succ]).$$

Bibliography

[1] T. Altenkirch. *Constructions, Inductive Types and Strong Normalization*. PhD thesis, University of Edinburgh, 1993.

[2] J. L. Bell. *Toposes and Local Set Theories: An Introduction*, volume 14 of *Oxford Logic Guides*. Oxford Science Publications, 1988.

[3] J. Bénabou. Structures algébraiques dans les catégories. *Cahiers de Top. et Géom. Diff.*, 10:1–126, 1968.

[4] J. Bénabou. 2-dimensional limits and colimits of distributors. *Mathematisches Forschungsinstitut Oberwolfach*, Tagungsbericht 30:6–7, 1972.

[5] J. Bénabou. Les distributeurs. *Seminaires de Math Pure*, Rapport No 33, 1973.

[6] J. Bénabou. Fibrations petites et localement petites. *Comptes Rendus Hebdomadaires de Séances de l'Académie de Sciences (Paris), Séries A et B*, 281:A897–A900, 1975.

[7] J. Bénabou. Théories relative à un coupus. *Comptes Rendus Hebdomadaires de Séances de l'Académie de Sciences (Paris), Séries A et B*, 281:A831–A834, 1975.

[8] J. Bénabou. Fibered categories and the foundations of naive category theory. *The Journal of Symbolic Logic*, 50:10–37, 1985.

[9] R. Betti, A. Carboni, R. Street, and R. Walters. Variation through enrichment. *Journal of Pure and Applied Algebra*, 29:109–127, 1983.

[10] Richard Buckland and Michael Johnson. Modelling algebraic structures with prolog, 1995. Extended abstract of full paper for CANT'95.

[11] R. Burstall and J. McKinna. Deliverables: a categorical approach to program development in type theory. Technical Report 92-242, LFCS, 1992.

[12] A. Carboni, S. Kasangian, and R. Walters. An axiomatics for bicategories of modules. *Journal of Pure and Applied Algebra*, 45:127–141, 1987.

[13] J. Cartmell. *Generalized Algebraic Theories and Contextual Categories*. PhD thesis, Oxford, 1978.

[14] J. Cartmell. Generalized algebraic theories and contextual categories. *Annals Pure Applied Logic*, 32:209–243, 1986.

[15] J. Celeyrette. Fibrations et extensions de Kan, 1975. Thèse de troisièm cycle, Université Paris-Nord.

[16] J. R. B. Cockett. Distributive logic. Technical Report CS-89-01, University of Tennessee, 1989.

[17] J. R. B. Cockett and D. Spencer. Strong categorical datatypes I. In R. A. G. Seely, editor, *Proc. International Meeting on Category Theory 1991, Canadian Mathmatical Society Proceedings*, pages 141–169, 1991.

[18] G. Cousineau, P-L. Curien, and M. Mauny. The categorical abstract machine. In *ACM Conf. on Functional Programming Languages and Computer Architecture, Nancy*, 1985.

[19] P. L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programming*. Pitman, 1986.

[20] G. Dowek *et. al.* The Coq proof assistant user's guide. Rapport Techniques 154, INRIA, Rocquencourt, France, 1993. Version 5.8.

[21] E. J. Dubuc and G. M. Kelly. A presentation of topoi as algebraic relative to categories or graphs. *Journal of Pure and Applied Algebra*, 81:420–433, 1983.

[22] T. Ehrhard. A categorical semantics of constructions. In *Logic in Computer Science*, pages 264–273, 1988.

[23] P. Gardner. *Representing Logics in Type Theory*. PhD thesis, University of Edinburgh, 1992.

[24] J. Giraud. *Cohomologie non abélienne*. Springer-Verlag, 1971.

[25] J. A. Goguen and J. Meseguer. Completeness of Many-Sorted Equational Logic. Technical Report CSLI-84-15, CENTER FOR THE STUDY OF LANGUAGE AND INFORMATION, 1984.

[26] R. Gordon and A. J. Power. Enrichment through variation. Technical Report 93-254, LFCS, 1993.

[27] J. W. Gray. *Formal Category Theory: Adjointness for 2-Categories*, volume 391 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1974.

[28] A. Grothendieck. Catégories fibrées et descente. In *Revêtements étales et groupe foundamental, Séminaire de Géométrie Algébriques du Bois-Marie 1960/61 (SGA 1)*. Exposé VI, 3rd ed., Institut des Haute Études Scientifiques, Paris, 1963. reprint, LNM 224, Springer-Verlag, Berlin,1971, pp.145-194.

[29] T. Hagino. A Typed Lambda Calculus with Categorical Type Constructors. In D.H. Pitt et.al., editor, *Category Theory and Computer Science (LNCS 240)*, pages 140–157. Springer Verlag, 1987.

[30] T. Hagino. *A Categorical Programming Language*. PhD thesis, University of Edinburgh, 1987.

[31] B. Harper and H. Pollack. Type checking with universes. *Theoretical Computer Science*, 89:107–136, 1991.

[32] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. In *Proc. Symposium on Logic in Computer Science*, pages 194–204, 1987.

[33] S. Hayashi. Adjunction of Semifunctors: Categorical Structures in Nonextensional Lambda Calculus. *Theo. Comp. Sci.*, 41:95–104, 1985.

[34] C. Hermida. *Fibrations, Logical Predicates and Indeterminates*. PhD thesis, University of Edinburgh, 1993.

[35] Martin Hofmann. *Extensional Concepts in Intensional Type Theory*. PhD thesis, University of Edinburgh, 1995.

[36] J.M.E. Hyland and A.M. Pitts. The theory of constructions: Categorical semantics and topos-theoretic models. In *Categories in Computer Science and Logic*, pages 137–199, 1989.

[37] B. Jacobs. *Categorical Type Theory*. PhD thesis, Catholic University of Nijmegen, 1991.

[38] B. Jacobs. Comprehension categories and the semantics of type dependency. *Theoretical Computer Science*, 107:169–207, 1993.

[39] C. Jones and S. Maharaj. The New LEGO Library, 1995. available as a part of the LEGO distribution, will be available as a LFCS technical report.

[40] G.M. Kelly. *Basic Concepts of Enriched Category Theory*. Cambridge University Press, 1982.

[41] J. Lambek. Deductive systems and categories II. In *Category Theory, Homology Theory and their Applications I*, volume 86 of *Lecture Notes in Mathematics*, pages 76–122. Springer-Verlag, Berlin, 1969.

[42] J. Lambek. Deductive systems and categories III. In *Toposes, Algebraic Geometry and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 57–82. Springer-Verlag, Berlin, 1972.

[43] J. Lambek. From λ-calculus to cartesian closed categories. In *To H.B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 374–402. Academic Press, London, 1980.

[44] J. Lambek and P.J. Scott. *Introduction to higher order categorical logic*. Cambride University Press, Cambridge, 1986.

[45] F. W. Lawvere. Metric spaces, generalized logic, and closed categories. *Rend. Sem. Mat. e Fis. di Milano*, XLIII:135–166, 1973.

[46] F.W. Lawvere. Functorial semantics of algebraic theories. *Proc. Nat. Acad. Sci. U.S.A*, 50:869–872, 1963.

[47] Z. Luo. *An Extended Calculus of Constructions*. PhD thesis, University of Edinburgh, 1990.

[48] Z. Luo and R. Pollack. LEGO proof development system: User's manual. Technical Report 92-211, LFCS, 1992.

[49] S. Mac Lane. *Categories for the working mathematician*, volume 5 of *Graduate Text Mathematics*. Springer-Verlarg, 1971.

[50] S. Mac Lane and I. Moerdijk. *Sheaves in Geometry and Logic*. Universitext. Springer-Verlag, New York, 1992.

[51] M. Makkai and G.E. Reys. *First Order Categorical Logic*, volume 611 of *Lecture Notes in Mathematics*. Springer Verlag, 1977.

[52] J. McKinna. *Deliverables: a categorical approach to program development in type theory*. PhD thesis, University of Edinburgh, 1992.

[53] J. McKinna and R. Pollack. Pure type systems formalized. In M.Bezem and J.F.Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications, TLCA'93*, pages 289–305. Springer-Verlag, LNCS 664, March 1993.

[54] M. V. Mendler. *A Modal Logic for Handling Behavioral Constraints in Formal Hardware Verification*. PhD thesis, University of Edinburgh, 1993.

[55] W. Mitchell. Boolean topoi and the theory of sets. *Journal of Pure and Applied Algebra*, 2:261–274, 1972.

[56] P. W. O'Hearn. A model for syntactic control of interference. *Mathematical Structures in Computer Science*.

[57] P. W. O'Hearn and R. D. Tennent. Semantics of local variables. In M. P. Fourman, P. T. Johnstone, and A. M. Pitts, editors, *Applications of Categories in Computer Science*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 217–238. Cambridge University Press, 1992.

[58] P. W. O'Hearn and R. D. Tennent. Relational parametricity and local variables. In *POPL93*, pages 171–184, 1993.

[59] R. Paré and D. Schumacher. Abstract families and the adjoint functor theorems. In P. T. Johnston and R. Paré, editors, *Indexed categories and their applications*, volume 661 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1978.

[60] L. C. Paulson. The Isabelle reference manual. Technical Report 283, University of Cambridge, Computer Laboratory, 1993.

[61] D. Pavlović. *Predicates and fibrations*. PhD thesis, Utrecht, 1990.

[62] R. Pollack. Incremental changes in LEGO: 1994, 1994. available as a part of the LEGO distribution from LFCS.

[63] R. Pollack. *The Theory of LEGO: A proof Checker for the Extended Calculus of Constructions*. PhD thesis, University of Edinburgh, 1994.

[64] R. Pollack and C. Jones. Incremental changes in LEGO, 1993. available as a part of the LEGO distribution from LFCS.

[65] A. J. Power. Why tricategories? Technical Report 94-289, LFCS, 1994.

[66] A. J. Power. Why tricategories? *Information and Computation*, To appear.

[67] A. J. Power and C. Hermida. Abstract control structure, 1995. Draft.

[68] Project Coq. the Coq Library, 1994. available as a part of the Coq distribution from INRIA.

[69] J. C. Reynolds. Syntactic control of interference, part 2. In G. Ausiello, M. D. Ciancaglini, and S. R. D. Rocca, editors, *Automata, Languages and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 704–722, 1989.

[70] E. Ritter. *Categorical Abstract Machines for Higher-Order Typed Lambda Calculi*. PhD thesis, University of Cambridge, 1992.

[71] D. E. Rydeheard and R. Burstall. *Computational Category Theory*. International Series in Computer Science. Prentice Hall, 1988.

[72] R.A.G. Seely. Hyperdoctrines, natural deduction and the beck condition. *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 29:505–542, 1983.

[73] R.A.G. Seely. Locally cartesian closed categories and type theory. *Math. Proc. Camb. Phil. Soc*, 95:33–48, 1984.

[74] R.A.G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *The Journal of Symbolic Logic*, 52:969–989, 1987.

[75] D. L. Spencer. *Categorical Programming with Functorial Strength*. PhD thesis, Oregon Graduate Institute of Science & Technology, 1993.

[76] R. Street. The formal theory of monads. *Journal of Pure and Applied Algebra*, 2:149–168, 1972.

[77] R. Street. Cosmoi of internal categories. *Transactions of the American Mathematical Society*, 258(2):271–318, 1980.

[78] R. Street. Some parameterized categorical concepts. *Communications in Algebra*, 16(11):2313–2347, 1988.

[79] R. Street. Enriched categories and cohomology. *Questiones Mathematicae*, 6:265–283, 83.

[80] R. H. Street. Elementary cosmoi. I. In *Category seminar Sydney 1972/73*, volume 420 of *Lecture Notes in Mathematics*, pages 134–180. Springer-Verlag, Berlin, 1974.

[81] R. H. Street and R. F. C. Walters. Yoneda structures on 2-categories. *Journal of Algebra*, 50:350–379, 1978.

[82] T. Streicher. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions*. PhD thesis, Passau University, 1989.

[83] M. E. Szabo. *Algebra of proofs*, volume 88 of *Studies in Logic and The Foundations of Mathematics*. North-Holland, Amsterdam, 1978.

[84] M. Takeyama. A type system based on locally small fibrations, 1990. Master's Thesis, University of Tokyo.

[85] A. Tarlecki, R. M. Burstall, and J. A. Goguen. Some fundamental algebraic tools for the semantics of computation part iii: Indexed categories. Technical Report 88-60, LFCS, 88.

[86] P. Taylor. *Recursive Domains, Indexed Categories and Polymorphism*. PhD thesis, University of Cambridge, 1986.

[87] R. D. Tennent. Semantics of interference control. *Theoretical Computer Science*, 27:297–310, 1983.

[88] R. D. Tennent. Semantical analysis of specification logic. *Information and Computation*, 85(2):135–162, 1990.

[89] R. D. Tennent. Semantical analysis of specification logic, 2. *Information and Computation*, 107(1):25–57, 1993.