A Relevant Analysis of Natural Deduction

Ishtiaq, Samin

ISSN 1369-1961

Department of Computer Science

Technical Report No. 762

A Relevant Analysis of Natural Deduction

Samin Ishtiaq

QUEEN MARY
AND WESTFIELD COLLEGE
UNIVERSITY OF LONDON

April 1999

A Relevant Analysis of Natural Deduction

Samin Ishtiaq

Submitted for the degree of Doctor of Philosophy

Queen Mary and Westfield College

University of London

April, 1999

A Relevant Analysis of Natural Deduction

Samin Ishtiaq

Abstract

Linear and other relevant logics have been studied widely in mathematical, philosophical and computational logic. We describe a logical framework, RLF, for defining natural deduction presentations of such logics. RLF consists in a language together, in a manner similar to that of Harper, Honsell and Plotkin's LF, with a representation mechanism: the language of RLF is the $\lambda\Lambda$-calculus; the representation mechanism is judgements-as-types, developed for relevant logics.

The $\lambda\Lambda$-calculus type theory is a first-order dependent type theory with two kinds of dependent function spaces: a linear one and an intuitionistic one. We study a natural deduction presentation of the type theory and establish the required proof-theoretic meta-theory. The RLF framework is a conservative extension of LF. We show that RLF uniformly encodes (fragments of) intuitionistic linear logic, Curry's λ_I-calculus and ML with references.

We describe the Curry-Howard-de Bruijn correspondence of the $\lambda\Lambda$-calculus with a structural fragment of O'Hearn and Pym's logic BI of bunched implications.

We show that a categorical semantics of the $\lambda\Lambda$-calculus is given by Kripke resource models, which are a monoid-indexed set of Kripke functors. The indexing monoid is seen as providing an account of resource consumption. The models can be seen as an indexed formulation of categorical models of Read's bunches.

We also study a Gentzenized version of the $\lambda\Lambda$-calculus, and prove a cut-elimination theorem for it.

Submitted for the degree of Doctor of Philosophy

Queen Mary and Westfield College
University of London

April, 1999

Acknowledgements

I thank my supervisor David Pym, from whom I have learned much and whose help and support has been of great importance. I also thank the other members of my supervisory committee, Edmund Robinson and Wilfrid Hodges, in this regard.

I thank Pablo Armelín, Adam Eppendahl, Paul Blain Levy and David Streader for their advice, suggestions and support.

This work was supported by an EPSRC Mathfit research studentship and, in the last stages, by the Department of Computer Science. The EU *Types* Working Group provided funds to attend Types '96 in Aussois, France and Types '98 in Irsee, Germany. All diagrams were drawn with Paul Taylor's diagram and proof tree packages.

Contents

List of Tables

Chapter 1

Introduction

In this chapter we introduce the notion of relevant logical frameworks and the study of the proof theory, internal logic and categorical model theory of the type-theoretic language of one such framework by introducing the subsequent chapters of this thesis.

The starting point of this thesis is the study of a logical framework for a particular class of logics; that of linear and relevant logics. Such a general theory of logics has two main motivations. Firstly, the desire to study the common "core" of these logics. And, secondly, to provide a single meta-language, capable of representing the class of such logics, which can be implemented on a machine. The implementation of this meta-language then automatically provides implementations for the class of logics [HHP93].

The logical framework that we study has as its meta-language the $\lambda\Lambda$-calculus, a first-order dependent type theory with two kinds of dependent function spaces, a linear one and an intuitionistic one. This thesis concentrates on the semantics, both proof- and model-theoretic, of the type theory.

We should like to explain our use of the word "relevant" throughout and in the title of this thesis. Following Read [Rea88], we use the term relevant for the family of logics which have weaker structural properties than intuitionistic or classical logic. We do not restrict its use merely for those logics which have contraction but not weakening.

We refer to related work at the appropriate places in the text.

The $\lambda\Lambda$-calculus and the RLF logical framework

Consider the following situation: in an ambient universe, one uses a meta-language to reason or compute about some (possibly formal) system. Examples of such a general situation abound. For example, in mathematics, one uses logic to reason about group theory. Another example, and one that is at the heart of logical frameworks, is this: in informatics, one uses a low-level language to implement a class of high-level languages, exploiting the computational characteristics of the low-level language, and the correct compilation of the high-level languages into the low-level language, to execute the high-level languages.

Logical frameworks are formal meta-logics which provide languages for describing logics in a manner that is suitable for mechanical implementation. The LF logical framework [AHMP92, HHP93, Pym90] provides such a meta-theory and is suitable for logics which have at least the structural strength of minimal propositional logic. We wish to study a logical framework for describing relevant logics. Such a class of logics is interesting from both a philosophical–logical and from a computer science point of view. Now, in order to describe a logical framework one must :

1. Characterize the class of object-logics to be represented;
2. Give a meta-logic or language, together with its meta-logical status *vis-à-vis* the class of object-logics; and
3. Characterize the representation mechanism for object-logics.

The above prescription can conveniently be summarized by the slogan

$$Framework = Language + Representation.$$

One representation mechanism is that of judgements-as-types, which originates from Martin-Löf's development of Kant's notion of judgement [ML96]. The methodology of judgements-as-types is that judgements are represented as the type of their proofs. A logical system is represented by a signature which assigns kinds and types to a finite set of constants that represent its syntax, its judgements and its rule schemes. Representation theorems relate consequence in an object-logic $\vdash_L$ to consequence in an encoded logic $\vdash_{\Sigma_L}$.

A certain class of uniform representations [HST94] is identified by considering surjective encodings between consequences of the object-logic $\vdash_L$ and consequences of the meta-logic $\vdash_{\Sigma_L}$.

So, all judgements in the meta-logic have corresponding judgements in the object-logic. The judgement-as-types methodology has the property that encoded systems inherit the structural properties of the meta-logic. It is for this reason that LF — whose language, the λΠ-calculus, admits weakening and contraction — cannot uniformly encode linear and other relevant logics.

This limitation can be overcome in several ways. One way is to add equations to the signature that constrain the judgement *ex post facto*. This is the method adopted in Avron *et al.* [AHMP92] where, in the encoding of the linear λ-calculus in LF, the linearity constraint is enforced by introducing extraneous language to axiomatize the concept of strictness in domain theory. Another way to overcome this limitation is to weaken the framework by working with a weaker framework language. This approach is more natural as it brings out the fact that the structural strength of the framework effectively reflects the structural strength of the class of object logics it encodes. And that, in order to represent and reason about an object logic of a particular strength, one must endow the framework with an appropriate level of structure. This second method can be seen as an application of what the physicist John Baez calls the *microcosm principle* [BD97].

Thus we seek a language in which weakening and contraction are not forced. We motivate the connectives of the language by considering the natural deduction form of rules for relevant logics. As a specific example for this chapter, we consider the natural deduction rules for the ⊗ and & connectives of linear logic, in which the sole judgement is concerned with the *proof* of a proposition:

$$\dfrac{\begin{array}{c}\Gamma\\ \vdots\\ \phi\ \mathit{proof}\end{array}\qquad\begin{array}{c}\Delta\\ \vdots\\ \psi\ \mathit{proof}\end{array}}{\phi\otimes\psi\ \mathit{proof}}\ \otimes I \qquad\qquad \dfrac{\begin{array}{c}\Gamma\\ \vdots\\ \phi\otimes\psi\ \mathit{proof}\end{array}\qquad\begin{array}{c}\Delta,[\phi,\psi]\\ \vdots\\ \chi\ \mathit{proof}\end{array}}{\chi\ \mathit{proof}}\ \otimes E$$

$$\dfrac{\begin{array}{c}\Gamma\\ \vdots\\ \phi\ \mathit{proof}\end{array}\qquad\begin{array}{c}\Gamma\\ \vdots\\ \psi\ \mathit{proof}\end{array}}{\phi\&\psi\ \mathit{proof}}\ \& I \qquad\qquad \dfrac{\begin{array}{c}\Gamma\\ \vdots\\ \phi_0\&\phi_1\ \mathit{proof}\end{array}}{\phi_i\ \mathit{proof}}\ (i\in\{0,1\})\ \& E$$

Table 1.1: A fragment of ILL

In order to distinguish the encoding of the two introduction (or elimination) rules, we need two kinds of conjunctions; an additive one and a multiplicative one. We also need a connective to represent inference in the object-logic. The representation of the rules as constants in the signature Σ_{ILL} is then as follows:

$$\begin{array}{rcl}
\text{TENSOR-I} & ! & \Pi\phi,\psi{:}o\,.\,proof(\phi) \multimap proof(\psi) \multimap proof(\otimes(\phi,\psi)) \\
\text{TENSOR-E} & ! & \Pi\phi,\psi,\chi{:}o\,.\,proof(\otimes(\phi,\psi)) \multimap \\
 & & \quad (proof(\phi) \multimap proof(\psi) \multimap proof(\chi)) \multimap proof(\chi) \\
\text{WITH-I} & ! & \Pi\phi,\psi{:}o\,.\,proof(\phi)\,\&\,proof(\psi) \multimap proof(\&(\phi,\psi)) \\
\text{WITH-E}_i & ! & \Pi\phi_0,\phi_1{:}o\,.\,proof(\&(\phi_0,\phi_1)) \multimap proof(\phi_i)
\end{array}$$

In the above, o is the type of propositions, and the constructors $\otimes$ and $\&$ are both of type $o \multimap o \multimap o$. In fact, the $\otimes$ connective is not needed as part of the meta-language, as it can always be curried away to the $\multimap$, the connective which represents inference. The Π quantification is intuitionistic. However, in logics that describe more resource-conscious situations, such as the the dynamic semantics of ML with references, we need a linear quantification Λ too.

The above analysis can be undertaken more generally, by considering Prawitz's [Pra78] general form of schematic introduction and elimination rules from a more relevant point of view. The resulting meta-language, the $\lambda\Lambda$-calculus, has the following connectives: the additive conjunction $\&$; two quantifiers, the linear one Λ and the intuitionistic one Π; and the two implications arising as non-dependent versions of the quantifiers.

The $\lambda\Lambda$-calculus is a first-order dependent type theory with both intuitionistic and linear dependent function spaces. The basic judgement of the type theory is $\Gamma \vdash_\Sigma M{:}A$, that M is a term of type A in context Γ and signature Σ. The definition relies crucially on a few notions about the joining and maintenance of contexts. In particular, full dependency can only be set up by a notion of "sharing", which deals with the well-formedness of linear dependent types.

The introduction rules for the two function spaces, the linear one $\Lambda x{:}A\,.B$ and the intuitionistic one $\Lambda x{!}A\,.B$ (which is just the notation for $\Pi x{:}A\,.B$) are as follows:

$$\frac{\Gamma, x{:}A \vdash_\Sigma M{:}B}{\Gamma \vdash_\Sigma \lambda x{:}A\,.M : \Lambda x{:}A\,.B}\,(M\lambda\Lambda I) \qquad \frac{\Gamma, x{!}A \vdash_\Sigma M{:}B}{\Gamma \vdash_\Sigma \lambda x{!}A\,.M : \Lambda x{!}A\,.B}\,(M\lambda\Lambda!I)$$

This preview of the rules introduces a key characteristic of the type theory, that of building the context with either linear, $x{:}A$, or intuitionistic, $x{!}A$, variables. In particular, there is no zone or "stoup" separating the linear from the intuitionistic parts of the context.

The judgement $\Gamma \vdash_\Sigma M{:}A$ is decidable. The proof of this statement involves proving Church-Rosser and Strong Normalization, along with several other essential meta-theoretic properties.

In Chapter 2, we motivate the RLF logical framework and give a natural deduction presentation of its language, the $\lambda\Lambda$-calculus. We establish the required proof-theoretic meta-theory of the language. We show that RLF is a conservative extension of LF; that RLF uniformly encodes everything that LF encodes. We also present several object-logic encodings that make use of the characteristics of the $\lambda\Lambda$-calculus. A large part of Chapter 2 has already appeared as:

[IP98] SS Ishtiaq and DJ Pym. A Relevant Analysis of Natural Deduction. *Journal of Logic and Computation*, 8(6):809–838, 1998.

We make a brief remark concerning the choice and methodology of logical frameworks. Earlier, we mentioned logic as a language for mathematics and our purpose in this thesis is to develop a type-theoretic logical framework for representing certain classes of formal systems. We do so without any logicist intention or claim. One can consider linguistic, algebraic or other frameworks too. But it is important to stress the methodology of type-theoretic logical frameworks as evidenced by the work on LF [HHP93] and, indeed, earlier, from Automath [dB91]. This is to adopt a principle of weakness: to require a mechanism for binding and a minimal, decidable meta-language — and no more. One can always add strength to the meta-language but it is more interesting to study the weak situations first.

The propositions-as-types correspondence

The $\lambda\Lambda$-calculus type theory is motivated by a consideration, *inter alia*, of linear logic. However, it is structurally much closer to **BI**, the logic of bunched implications. Linear logic begins by removing the structural rules and querying the consequences for the conjunction $\wedge$, which decomposes into the additive conjunction & and the multiplicative conjunction $\otimes$. **BI**, in contrast, begins by decomposing the implication directly, rather than in terms of the conjunction.

In **BI**, we have two kinds of function spaces, the linear one $\multimap$ and the intuitionistic one $\rightarrow$ and, correspondingly, two kinds of quantifiers, the linear one $\forall_{\mathrm{new}}$ and the intuitionistic one $\forall$. Proof-theoretically, these arise because of extra structure in the context. There are two distinct context-formation operators, the ";", which admits the structural rules of weakening and contraction, and the ",", which doesn't. In this scheme, contexts are not lists but trees, with internal nodes labelled by "," and ";". The introduction rules for the two implications are as follows:

$$\frac{(X)\Gamma,\phi \vdash \psi}{(X)\Gamma \vdash \phi \multimap \psi} \qquad \frac{(X)\Gamma;\phi \vdash \psi}{(X)\Gamma \vdash \phi \rightarrow \psi}$$

There is a similar pair of rules for the two quantifiers in **BI**; the bunched structure in the variable context X determines which type of quantifier, whether linear or intuitionistic, is formed by the abstraction. In eliminating the connectives, the premiss contexts must be combined with regard to the type (whether linear or intuitionistic) of the connective.

The conceptual similarity with the $\lambda\Lambda$-calculus arises because contexts there can also be seen to be extended in two ways:

$\lambda\Lambda$-calculus	**BI**
$\Gamma, x{:}A$	Γ, A
$\Gamma, x!A$	$\Gamma; A$

Subject to some restrictions on **BI** (so that we are working with a fragment of the original logic), this is exactly what allows us to show the Curry-Howard-de Bruijn correspondence between the $\lambda\Lambda$-calculus and **BI**. The statement of this result and its proof is given in Chapter 3.

BI is probably a folklore relevant logic though it is only recently that a decent proof theory, semantics and computational interpretation have been given [Dun86, Rea88, OP99].

Kripke resource models

In Chapter 4, we study the categorical semantics of the type theory. The nature of the class of these models can be motivated by considering the notion of "resource", a primitive in informatics. A resource, such as space or time, can be seen to have a monoidal structure, in the sense that we can identify elements (including the null one) and combinations of resources. We can also consider a comparison of resources, in the sense of one being better — perhaps in the sense of being able to prove more facts — than another. A resource semantics elegantly explains the difference between the linear and intuitionistic connectives in that the action, or computation, of the linear connectives can be seen to consume resources. Let $\mathcal{M} = (M, \cdot, 1, \sqsubseteq)$ be such a resource monoid with, in addition, a partial ordering on the elements, then we can give the following (simplified) version of the forcing relation for the internal logic of the type theory:

1. $r \models \phi \multimap \psi$ if and only if for all $s \in \mathcal{M}$ if $s \models \phi$ then $r \cdot s \models \psi$;

2. $r \models \phi \rightarrow \psi$ if and only if for all $s \sqsubseteq r$ if $s \models \phi$ then $s \models \psi$.

The resource semantics combines and generalizes Kripke's semantics for intuitionistic logic, in which a proposition is interpreted in $\mathbf{Set}^P$, where P is a poset, and Urquhart's semantics for relevant logic, in which a proposition is interpreted $\mathbf{Set}^{\mathcal{M}}$, where $\mathcal{M}$ is a monoid [Kri65, Urq72].

The semantics of the $\lambda\Lambda$-calculus is given by Kripke resource models, which are a monoid-indexed set of Kripke functors. Each such Kripke functor is, roughly speaking, an indexed category with some extra structure. Two items of this extra structure are worth mentioning. The first is that the base of the indexed category has two monoidal structures on it. This allows us to model the two kinds of context extension in the syntax: $\Gamma, x{:}A$ is interpreted as $[\![\Gamma]\!] \otimes [\![A]\!]$ and $\Gamma, x{!}A$ is interpreted as $[\![\Gamma]\!] \times [\![A]\!]$. The second item is the existence of a natural isomorphism which allows the formation of function spaces. This is necessary to model the linear dependent function space. The intuitionistic dependent function space is defined by the natural isomorphism too; this is equivalent to the usual characterization of the function space as right adjoint to weakening.

We include two example Kripke resource models. The first is the term model, constructed out of the syntax of the type theory. The main purpose behind its construction is to show a Henkin-style completeness theorem. The second model, a set-theoretic one built in the spirit of Cartmell and Streicher's contextual category of families of sets [Car86, Str88], is much more interesting. The model constructs two kinds of indexed products in $\mathbf{Set}^{C^{op}}$, where C is a small monoidal category. The first is the usual cartesian product, as described in Cartmell for instance. The second is a restriction of Day's tensor product [Day70].

The Kripke resource models, especially the set-theoretic model, can also be seen as an indexed formulation of categorical models of Read's bunches.

The Gentzenized $\lambda\Lambda$-calculus

The presentation of the $\lambda\Lambda$-calculus in Chapter 2 is as a linearized natural deduction system. In Chapter 5, we present a Gentzenized version of the natural deduction system, in which the $(M\Lambda\mathcal{E})$ and $(M\Lambda!\mathcal{E})$ elimination rules (and their non-dependent versions) are replaced by rules which introduce the connectives into the context, or left-hand side, of the sequent. Consider the linear case. The $(M\Lambda\mathcal{E})$ rule of the natural deduction system is as follows:

$$\frac{\Gamma \vdash_\Sigma M{:}\Lambda x{:}A\,.B \quad \Delta \vdash_\Sigma N{:}A}{\Xi \vdash_\Sigma MN{:}B[N/x]}$$

where the context Ξ is formed by the sharing-sensitive join of the premiss contexts Γ and Δ. In the sequent calculus system, this rule is replaced by the following $({:}\,\Lambda L)$ rule:

$$\frac{\Gamma \vdash_\Sigma N{:}B \quad \Delta, x{:}D \vdash_\Sigma M{:}A}{\Xi, y{:}\Lambda z{:}B\,.C \vdash_\Sigma M[yN/x]{:}A}\;({:}\,\Lambda L)$$

where y is new, $x \notin FV(A)$ and $D \equiv C[N/z]$. The $(:\Lambda L)$ rule is the most obvious candidate for the Gentzenizing of $(M\Lambda\mathcal{E})$, but is not the only one. We can also consider the rule where the y intuitionistically extends Ξ. A similar analysis applies to the $(M\Lambda!\mathcal{E})$ rule.

The completeness of the sequent calculus version of the λΛ-calculus with respect to the natural deduction version depends crucially on the presence of the two cut rules which, as they are not admissible, are taken explicitly in the sequent calculus system. However, by working with β-normal forms of proofs, we can prove a cut-elimination theorem.

Chapter 5 will allow us to begin the study of proof-search in the type theory and, hence, in uniformly-encoded object-logics of the RLF logical framework.

Chapter 2

The $\lambda\Lambda$-calculus and the RLF logical framework

2.1 Introduction

Linear and other relevant logics have been studied widely in mathematical [Avr88, Gir87, Mey76, SH91], philosophical [AB75, Dun86, Rea88] and computational [Abr93, HM94, MTV93, PH94, Ten92] logic. We present a study of a logical framework, RLF, for defining natural deduction presentations of such logics.[1] RLF consists in a language together, in a manner similar to that of LF [AHMP92, HHP93, Pym90], with a representation mechanism. The language of RLF, the $\lambda\Lambda$-calculus, is a system of first-order linear dependent function types which uses a function κ to describe the degree of sharing of variables between functions and their arguments. The representation mechanism is judgements-as-types, developed for linear and other relevant logics.

We motivate the $\lambda\Lambda$-calculus by considering an abstract form of relevant natural deduction. We specify the $\lambda\Lambda$-calculus, a family of first-order dependent type theories with both linear and intuitionistic function spaces, discussing only briefly the possible intermediate systems. The framework RLF is a conservative extension of LF; the notion of conservative extension takes account of the representation mechanism as well as the type theory. The work reported here builds on ideas presented by Pym in [Pym92].

Recall that we use the word "relevant" according to Read's description; for the family of logics which have weaker structural properties than intuitionistic or classical logic, not merely for those which have contraction but not weakening. Read's taxonomy would place linear logic

[1]RLF, in common with LF, is also able to define Hilbert-type systems, although this is beyond our present scope.

(without exponentials) at the lowest point in the "lattice" (we use the word informally and not in any technical sense) of logics. We follow this taxonomy and thus obtain a lattice of logical frameworks, the weakest being RLF, the type theory of which has neither weakening nor contraction.[2] We emphasize that the λΛ-calculus lies properly in the world of relevant logics: the type theory's contexts are a dependently-typed notion of Read's *bunches* [Rea88]. Our framework RLF provides a relevant analysis of natural deduction just as LF provides an intuitionistic analysis of natural deduction. We do not study the variety of distributivity laws usually considered for relevant contexts [Rea88]. However, such an investigation should fit into our framework, possibly via variations on the λΛ-calculus, quite straightforwardly.

This chapter is organized as follows. In Section 2.2, we motivate the λΛ-calculus in the context of a logical framework by considering an abstract form of relevant natural deduction. We formally define it as a type theory and summarize its meta-theory in Section 2.3, concluding the section with a comparison with related work. In Section 2.4, we show that RLF is a conservative extension of LF. In Section 2.5, we illustrate several example encodings in the RLF framework. The object-logics we consider are a fragment of propositional intuitionistic linear logic, the dynamic semantics of ML with references and a relevant λ-calculus.

2.2 Motivation

Logical frameworks are formal meta-logics which, *inter alia*, provide languages for describing logics in a manner that is suitable for mechanical implementation. The LF logical framework [AHMP92, HHP93, Pym90] provides such a metatheory and is suitable for logics which have at least the structural strength of minimal propositional logic. We wish to study a logical framework for describing relevant logics. Now, in order to describe a logical framework one must:

1. Characterize the class of object-logics to be represented;
2. Give a meta-logic or language, together with its meta-logical status *vis-à-vis* the class of object-logics; and
3. Characterize the representation mechanism for object-logics.

The above prescription can be conveniently summarized by the slogan

$$\mathit{Framework} = \mathit{Language} + \mathit{Representation}.$$

[2]In the literature, the terms "sub-structural" and "weak" are sometimes used in this way.

We remark that these components are not entirely independent of each other [Pym96]. We will point out some interdependencies later in this section.

One representation mechanism is that of judgements-as-types, which originates from Martin-Löf's [ML96] development of Kant's [Kan00] notion of judgement. The two higher-order judgements, the hypothetical $J \vdash J'$ and the general $\bigwedge_{x \in C} . J(x)$, correspond to ordinary and dependent function spaces, respectively. The methodology of judgements-as-types is that judgements are represented as the type of their proofs. A logical system is represented by a signature which assigns kinds and types to a finite set of constants that represent its syntax, its judgements and its rule schemes. An object-logic's rules and proofs are seen as proofs of hypothetico-general judgements $\bigwedge_{x_1 \in C_1} \cdots \bigwedge_{x_m \in C_m} . J \vdash J'$. Representation theorems relate consequence in an object-logic $\vdash_L$ to consequence in an encoded logic $\vdash_{\Sigma_L}$:

$$\begin{array}{ll} (X, J_1(\phi_1), \ldots, J_m(\phi_m)) \vdash_L \delta : J(\phi) & \mathit{object-consequence} \\ \Downarrow & \mathit{encoding} \\ \Gamma_X, x_1{:}J_1(\phi_1), \ldots, x_m{:}J_m(\phi_m) \vdash_{\Sigma_L} M_\delta : J(\phi) & \mathit{meta-consequence}, \end{array}$$

where X is the set of variables that occur in ϕ_i, ϕ; J_i, J are judgements; δ is a proof-object (*e.g.*, a λ-term); Γ_X corresponds to X; each x_i corresponds to a place-holder for the encoding of J_i; and M_δ is a meta-logic term corresponding to the encoding of δ.

In the sequel, we do not consider the complete apparatus of judged object-logics. Our example encodings in Section 2.5 are pathological in the sense that they require only one judgement. For example, the encoding of a fragment of intuitionistic linear logic requires the judgement of $(J_i = J =)$ proof. This is in contrast to the general multi-judgement representation techniques [AHMP98]. We conjecture that our studies can be applied to the general case, although we defer this development to another occasion.

A certain class of uniform representations is identified by considering surjective encodings between consequences of the object-logic $\vdash_L$ and consequences of the meta-logic $\vdash_{\Sigma_L}$ [HST94].[3] So, all judgements in the meta-logic have corresponding judgements in the object-logic. The judgement-as-types methodology has the property that encoded systems inherit the structural properties of the meta-logic. It is for this reason that LF — whose language, the $\lambda\Pi$-calculus, admits weakening and contraction — cannot uniformly encode linear and other relevant logics.

[3]The specification in [HST94] is a stronger one, requiring uniformity over all "presentations" of a given logic. Such concerns are beyond our present scope.

To illustrate this point, suppose Σ_{ILL} is a uniform encoding of intuitionistic linear logic in LF, and that $\Gamma_X, \Gamma_\Delta \vdash_{\Sigma_{ILL}} M_\delta{:}J(\phi)$ is the image of the object-consequence $(X,\Delta) \vdash_{ILL} \delta{:}J(\phi)$. If $\Gamma_X, \Gamma_\Delta \vdash_{\Sigma_{ILL}} M_\delta{:}J(\phi)$ is provable, then so is $\Gamma_X, \Gamma_\Delta, \Gamma_\Theta \vdash_{\Sigma_{ILL}} M_\delta{:}J(\phi)$. By uniformity, the latter is the image of an object-logic consequence $(X,\Delta,\Theta) \vdash_{ILL} \delta'{:}J(\phi)$, which implies weakening in linear logic, a contradiction.

Thus we seek a language in which weakening and contraction are not forced. We motivate the connectives of the language by considering the natural deduction form of rules for weak logics. We do this in a general way, by considering Prawitz's general form of schematic introductions from a more relevant point of view. Prawitz [Pra78] gives these for intuitionistic logic. A schematic introduction rule for an n-ary sentential operator # is represented by an introduction rule of the form below. In the rule, only the bound assumptions for G_j are shown; we elide those for G_k, where $(k \neq j)$, for the sake of readability.

$$\dfrac{\begin{array}{ccccc} & & [H_{j,1}]\cdots[H_{j,h_j}] & & \\ \vdots & & \vdots & & \vdots \\ G_1 & \cdots & G_j & \cdots & G_p \end{array}}{\#(F_1,\ldots,F_n)}$$

In the above rule, $1 \leq j \leq p$. The Fs, Gs and Hs are formulae constructed in the usual way. An inference infers a formula $\#(F_1,\ldots,F_n)$ from p premisses $G_1,\ldots,G_p$ and may bind assumptions of the form $H_{j,1},\ldots,H_{j,h_j}$ that occur above the premiss G_j. We let the assumptions be multi-sets, thus keeping the structural rule of exchange. We require that discharge be compulsory. In the case of the natural deduction presentation of intuitionistic linear logic, for instance, we require that $\{F_1,\ldots,F_n\} = \{G_j, H_{j,1}\ldots,H_{j,h_j}\}$. For example, in the rule for $\multimap$-introduction, whose conclusion is $\phi \multimap \psi$, we have $\{F_1,F_2\} = \{\phi,\psi\}$, $G_1 = \psi$ and $H_{1,1} = \phi$.

We annotate the introduction schema below to indicate our method of encoding. o is the type of propositions, the Λ is the linear universal quantifier and Π is the intuitionistic universal quantifier. So we quantify over a linear proposition as $\Lambda F{:}o$ and over an intuitionistic proposition as $\Pi G{:}o$. We also use $\Lambda G!o$ for the latter and $\Lambda F{\in}o$ to range over both linear and intuitionistic quantifications. Each inference — that is, the binding of assumptions $H_{j,1},\ldots,H_{j,h_j}$ above premiss G_j and the inference of formula $\#(F_1,\ldots,F_n)$ from premisses $G_1,\ldots,G_p$ — is represented by a $\multimap$.

$$\bigwedge F_g, G_j, H_{j,k} \in o \qquad \dfrac{\begin{array}{ccccc} & & [H_{j,1}]\Box\cdots\Box[H_{j,h_j}] & & \\ \vdots & & \vdots & & \vdots \\ G_1\ \Box & \cdots & \Box\ G_j\ \Box & \cdots & \Box\ G_p \end{array}}{\#(F_1,\ldots,F_n)}$$

The premisses $G_1, \ldots, G_p$ are combined either multiplicatively or additively, depending on whether their contexts are disjoint or not. We distinguish between these combinations by the use of two conjunctions; the multiplicative $\otimes$ ("tensor") and the additive & ("with") and so force the structural rules. (In traditional relevance logics, multiplicative is referred to as intentional and additive as extensional.) We use $\Box$ as meta-syntax for both $\otimes$ and &, though mindful of the relationship between the two operators. Full expressivity is recovered by introducing the modality ! ("bang") into our language. The premiss $!G$ allows us to depart from relevant inference, and to choose the number of times we use G in the conclusion.

In the meta-logic, then, the schematic introduction rule would be represented by a constant of the following type:

$$\bigwedge F_g, G_j, H_{j,k} \in o. \ldots \Box(\Box_{l \leq h_j}(H_{j,l}) \multimap G_j)\Box \ldots \multimap \#(F_1,\ldots,F_n),$$

where $1 \leq l \leq h_j$ and $\Box_{l \leq h_j}$ represents an iterated $\Box$. From the general encoding formula above, it can be seen that the connectives $\Box$ (*i.e.*, $\otimes$ and &) and ! occur only negatively. In the tensor's case, this allows us to curry away the $\otimes$, modulo a permutation of the premisses. For example, in the following type, we are able to replace the occurrence of $\otimes$,

$$(\Box_{l \leq h_j}(H_{j,l}) \multimap G_j) \otimes (\Box_{l' \leq h_{j'}}(H_{j',l'}) \multimap G_{j'}) \multimap \#(F_1,\ldots,F_n),$$

by a $\multimap$,

$$(\Box_{l \leq h_j}(H_{j,l}) \multimap G_j) \multimap (\Box_{l' \leq h_{j'}}(H_{j',l'}) \multimap G_{j'}) \multimap \#(F_1,\ldots,F_n).$$

We can also consider a currying away of the & by a non-dependent version of the additive function space. A language with two kinds of dependent function space is very interesting but is beyond the scope of our current study.

We recapitulate exactly how we have used the three logical constants in the framework: the & is used to undertake additive conjunction; the Λ is used to quantify and (in its non-dependent form ⊸) to represent implication; and the Π is used to represent dereliction from relevant inference. We should then be able to formulate a precise idea regarding the completeness of the set $\{\&, \Lambda, \Pi\}$ with respect to all sentential operators that have explicit schematic introduction rules [Pra78, SH83].

A similar analysis can be undertaken for the corresponding elimination rule.

Our analysis allows us two degrees of freedom. The first is at the structural level of types. In this section, our main intention has been to motivate a language in which the structural rules of weakening and contraction are not forced, and so to be able to uniformly encode linear logic. But this language is only one of a range of relevant logics [Rea88], which includes, for instance, Anderson and Belnaps's relevance logic [AB75]. Choosing a different language, with it's particular structural and distributivity properties, would allow us to uniformly encode another class of logics. The family of relevant logics determined by these choices is very interesting from a representational perspective, though we pursue it no further in this thesis.

The second, orthogonal, degree of freedom, and one that we do concentrate on in the sequel, concerns the corresponding range of structural choices at the level of terms (as opposed to types). Considering this aspect from the logical point of view, we consider multiple occurrences of the same proof. The degree to which a proof can be shared by propositions is a structural property which determines, via the Curry-Howard-de Bruijn correspondence, a type theory whose functions and arguments share variables to a corresponding degree.

The language that we have motivated in this section, and develop in the sequel, is a type theory in Curry-Howard-de Bruijn correspondence with a structural fragment of **BI**, the logic of bunched implications. The details of this correspondence are given in the next chapter.

2.3 The λΛ-calculus

The λΛ-calculus is a first-order dependent type theory with both intuitionistic and linear function types. The calculus is used for deriving typing judgements. There are three entities in the λΛ-calculus: *objects*, *types* and *families of types*, and *kinds*. Objects (denoted by M, N) are classified by types. Families of types (denoted by A, B) may be thought of as functions which map objects to types. Kinds (denoted by K) classify families. In particular, there is a kind Type which classifies the types. We will use U, V to denote any of the entities.

We assume given three disjoint, countably infinite sets: the meta-variables x, y, z range over the set of variables; c, d range over the set of object-level constants; and a, b range over the set of type-level constants. The abstract syntax of the λΛ-calculus is given by the following grammar:

$$\begin{array}{llll} \textit{Kinds} & K & ::= & \mathsf{Type} \mid \Lambda x{:}A\,.K \mid \Lambda x!A\,.K \\ \textit{Types} & A & ::= & a \mid \Lambda x{:}A\,.B \mid \Lambda x!A\,.B \mid \lambda x{:}A\,.B \mid \lambda x!A\,.B \mid AM \mid A\&B \\ \textit{Objects} & M & ::= & c \mid x \mid \lambda x{:}A\,.M \mid \lambda x!A\,.M \mid MN \mid \langle M,N\rangle \mid \pi_0 M \mid \pi_1 M\,. \end{array}$$

We write $x{\in}A$ to range over both linear ($x{:}A$) and intuitionistic ($x!A$) variable declarations. The λ and Λ bind the variable x. The object $\lambda x{:}A\,.M$ is an inhabitant of the linear dependent function type $\Lambda x{:}A\,.B$. The object $\lambda x!A\,.M$ is an inhabitant of the type $\Lambda x!A\,.B$, which amounts to the Martin-Löf-style $\Pi x{:}A\,.B$. The notion of linear free- and bound-variables (LFV, LBV) and substitution may be defined accordingly [Bie94]. When x is not free in B we write $A \multimap B$ and $A \rightarrow B$ for $\Lambda x{:}A\,.B$ and $\Lambda x!A\,.B$, respectively. Our basic study does not include the units, but $\top$ and 1 can be added to the type theory with little difficulty.

We can define the notion of linear occurrence by extending the general idea of occurrence for the λ-calculus [Bar84], though we note that other definitions may be possible.

Definition 1 (linear occurrence in U)

1. *x linearly occurs in x;*

2. *If x linearly occurs in U or V (or both), then x linearly occurs in $\lambda y{\in}U\,.V$, in $\Lambda y{\in}U\,.V$, and in UV, where $x \neq y$;*

3. *If x linearly occurs in both M and N, then x linearly occurs in $\langle M,N\rangle$;*

4. *If x linearly occurs in M, then x linearly occurs in $\pi_i(M)$;*

5. *If x linearly occurs in both A and B, then x linearly occurs in $A\&B$.*

The definition is extended to an inhabited type and kind.

Definition 2 (linear occurrence in $U{:}V$) *A variable x linearly occurs in the expression $U{:}V$ if it linearly occurs in U, in V, or in both.*

We remark that the above definitions are not "linear" in the Girard sense [Ben94, Bar97]. However, they seem quite natural in the bunches setting. O'Hearn and Pym, for instance, have examples of **BI** terms — the next chapter will show that the λΛ-calculus is in propositions-as-types correspondence with a non-trivial fragment of **BI** — where linear variables appear more than once or not at all [OP99].

Example 3 *The linear variable x occurs in the terms $cx{:}Bx$ (assuming $c : \Lambda x{:}A.Bx$), $fx{:}d$ (assuming $f{:}a \multimap d$) and $\lambda y{:}Cx.y : Cx \multimap Cx$ (assuming $C{:}A \multimap \mathsf{Type}$).*

In the sequel we will often refer informally to the concept of a linearity constraint. Essentially this means that all linear variables declared in the context are used: a production-consumption contract. But we depart from the usual resource-conscious logics idea that formulae are produced in the antecedent and consumed in the succedent. Given this, the judgement $x{:}A, y{:}cx \vdash_\Sigma y{:}cx$ in which the linear x is consumed by the (type of) y declared after it and the y itself is consumed in the succedent, is a valid one.

In the λΛ-calculus, signatures are used to keep track of the types and kinds assigned to constants. Contexts are used to keep track of the types, both linear and intuitionistic, assigned to variables. The abstract syntax for signatures and contexts is given by the following grammar:

$$\begin{array}{lll} \textit{Signatures} & \Sigma ::= & \langle\rangle \mid \Sigma, a!K \mid \Sigma, c!A \\ \textit{Contexts} & \Gamma ::= & \langle\rangle \mid \Gamma, x{:}A \mid \Gamma, x!A \end{array}$$

So signatures and contexts consist of finite sequences of declarations. The dependency aspect of the type theory requires that "[bases] have to become linearly ordered" [Bar92, page 198]. We assume the usual extraction functions $(dom(\Gamma), ran(\Gamma))$ related to such lists. We also define the following two functions which extract out just the *lin*ear and intuitionistic or *exp*onential parts of a context:

$$\begin{array}{rclrcl} lin(\langle\rangle) & = & \langle\rangle & exp(\langle\rangle) & = & \langle\rangle \\ lin(\Gamma, x{:}A) & = & lin(\Gamma), x{:}A & exp(\Gamma, x{:}A) & = & exp(\Gamma) \\ lin(\Gamma, x!A) & = & lin(\Gamma) & exp(\Gamma, x!A) & = & exp(\Gamma), x!A \end{array}$$

The λΛ-calculus is a formal system for deriving the following judgements:

$\vdash \Sigma$ sig	Σ is a valid signature
$\vdash_\Sigma \Gamma$ context	Γ is a valid context in Σ
$\Gamma \vdash_\Sigma K$ Kind	K is a valid kind in Σ and Γ
$\Gamma \vdash_\Sigma A{:}K$	A has a kind K in Σ and Γ
$\Gamma \vdash_\Sigma M{:}A$	M has a type A in Σ and Γ

We write $\Gamma \vdash_\Sigma U{:}V$ for either of $\Gamma \vdash_\Sigma A{:}K$ or $\Gamma \vdash_\Sigma M{:}A$, and $\Gamma \vdash_\Sigma X$ for $\Gamma \vdash_\Sigma K$ Kind or $\Gamma \vdash_\Sigma U{:}V$. We abuse notation and also write $\Gamma \vdash_\Sigma X$ to indicate the derivability of X in the λΛ-calculus, in which case the K or U is said to be *valid* in the signature Σ and context Γ.

The definition of the type theory depends crucially on the following three notions:

1. The joining together of two contexts to form a third must be undertaken so that the order of declarations and type of variables (linear versus intuitionistic) is respected;

2. The idea of linear variable occurrences allows us to form contexts of the form $x{:}A, x{:}A$, for some type constant A in the signature. That is, contexts in which repeated but distinct declarations of the same variable are possible;

3. Following a joining of contexts, certain occurrences of linear variables – those that are shared by a function and its argument – are identified with one another. This sharing is implemented by the κ function.

These notions will be explicated at appropriate points in the sequel.

We now present the rules for deriving judgements in Tables 2.1 and 2.2 below. To save space, we place any side-conditions along with the premisses. The rules are conveniently separated into a linear and an intuitionistic set, the latter relating directly to the intuitionistic λΠ-calculus.

The signature formation rules enforce intuitionistic behaviour by allowing only a constant of intuitionistic type to extend the signature. The context formation rules allow only types to be assigned to variables. We distinguish between extending the context by linear $(\Xi, x{:}A)$ and intuitionistic $(\Xi, x!A)$ variables. The context formation rules introduce two particular characteristics of the type theory. The first one is that of joining the premiss contexts for the multiplicative rules. The join must respect the ordering of the premiss contexts and the concept of linear versus intuitionistic variables. A method to join Γ and Δ into Ξ – denoted by $[\Xi; \Gamma; \Delta]$ – is defined in Section 2.3.1 below.

Valid Signatures

$$\frac{}{\vdash () \text{ sig}}\,(\Sigma)$$

$$\frac{\vdash \Sigma \text{ sig} \quad \vdash_\Sigma K \text{ Kind} \quad a \notin \Sigma}{\vdash \Sigma, a!K \text{ sig}}\,(\Sigma K!) \qquad \frac{\vdash \Sigma \text{ sig} \quad \vdash_\Sigma A\text{:Type} \quad c \notin \Sigma}{\vdash \Sigma, c!A \text{ sig}}\,(\Sigma A!)$$

Valid Contexts

$$\frac{\vdash \Sigma \text{ sig}}{\vdash_\Sigma () \text{ context}}\,(\Gamma)$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad \Delta \vdash_\Sigma A\text{:Type} \quad [\Xi;\Gamma;\Delta] \quad (x \notin dom(\Xi) \text{ or } x{:}A \in \Xi)}{\vdash_\Sigma \Xi, x{:}A \text{ context}}\,(\Gamma A)$$

$$\frac{\vdash_\Sigma \Gamma \text{ context} \quad \Delta \vdash_\Sigma A\text{:Type} \quad [\Xi;\Gamma;\Delta] \quad (x \notin dom(\Xi) \text{ or } x{:}A \in \Xi)}{\vdash_\Sigma \Xi, x!A \text{ context}}\,(\Gamma A!)$$

Valid Kinds

$$\frac{\vdash_\Sigma \Gamma \text{ context}}{\Gamma \vdash_\Sigma \text{ Type Kind}}\,(KAx) \qquad \frac{\Gamma, x{:}A \vdash_\Sigma K \text{ Kind}}{\Gamma \vdash_\Sigma \Lambda x{:}A.K \text{ Kind}}\,(K\Lambda I1)$$

$$\frac{\Gamma \vdash_\Sigma A\text{:Type} \quad \Delta \vdash_\Sigma K\text{:Kind} \quad [\Xi';\Gamma;\Delta] \quad \Xi = \Xi' \backslash (lin(\Gamma) \cap lin(\Delta))}{\Xi \vdash_\Sigma A \multimap K\text{:Kind}}\,(K\Lambda I2)$$

$$\frac{\Gamma, x!A \vdash_\Sigma K \text{ Kind}}{\Gamma \vdash_\Sigma \Lambda x!A.K \text{ Kind}}\,(K\Lambda!I)$$

Table 2.1: λΛ-calculus

In order to motivate the second characteristic of the type theory, consider the following simple, apparently innocuous, derivation.

Example 4 *Let* $A!$Type, $c!A \multimap$ Type $\in \Sigma$ *and note that the argument type, cx, is a dependent one; the linear x is free in it.*

$$\frac{\dfrac{\dfrac{\begin{array}{c}\vdots\\ x{:}A \vdash_\Sigma cx\text{:Type}\end{array}}{x{:}A, z{:}cx \vdash_\Sigma z{:}cx}}{x{:}A \vdash_\Sigma \lambda z{:}cx.z : \Lambda z{:}cx.cx} \quad \dfrac{\begin{array}{c}\vdots\\ x{:}A \vdash_\Sigma cx\text{:Type}\end{array}}{x{:}A, y{:}cx \vdash_\Sigma y{:}cx}}{x{:}A, x{:}A, y{:}cx \vdash_\Sigma (\lambda z{:}cx.z)y : cx}$$

The problem is that an excess of linear xs now appear in the combined context after the application step. (In this step, the types match literally. However this problem arises where they are equal too.) Our solution is to recognize the two xs as two *distinct* occurrences of the *same* variable, the one occurring in the argument type cx, and to allow a degree of freedom in sharing these occurrences. It is now necessary to formally define a binding strategy for multiple occurrences;

Valid Families of Types

$$\frac{\vdash_\Sigma !\Gamma \text{ context} \quad a!K \in \Sigma}{!\Gamma \vdash_\Sigma a{:}K}\,(Ac)$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma B{:}\mathsf{Type}}{\Gamma \vdash_\Sigma \Lambda x{:}A.B : \mathsf{Type}}\,(A\Lambda I1) \qquad \frac{\Gamma \vdash_\Sigma A{:}\mathsf{Type} \quad \Delta \vdash_\Sigma B{:}\mathsf{Type} \quad [\Xi';\Gamma;\Delta] \quad \Xi = \Xi' \backslash (lin(\Gamma) \cap lin(\Delta))}{\Xi \vdash_\Sigma A \multimap B{:}\mathsf{Type}}\,(A\Lambda I2)$$

$$\frac{\Gamma, x!A \vdash_\Sigma B{:}\mathsf{Type}}{\Gamma \vdash_\Sigma \Lambda x!A.B : \mathsf{Type}}\,(A\Lambda!I)$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma B{:}K}{\Gamma \vdash_\Sigma \lambda x{:}A.B : \Lambda x{:}A.K}\,(A\lambda\Lambda I) \qquad \frac{\Gamma \vdash_\Sigma B : \Lambda x{:}A.K \quad \Delta \vdash_\Sigma N{:}A \quad [\Xi';\Gamma;\Delta] \quad \Xi = \Xi' \backslash \kappa(\Gamma,\Delta)}{\Xi \vdash_\Sigma BN : K[N/x]}\,(A\Lambda\mathcal{E})$$

$$\frac{\Gamma, x!A \vdash_\Sigma B{:}K}{\Gamma \vdash_\Sigma \lambda x!A.B : \Lambda x!A.K}\,(A\lambda\Lambda!I) \qquad \frac{\Gamma \vdash_\Sigma B : \Lambda x!A.K \quad !\Delta \vdash_\Sigma N{:}A \quad [\Xi;\Gamma;!\Delta]}{\Xi \vdash_\Sigma BN : K[N/x]}\,(A\Lambda!\mathcal{E})$$

$$\frac{\Gamma \vdash_\Sigma A{:}\mathsf{Type} \quad \Gamma \vdash_\Sigma B{:}\mathsf{Type}}{\Gamma \vdash_\Sigma A\&B{:}\mathsf{Type}}\,(A\&I)$$

$$\frac{\Gamma \vdash_\Sigma A{:}K \quad \Delta \vdash_\Sigma K' \text{ Kind} \quad K \equiv K' \quad [\Xi;\Gamma;\Delta]}{\Xi \vdash_\Sigma A{:}K'}\,(A\equiv)$$

Valid Objects

$$\frac{\vdash_\Sigma !\Gamma \text{ context} \quad c!A \in \Sigma}{!\Gamma \vdash_\Sigma c{:}A}\,(Mc)$$

$$\frac{\Gamma \vdash_\Sigma A{:}\mathsf{Type}}{\Gamma, x{:}A \vdash_\Sigma x{:}A}\,(MVar) \qquad \frac{\Gamma \vdash_\Sigma A{:}\mathsf{Type}}{\Gamma, x!A \vdash_\Sigma x{:}A}\,(MVar!)$$

$$\frac{\Gamma, x{:}A \vdash_\Sigma M{:}B}{\Gamma \vdash_\Sigma \lambda x{:}A.M : \Lambda x{:}A.B}\,(M\lambda\Lambda I) \qquad \frac{\Gamma \vdash_\Sigma M : \Lambda x{:}A.B \quad \Delta \vdash_\Sigma N{:}A \quad [\Xi';\Gamma;\Delta] \quad \Xi = \Xi' \backslash \kappa(\Gamma,\Delta)}{\Xi \vdash_\Sigma MN : B[N/x]}\,(M\Lambda\mathcal{E})$$

$$\frac{\Gamma, x!A \vdash_\Sigma M{:}B}{\Gamma \vdash_\Sigma \lambda x!A.M : \Lambda x!A.B}\,(M\lambda\Lambda!I) \qquad \frac{\Gamma \vdash_\Sigma M : \Lambda x!A.B \quad !\Delta \vdash_\Sigma N{:}A \quad [\Xi;\Gamma;!\Delta]}{\Xi \vdash_\Sigma MN : B[N/x]}\,(M\Lambda!\mathcal{E})$$

$$\frac{\Gamma \vdash_\Sigma M{:}A \quad \Gamma \vdash_\Sigma N{:}B}{\Gamma \vdash_\Sigma \langle M,N \rangle : A\&B}\,(M\&I) \qquad \frac{\Gamma \vdash_\Sigma M : A_0\&A_1}{\Gamma \vdash_\Sigma \pi_i M : A_i}\,(M\&\mathcal{E}_i)\ (i \in \{0,1\})$$

$$\frac{\Gamma \vdash_\Sigma M{:}A \quad \Delta \vdash_\Sigma A'{:}\mathsf{Type} \quad A \equiv A' \quad [\Xi;\Gamma;\Delta]}{\Xi \vdash_\Sigma M{:}A'}\,(M\equiv)$$

Table 2.2: λΛ-calculus (continued)

this we do in Section 2.3.2 below. The sharing aspect is implemented via the κ function, defined in Section 2.3.3 below. One implication of this solution is that repeated declarations of the same variable are allowed in contexts. For this reason, the usual side-condition of $x \notin dom(\Xi)$ is absent from the rules for valid contexts, though of course we don't allow the same variable to inhabit two distinct types.

The $(K\Lambda I)$ and $(A\Lambda I)$ pair of rules form linear function spaces. The first of each pair, in which $x \in FV(B)$, constructs linear dependent function spaces. The second rule of each pair constructs the ordinary linear function spaces. There are two side conditions for the latter rules: the first joins the premiss contexts and the second then does a necessary book-keeping for those occurrences of linear variables which are identified with each other under the current binding strategy. The side-conditions in the $(A\Lambda\mathcal{E})$ and $(M\Lambda\mathcal{E})$ rules are of a similar nature. The κ function selects those such "critical" linear occurrences. These occurrences are removed to give the conclusion context. It can be seen that these side-conditions are type-theoretically and, via the propositions-as-types correspondence, logically natural.

The essential difference between linear and intuitionistic function spaces can be observed by considering the $(M\Lambda\mathcal{E})$ and $(M\Lambda!\mathcal{E})$ rules. For the latter, the context for the argument $N{:}A$ is an entirely intuitionistic one $(!\Delta)$, which allows the function to use N as many times as it likes.

Example 5 *We end this sub-section with an example of a derivation which does not involve sharing. Let* $A!\mathsf{Type}$, $d!A \multimap \mathsf{Type}$, $e!\Lambda y{:}A.dy \in \Sigma$. *Then we construct*

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash_\Sigma () \text{ context}}{\vdash_\Sigma e : \Lambda y{:}A.dy} \quad \dfrac{\vdash_\Sigma A{:}\mathsf{Type}}{x{:}A \vdash_\Sigma x{:}A}}{x{:}A \vdash_\Sigma ex : dx}}{\vdash_\Sigma \lambda x{:}A.ex : \Lambda x{:}A.dx} \quad \dfrac{\vdash_\Sigma A{:}\mathsf{Type}}{z{:}A \vdash_\Sigma z{:}A}}{z{:}A \vdash_\Sigma (\lambda x{:}A.ex)z : (dx)[z/x]}$$

Now, $(\lambda x{:}A.ex)z \to_\beta ez$ *and* $ez{:}dz$*, which maintains the linear occurrence of the variable* z*.*

2.3.1 Context joining

The method of joining two contexts is a ternary relation $[\Xi;\Gamma;\Delta]$, to be read as "the contexts Γ and Δ are joined to form the context Ξ". Or, for proof-search: "the context Ξ is split into the contexts Γ and Δ".

The first rule for defining $[\Xi;\Gamma;\Delta]$ states that an empty context can be formed by joining together two empty contexts. The second and third rules comply with the linearity constraint, and imply that the linear variables in Ξ are exactly those of Γ and Δ. The last rule takes account of the structural properties of intuitionistic variables. In search, the intuitionistic variable $x!A$ would be sent both ways when the context is split.

$$\frac{}{[\langle\rangle;\langle\rangle;\langle\rangle]}\ \text{(JOIN)}$$

$$\frac{[\Xi;\Gamma;\Delta]}{[\Xi,x{:}A;\Gamma,x{:}A;\Delta]}\ \text{(JOIN-L)} \qquad \frac{[\Xi;\Gamma;\Delta]}{[\Xi,x{:}A;\Gamma;\Delta,x{:}A]}\ \text{(JOIN-R)}$$

$$\frac{[\Xi;\Gamma;\Delta]}{[\Xi,x!A;\Gamma,x!A;\Delta,x!A]}\ \text{(JOIN-!)}$$

Table 2.3: Context joining

Further, the context joining relation must respect the ordering of the contexts and the linearity constraint (as defined by the binding strategy in the next section). That is, if $\vdash_\Sigma \Gamma$ context, $\vdash_\Sigma \Delta$ context and $[\Xi;\Gamma;\Delta]$, then $\vdash_\Sigma \Xi$ context (and *vice versa* for when Ξ is split into Γ and Δ). We remark that if we were also studying the distribution laws for relevant contexts, then the context joining relation would need to take regard of these context equalities.

We make a brief remark about the $[\Xi;\Gamma;\Delta]$ relation with regard to logic programming. As we noted above, in proof-search (the basis of logic programming) the relation $[\Xi;\Gamma;\Delta]$ is read as "split Ξ into Γ and Δ". An implementation of the λΛ-calculus as a logic programming language would have to calculate such splittings, perhaps using techniques similar to those for Lolli and Lygon [HPW96, HM94], although it would be interesting to consider approaches in which $[\Xi;\Gamma;\Delta]$ remained unevaluated for as long as possible during search. Such an approach would resemble matrix methods [Wal90].

2.3.2 Multiple occurrences

Consider the multiple occurrences idea from a proposition-as-types reading. Then $x{:}A, x{:}A$ can be understood as two uses of the same proof of the same proposition, as opposed to $x{:}A, y{:}A$, which can be seen as distinct proofs of the same proposition. Though this idea can be seen, in the presence of the binding strategy that we are about to define, as an internalization of α-conversion, it allows us a degree of freedom, that at the structural level of terms (as opposed to types), which is useful in dealing with variable sharing (Section 2.3.3).

In this section, we define the "left-most free occurrence of x" in U and a corresponding binding strategy for it. We use this in the sequel, later noting that it can be generalized.

Definition 6 *The left-most linear occurrence of x in U is defined as follows, provided that $x \in LFV(U)$. We use @ to denote atoms (constants and variables) and say "x, @ distinct" if @ is a, c or y.*

(Constant, Variable) The constant and variable cases are trivial:

$$\begin{array}{lcll} lm_x(@) & = & \{\} & x, @ \text{ distinct} \\ lm_x(x) & = & \{x\} & \end{array}$$

(Abstraction) We adopt the usual technique of capture-avoiding substitution for the case where another occurrence of x has already been bound. By induction, the λ (Λ) binds a given occurrence — the left-most one — of x in U. So we can α-convert this to $\lambda z{\in}A.V[z/x]$ ($\Lambda z{\in}A.V[z/x]$) and continue. We give the cases for the λ binder; the ones for Λ are exactly similar.

$$\begin{array}{lcll} lm_x(\lambda y{\in}A.V) & = & \left\{ \begin{array}{ll} lm_x(A) & x \in LFV(A) \\ lm_x(V) & \text{otherwise} \end{array} \right\} & x, y \text{ distinct} \\ lm_x(\lambda x{\in}A.V) & = & lm_x(\lambda z{\in}A.V[z/x]) & z \text{ new} \end{array}$$

(Application) The left-most occurrence of x in VM is in V or, failing that, in M. The case where V is a constant or variable is straight-forward:

$$\begin{array}{lcll} lm_x(@M) & = & lm_x(M) & x, @ \text{ distinct} \\ lm_x(xM) & = & \{x\} & \end{array}$$

Otherwise, we need to check whether x is free in V or not:

$$lm_x(VM) = \left\{ \begin{array}{ll} lm_x(V) & x \in LFV(V) \\ lm_x(M) & \text{otherwise} \end{array} \right.$$

(Pairing) We deal with the additive cases by a disjoint union of the left-most occurrence of x in both components of the pair:

$$\begin{array}{lcll} lm_x(\langle M,N\rangle) & = & lm_x(M) \uplus lm_x(N) & \\ lm_x(\pi_i(M)) & = & lm_x(M) & i \in \{0,1\} \\ lm_x(A \& B) & = & lm_x(A) \uplus lm_x(B) & \end{array}$$

We define the left-most occurrence of $x{:}A$ in a context Γ as the first declaration of $x{:}A$ in Γ. Similarly, the right-most occurrence of $x{:}A$ in Γ is the last such declaration.

The binding strategy now formalizes the concept of linearity constraint:

Definition 7 (left-most binding) *Assume $\Gamma, x{:}A, \Delta \vdash_\Sigma U{:}V$ and that $x{:}A$ is the right-most occurrence of x in the context. Then x binds:*

1. *The first left-most occurrence of x in $ran(\Delta)$, if there is such a declaration;*
2. *The unbound left-most linear occurrences of x in $U{:}V$.*

There is no linearity constraint for intuitionistic variables: the right-most occurrence of $x!A$ in the context binds all the unbound xs used in the type of a declaration in Δ and all the occurrences of x in $U{:}V$.

The rules for deriving judgements are now read according to the strategy in place. For example, in the $(M\lambda\Lambda I)$ rule, the $\lambda(\Lambda)$ binds the left-most occurrence of x in $M(B)$. Similarly, in the (admissible) cut rule, the term $N{:}A$ cuts with the left-most occurrence of $x{:}A$ in the context $\Delta, x{:}A, \Delta'$. In the corresponding intuitionistic rules, the $\lambda!(\Lambda!)$ binds all occurrences of x in $M(B)$ and $N{:}A$ cuts all occurrences of $x!A$ in the context $\Delta, x!A, \Delta'$.

In the sequel we use the left-most binding and cutting strategy as discussed above. We remark that there is a general ij strategy, that of binding the i^{th} variable from the left and cutting the j^{th} variable from the left.

2.3.3 Variable sharing

Variable sharing is a central notion which allows linear dependency to be set up. In fact, this notion is already implicit in our definition (1) of linear occurrence. The $\lambda\Lambda$-calculus uses a function κ which implements the degree of sharing of variables between functions and their arguments.

We define κ by considering the situation when either of the two contexts Γ or Δ are of the form $\ldots, x{:}A$ or $\ldots, x{:}A, y{:}Bx$. The only case when the two declarations of $x{:}A$ are not identified with each other is when both Γ and Δ are of the form $\ldots, x{:}A, y{:}Bx$.

Definition 8 *The function κ is defined for the binary, multiplicative $(A\Lambda\mathcal{E})$, $(M\Lambda\mathcal{E})$ and (Cut) rules*

$$\frac{\Gamma \vdash_\Sigma U : \Lambda z{:}C.V \quad \Delta \vdash_\Sigma N{:}C \quad [\Xi';\Gamma;\Delta] \quad \Xi = \Xi' \backslash \kappa(\Gamma,\Delta)}{\Xi \vdash_\Sigma UN : V[N/z]} (A\Lambda\mathcal{E}), (M\Lambda\mathcal{E})$$

$$\frac{\begin{array}{l}\Pi(\Delta[N/x]) \\ \quad\vdots \\ \Delta, z{:}C, \Delta' \vdash_\Sigma U{:}V \quad \Gamma \vdash_\Sigma N{:}C \quad [\Xi';\Gamma;\Delta,\Delta'[N/x]] \quad \Xi = \Xi' \backslash \kappa(\Gamma,(\Delta,\Delta'[N/x]))\end{array}}{\Xi \vdash_\Sigma (U{:}V)[N/z]} \text{(Cut)}.$$

For each $x{:}A$ occurring in both Γ and Δ, construct from right to left as follows:[4]

$$\kappa(\Gamma,\Delta) = \{\} \qquad \textit{if } lin(\Gamma)\cap lin(\Delta)=\emptyset$$

$$\kappa(\Gamma,\Delta) = \left\{ \begin{array}{r} \{x{:}A \in lin(\Gamma)\cap lin(\Delta) \mid \textit{either } (i) \textit{ there is no } y{:}B(x) \textit{ to} \\ \textit{the right of } x{:}A \textit{ in } \Gamma \\ \textit{or } (ii) \textit{ there is no } y{:}B(x) \textit{ to} \\ \textit{the right of } x{:}A \textit{ in } \Delta \\ \textit{or both } (i) \textit{ and } (ii)\} \end{array} \right\} \quad \textit{otherwise}$$

The second clause of the definition can also be stated as follows: in at least one of Γ and Δ there is no $y{:}B(x)$ to the right of the occurrence of $x{:}A$. This clause is needed to form a consistent type theory which allows the formation of sufficiently complex dependent types. By this, we mean types such as $\Lambda x_1{:}A_1 \ldots \Lambda x_n{:}A_n(x_1,\ldots,x_{n-1}).A$ in which the abstracting types depend upon previously abstracted variables. In binary rules, it can be that some variables must occur, in order to establish the well-formedness of types in each premiss, in the contexts of both premisses, and must occur only once in order to establish the well-formedness of types in the conclusion. However, it is possible for other variables occurring in both premisses to play a role in the logical structure of the proof; these variables must be duplicated in the conclusion. These requirements are regulated by κ.

In the absence of sharing of variables, when the first clause only applies, we still obtain a useful linear dependent type theory, with a linear dependent function space but without the dependency of the abstracting A_is on the previously abstracted variables. For example, we use such a type theory to encode the dynamic semantics of ML with references in Section 2.5 later.

With the definition of κ given above, we can consider the following example.

Example 9 *Suppose $A!\mathsf{Type}, c!A \multimap \mathsf{Type} \in \Sigma$. Then we construct the following:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\vdash_\Sigma c{:}A \multimap \mathsf{Type} \quad \dfrac{\vdash_\Sigma A{:}\mathsf{Type}}{x{:}A \vdash_\Sigma x{:}A}\ (*)}{x{:}A \vdash_\Sigma cx{:}\mathsf{Type}}}{x{:}A, z{:}cx \vdash_\Sigma z{:}cx}}{x{:}A \vdash_\Sigma \lambda z{:}cx.z : \Lambda z{:}cx.cx} \qquad \dfrac{\dfrac{\vdash_\Sigma c{:}A \multimap \mathsf{Type} \quad \dfrac{\vdash_\Sigma A{:}\mathsf{Type}}{x{:}A \vdash_\Sigma x{:}A}\ (*)}{x{:}A \vdash_\Sigma cx{:}\mathsf{Type}}}{x{:}A, y{:}cx \vdash_\Sigma y{:}cx} \quad (**)}{x{:}A, y{:}cx \vdash_\Sigma (\lambda z{:}cx.z)y : cx}$$

[4]Formally, $\kappa(\Gamma,\Delta)$ is defined recursively on the structure of Γ and Δ, read from right to left. We adopt the above informal notation for ease of expression.

The $(*)$ *denotes the context join to get* $x{:}A$. *The* $(**)$ *side-condition is more interesting. First, the premiss contexts are joined together to get* $x{:}A, x{:}A, y{:}cx$. *Then,* κ *removes the extra occurrence of* $x{:}A$ *and so restores the linearity constraint. A similar situation arises when the* y *is cut in for the* z*:*

$$\frac{x{:}A, z{:}cx \vdash_\Sigma z{:}cx \quad x{:}A, y{:}cx \vdash_\Sigma y{:}cx \quad (**')}{x{:}A, y{:}cx \vdash_\Sigma (z{:}cx)[y/z]}$$

The function κ is not required, *i.e.*, its use is vacuous, when certain restrictions of the λΛ-calculus type theory are considered. For instance, if we restrict type-formation to be entirely intuitionistic so that type judgements are of the form $!\Gamma \vdash_\Sigma A{:}\mathrm{Type}$, then we recover the $\{\Pi, \multimap, \&\}$-fragment of Cervesato and Pfenning's $\lambda^{\Pi\multimap\&\top}$ type theory [CP96]. Our fragment does not include $\top$, the unit of &; we will remark on this while stating the subject reduction property in Section 2.3.5 later. Like the simple dependency case above, this restricted type theory is useful too; we use it to encode a fragment of intuitionistic linear logic in Section 2.5 later.

2.3.4 Definitional equality

The definitional equality relation that we consider here is the β-conversion of terms at all three levels. The definitional equality relation, $\equiv$, between terms at each respective level is defined to be the symmetric and transitive closure of the parallel nested reduction relation, $\rightarrow$, defined in Table 2.4 below. We note that, in the β-rules, substitution is performed only for the bound occurrences of x. The transitive closure of $\rightarrow$ is denoted by $\rightarrow^*$.

We remark that while β-conversion is sufficient for our current purposes, we foresee little difficulty (other than that for the λΠ-calculus [Coq91, Sal90]) in strengthening the definitional equality relationship by the η-rule. We remark on this again in Section 2.3.6 later.

2.3.5 Basic properties of the λΛ-calculus

In this section, we study the basic properties of the λΛ-calculus. The proof techniques are not new and are adapted from Harper *et al.* [HHP93] for this setting. There are some minor points to note regarding the proofs of the meta-theory. The first point to note is this. As the λΛ-calculus is a conservative extension of the λΠ-calculus, the number of cases we have to deal with in the proofs of structural induction is much greater. We concentrate on the $\Gamma \vdash_\Sigma M{:}A$-fragment of the

$$\frac{}{U \to U}\,(\to refl) \qquad \frac{A \to A' \quad M \to M'}{\lambda x{\in}A.M \to \lambda x{\in}A'.M'}\,(\to M\lambda)$$

$$\frac{A \to A' \quad K \to K'}{\Lambda x{\in}A.K \to \Lambda x{\in}A'.K'}\,(\to K\Lambda) \qquad \frac{M \to M' \quad N \to N'}{MN \to M'N'}\,(\to Mapp)$$

$$\frac{A \to A' \quad B \to B'}{\Lambda x{\in}A.B \to \Lambda x{\in}A'.B'}\,(\to A\Lambda) \qquad \frac{M \to M' \quad N \to N'}{(\lambda x{\in}A.M)N \to M'[N'/x]}\,(\to M\beta)$$

$$\frac{A \to A' \quad B \to B'}{\lambda x{\in}A.B \to \lambda x{\in}A'.B'}\,(\to A\lambda) \qquad \frac{M \to M' \quad N \to N'}{\langle M,N\rangle \to \langle M',N'\rangle}\,(\to M\&)$$

$$\frac{A \to A' \quad M \to M'}{AM \to A'M'}\,(\to Aapp) \qquad \frac{M \to M'}{\pi_i M \to \pi_i M'}\,(\to M\pi)$$

$$\frac{B \to B' \quad N \to N'}{(\lambda x{\in}A.B)N \to B'[N'/x]}\,(\to A\beta) \qquad \frac{M \to M'}{\pi_0\langle M,N\rangle \to M'}\,(\to M\pi_0)$$

$$\frac{A \to A' \quad B \to B'}{A\&B \to A'\&B'}\,(\to A\&) \qquad \frac{N \to N'}{\pi_1\langle M,N\rangle \to N'}\,(\to M\pi_1)$$

Table 2.4: Parallel nested reduction

type theory (the $\Gamma \vdash_\Sigma A{:}K$-fragment is dealt with quite similarly). Even then, we show only the representative or interesting cases.

Another reason for the large number of cases is the linear dependent aspect of the type theory. For instance, consider the judgement $\Xi, x{:}A, y{:}B, \Xi' \vdash_\Sigma U{:}V$ which is obtained via the application of some binary rule. The linearity constraint allows the following: x can linearly occur in one of the types declared in $y{:}B, \Xi'$ and can linearly occur in $U{:}V$, and y can linearly occur in one of the types declared in Ξ' and can linearly occur in $U{:}V$. If no sharing has occurred, then it can be assumed that x and y are sent to separate premiss contexts. Sharing complicates the analysis we have to do, though often the argument is still a fairly routine inductive one.

We consider some of the characteristic properties of the reduction relation $\to$. All three of the following are desirable technical properties, as well as of use in proving the more major results later. We first show that substitution is conserved under reduction.

Lemma 10 *If $U \to U'$ and $V \to V'$ then $U[V/x] \to U'[V'/x]$.*

Proof : By induction on the length of the proof of $U \to U'$. We do some representative cases.

U is M Then take $U' = U$ and $U[V/x] \to U'[V'/x]$ is either $U \to U'$ or $V \to V'$.

U is $\lambda y{\in}A\,.M$ $U \to U'$ is inferred by $(\to M\lambda)$. Then U' is of the form $\lambda y{\in}A'\,.M'$ and $\lambda y{\in}A\,.M \to \lambda y{\in}A'\,.M'$. By induction hypothesis twice we get

$$A[V/x] \to A'[V'/x]$$

and

$$M[V/x] \to M'[V'/x]$$

The rule $(\to M\lambda)$ allows us to infer $\lambda y{\in}A[V/x]\,.M[V/x] \to \lambda y{\in}A'[V'/x]\,.M'[V'/x]$, which is the same as $(\lambda y{\in}A\,.M)[V/x] \to (\lambda y{\in}A'\,.M')[V'/x]$.

U is MN There are two sub-cases to consider.

1. $U \to U'$ is inferred by $(\to Mapp)$. Then U' is of the form $M'N'$ and $MN \to M'N'$. By induction hypothesis twice we get

$$M[V/x] \to M'[V'/x]$$

and

$$N[V/x] \to N'[V'/x]$$

The rule $(\to Mapp)$ allows us to infer $(M[V/x]N[V/x]) \to (M'[V'/x]N'[V'/x])$, which is the same as $(MN)[V/x] \to (M'N')[V'/x]$.

2. $U \to U'$ is inferred by $(\to M\beta)$. Then U' is of the form $M'[N'/y]$ and $(\lambda y{\in}A\,.M)N \to M'[N'/y]$. By induction hypothesis twice we get that

$$M[V/x] \to M'[V'/x]$$

and

$$N[V/x] \to N'[V'/x]$$

The rule $(\to M\beta)$ allows us to infer

$$(\lambda y{\in}A\,.M[V/x])(N[V/x]) \to (M[V'/x])[N'[V'/x]/y]$$

which is the same as $((\lambda y{\in}A\,.M)N)[V/x] \to (M'[N'/y])[V'/x]$. □

The proof of confluency for the parallel reduction relation $\twoheadrightarrow$ is an adaption of the combinatorial proof by Stenlund [Ste72], who traces it back to Tait and Martin-Löf [ML75]. The combinatorial approach works as the reduction relation (β for Kinds, Types and Objects) is defined disjointly for each of the three levels of terms.

Lemma 11 ($\twoheadrightarrow \models \diamond$) *If $U \twoheadrightarrow U'$ and $U \twoheadrightarrow U''$ then there exists a V such that $U' \twoheadrightarrow V$ and $U'' \twoheadrightarrow V$.*

Proof : By induction on the sum of the lengths of the proofs of $U \twoheadrightarrow U'$ and $U \twoheadrightarrow U''$.

U **is** M Then take $U' = U'' = M$.

U **is** $\lambda x{\in}A\,.M$ Then we have

$$\begin{tikzcd}
 & \lambda x{\in}A\,.M \arrow[dl] \arrow[dr] & \\
\lambda x{\in}A'\,.M'' \arrow[dr, dotted] & & \lambda x{\in}A''\,.M'' \arrow[dl, dotted] \\
 & \lambda x{\in}A'''\,.M''' &
\end{tikzcd}$$

with $A \twoheadrightarrow A', A''$ and $M \twoheadrightarrow M', M''$. By induction hypothesis twice we have that there exists a A''' such that $A' \twoheadrightarrow A''' \twoheadleftarrow A''$ and there exists a M''' such that $M' \twoheadrightarrow M''' \twoheadleftarrow M''$. The result follows by ($\twoheadrightarrow M\lambda$) if we take V to be $A'''M'''$.

U **is** MN There are three cases to consider, depending on how $U \twoheadrightarrow U'$ and $U \twoheadrightarrow U''$ have been derived by ($\twoheadrightarrow Mapp$) and ($\twoheadrightarrow M\beta$).

1. Both $U \twoheadrightarrow U'$ and $U \twoheadrightarrow U''$ are derived by ($\twoheadrightarrow Mapp$). Then we have

$$\begin{tikzcd}
 & MN \arrow[dl] \arrow[dr] & \\
M'N' \arrow[dr, dotted] & & M''N'' \arrow[dl, dotted] \\
 & M'''N''' &
\end{tikzcd}$$

with $M \twoheadrightarrow M', M''$ and $N \twoheadrightarrow N', N''$. By induction hypothesis twice we have that there exists a M''' such that $M' \twoheadrightarrow M''' \twoheadleftarrow M''$ and there exists a N''' such that $N' \twoheadrightarrow N''' \twoheadleftarrow N''$. The result follows by ($\twoheadrightarrow Mapp$) if we take V to be $M'''N'''$.

2. Both $U \to U'$ and $U \to U''$ are derived by $(\to M\beta)$. Then we have

$$\begin{tikzcd}
 & (\lambda x{\in}A\,.M)N \arrow[dl] \arrow[dr] & \\
M'[N'/x] \arrow[dr, dotted] & & M''[N''/x] \arrow[dl, dotted] \\
 & M'''[N'''/x] &
\end{tikzcd}$$

with $M \to M', M''$ and $N \to N', N''$. By induction hypothesis twice we have that there exists a M''' such that $M' \to M''' \leftarrow M''$ and there exists a N''' such that $N' \to N''' \leftarrow N''$. By Lemma 10 we have that $M'[N'/x] \to M'''[N'''/x] \leftarrow M''[N''/x]$ and the result follows for V as in the diagram above.

3. $U \to U'$ follows by $(\to Mapp)$and $U \to U''$ by $(\to M\beta)$. Then we have

$$\begin{tikzcd}
 & (\lambda x{\in}A\,.M)N \arrow[dl] \arrow[dr] & \\
(\lambda x{\in}A'\,.M')N' \arrow[dr, dotted] & & M''[N''/x] \arrow[dl, dotted] \\
 & M'''[N'''/x] &
\end{tikzcd}$$

with $M \to M', M''$ and $N \to N', N''$. $\lambda x{\in}A\,.M \to \lambda x{\in}A'\,.M'$ is derived by an $(\to M\lambda)$. By induction hypothesis twice we have that there exists a M''' st $M' \to M''' \leftarrow M''$ and there exists a N''' such that $N' \to N''' \leftarrow N''$. By Lemma 10 we have that $M''[N''/x] \to M'''[N'''/x]$ and by $(\to M\beta)$ we have that $(\lambda x{\in}A'\,.M')N \to M'''[N'''/x]$, so the result follows as in the diagram.

U is $\langle M, N\rangle$ Then we have

$$\begin{tikzcd}
 & \langle M, N\rangle \arrow[dl] \arrow[dr] & \\
\langle M', N'\rangle \arrow[dr, dotted] & & \langle M'', N''\rangle \arrow[dl, dotted] \\
 & \langle M''', N'''\rangle &
\end{tikzcd}$$

with $M \to M', M''$ and $N \to N', N''$. By induction hypothesis twice we have that there exists a M''' such that $M' \to M''' \leftarrow M''$ and there exists a N''' such that $N' \to N''' \leftarrow N''$. The result follows by $(\to M\&)$ if we take V to be $\langle M''', N'''\rangle$.

U is $\pi_i M$ $i \in \{0,1\}$. Then we have

$$\begin{tikzcd}
 & \pi_i M \arrow[dl] \arrow[dr] & \\
\pi_i M' \arrow[dr, dotted] & & \pi_i M'' \arrow[dl, dotted] \\
 & \pi_i M''' &
\end{tikzcd}$$

with $M \rightarrow M', M''$ and $N \rightarrow N', N''$. By induction hypothesis we have that there exists a M''' st $M' \rightarrow M''' \leftarrow M''$. The result follows by $(\rightarrow M\pi)$ if we take V to be $\pi_i M'''$. □

The third property of $\rightarrow$, Church-Rosser, follows from confluency.

Lemma 12 (CR Property) *If $U \rightarrow^* U'_m$ and $U \rightarrow^* U''_n$, then there exists a V such that $U'_m \rightarrow^* V$ and $U''_n \rightarrow^* V$*

Proof : By an induction on the number of β-steps. For the base case we just use the Diamond property (confluence). For the inductive step, we assume the hypothesis for $(n+m)-1$ steps.

$$\begin{tikzcd}
U \arrow[r] \arrow[d] & U''_1 \arrow[r] \arrow[d, dotted] & U''_2 \arrow[r] \arrow[d, dotted] & \dots \arrow[r] & U''_n \arrow[d, dotted] \\
U'_1 \arrow[r, dotted] \arrow[d] & V_{1,1} \arrow[r, dotted] \arrow[d, dotted] & V_{2,1} \arrow[r, dotted] \arrow[d, dotted] & \dots \arrow[r, dotted] & V_{n,1} \arrow[d, dotted] \\
U'_2 \arrow[r, dotted] \arrow[d] & V_{1,2} \arrow[r, dotted] \arrow[d, dotted] & V_{2,2} \arrow[r, dotted] \arrow[d, dotted] & \dots \arrow[r, dotted] & V_{n,2} \arrow[d, dotted] \\
\vdots \arrow[d] & \vdots \arrow[d, dotted] & \vdots \arrow[d, dotted] & & \vdots \arrow[d, dotted] \\
U'_m \arrow[r, dotted] & V_{1,m} \arrow[r, dotted] & V_{2,m} \arrow[r, dotted] & \dots \arrow[r, dotted] & V_{n,m}
\end{tikzcd}$$

We complete the $n+m$ case by using confluency to fill-in the bottom-right rectangle. □

We consider the admissibility of the structural rules. The following lemma is useful for proving these admissibilities. It analyses how a type assignment for an abstraction can be obtained. It is a specific (part 4) and specialized (for linearity) case of Barendregt's Generation Lemma for Pure Type Systems (PTS) [Bar92].

Lemma 13 (Inversion) *If $\Gamma \vdash_\Sigma \lambda x{\in}A.U : \Lambda x{\in}A.V$, then $\Gamma, x{\in}A \vdash_\Sigma U{:}V$.*

Proof Consider a derivation of $\lambda x{\in}A\,.U : \Lambda x{\in}A\,.V$. The conversion rules do not change the term $\lambda x{\in}A\,.U$. We follow the branch of the derivation until the term $\lambda x{\in}A\,.U$ is introduced for the first time. This can be done by an abstraction rule. The conclusion of the abstraction rule is

$$\Gamma \vdash_\Sigma \lambda x{\in}A\,.U : \Lambda x{\in}B\,.W$$

with $\Lambda x{\in}A\,.V \equiv \Lambda x{\in}B\,.W$. The statement of the lemma follows by inspection of the abstraction.
□

We remark that, given the next theorem, the above lemma could allow weakening or contraction in the intuitionistic parts of Γ.

We recall our earlier comments, in Section 2.2, regarding how the consideration of a particular language allows us to admit certain structural rules. The next theorem details this for the λΛ-calculus. We comment on the form of the admissible structural rules. The exchange and contraction rules are inherited from dependent and linear type theory, respectively. The rule for weakening requires that the context for proving the well-typedness of A is entirely intuitionistic. The rule for dereliction requires the derelicting of the free variables in the linear type too. Cut comes in two forms, one for cutting a linear variable and one for cutting an exponential one. The rules are read according to the current binding strategy. The extra side-conditions for exchange enforce the context well-formedness in accordance with the left-most binding strategy.

Theorem 14 (Structural Admissibilities) *The following structural rules are admissible:*

1. Exchange*: If* $\Gamma, x{\in}A, y{\in}B, \Delta \vdash_\Sigma U{:}V$, *then* $\Gamma, y{\in}B, x{\in}A, \Delta \vdash_\Sigma U{:}V$, *provided* $x \notin FV(B)$, $y \notin FV(A)$ *and* $\Gamma \vdash_\Sigma B{:}\mathsf{Type}$*;*
2. Weakening*: If* $\Gamma \vdash_\Sigma U{:}V$ *and* $!\Delta \vdash_\Sigma A{:}\mathsf{Type}$*, then* $\Xi, x!A \vdash_\Sigma U{:}V$*, where* $[\Xi; \Gamma; !\Delta]$*;*
3. Dereliction*: If* $\Gamma, x{:}A \vdash_\Sigma U{:}V$*, then* $\Gamma', x!A \vdash_\Sigma U{:}V$*, where* Γ' *is* Γ *in which the free variables of* A *have been derelicted too;*
4. Contraction*: If* $\Gamma, x!A, y!A \vdash_\Sigma U{:}V$*, then* $\Gamma, x!A \vdash_\Sigma (U{:}V)[x/y]$*;*
5. Cut*: If* $\Pi(\Delta[N/x])$ *is a sub-proof of* $\Delta, x{:}A, \Delta' \vdash_\Sigma U{:}V$ *and* $\Gamma \vdash_\Sigma N{:}A$*, then* $\Xi \vdash_\Sigma (U{:}V)[N/x]$*, where* $[\Xi'; \Gamma; \Delta, \Delta'[N/x]]$ *and* $\Xi = \Xi' \backslash \kappa(\Gamma, (\Delta, \Delta'[N/x]))$*;*
6. Cut!*: If* $\Delta, x!A, \Delta' \vdash_\Sigma U{:}V$ *and* $!\Gamma \vdash_\Sigma N{:}A$*, then* $\Xi \vdash_\Sigma (U{:}V)[N/x]$*, where* $[\Xi; \Gamma; \Delta, \Delta'[N/x]]$.

Proof By induction on the structure of the proof of the premisses. We do some representative cases.

Admissibility of Weakening:

(Mc) $!\Gamma \vdash_\Sigma c{:}B$ because $\vdash_\Sigma !\Gamma$ context with $c!B \in \Sigma$. Then we construct

$$\cfrac{\cfrac{\vdash_\Sigma !\Gamma \text{ context} \quad !\Delta \vdash_\Sigma A{:}\mathsf{Type}}{\vdash_\Sigma \Xi, x!A \text{ context}}\,(\Gamma A!)}{\Xi, x!A \vdash_\Sigma c{:}B}\,(Mc)$$

with $[\Xi; !\Gamma; !\Delta]$ the side-condition for the $(\Gamma A!)$ application.

$(MVar)$ $\Gamma, y{:}B \vdash_\Sigma y{:}B$ because $\Gamma \vdash_\Sigma B{:}$Type. By induction hypothesis we have that

$$\Xi, x!A \vdash_\Sigma B{:}\text{Type}$$

Then we construct

$$\cfrac{\cfrac{\Xi, x!A \vdash_\Sigma B{:}\mathsf{Type}}{\Xi, x!A, y{:}B \vdash_\Sigma y{:}B}\,(MVar)}{\Xi, y{:}B, x!A \vdash_\Sigma y{:}B}\,\mathrm{X}$$

The exchange in the last step is possible as the B cannot depend on this particular x. If it did, then the initial judgement $\Gamma \vdash_\Sigma B{:}$Type would not be correct.

$(M\lambda\Lambda I)$ $\Gamma \vdash_\Sigma \lambda y{:}B.M : \Lambda y{:}B.C$ because $\Gamma, y{:}B \vdash_\Sigma M{:}C$. By induction hypothesis we have $\Xi, y{:}B, x!A \vdash_\Sigma M{:}C$. Then we construct

$$\cfrac{\cfrac{\Xi, y{:}B, x!A \vdash_\Sigma M{:}C}{\Xi, x!A, y{:}B \vdash_\Sigma M{:}C}\,\mathrm{X}}{\Xi, x!A \vdash_\Sigma \lambda y{:}B.C : \Lambda y{:}M.C}\,(M\lambda\Lambda I)$$

The exchange in the first step is possible as the A cannot be dependent on this particular y. As we have the judgement $\Delta \vdash_\Sigma A{:}$Type, any variable that A is dependent upon must be in Δ already.

$(M\Lambda\mathcal{E})$ $\Gamma \vdash_\Sigma MN : C[N/y]$ because $\Phi \vdash_\Sigma B : \Lambda y{:}B.C$ and $\Psi \vdash_\Sigma N{:}B$, with $[\Gamma; \Phi; \Psi]$. By induction hypothesis twice, we have

$$\Phi', x!A \vdash_\Sigma M : \Lambda y{:}B.C$$

and

$$\Psi', x!A \vdash_\Sigma N{:}B$$

where Φ' is the κ-sensitive join of Φ and $!\Delta$, and Ψ' is the κ-sensitive join of Ψ and $!\Delta$. Then we use $(M\Lambda\mathcal{E})$ to construct

$$\frac{\Phi', x!A \vdash_\Sigma M : \Lambda y{:}B.C \quad \Psi', x!A \vdash_\Sigma N : B}{\Xi, x!A \vdash_\Sigma MN : C[N/y]}$$

where $[\Xi';\Phi';\Psi']$ and $\Xi = \Xi' \backslash \kappa(\Phi', \Psi')$.

Admissibility of Dereliction:

$(MVar)$ $\Gamma, x{:}A \vdash_\Sigma x{:}A$ because $\Gamma \vdash_\Sigma A{:}\mathrm{Type}$. In the case where there are no free variables in A, we just use $(MVar!)$ to get $\Gamma, x!A \vdash_\Sigma x{:}A$.

Now suppose $y \in FV(A)$. If y is of an intuitionistic type then we are done. Otherwise, we consider the step where the y is introduced and replace the application of $(MVar)$ with $(MVar!)$;

$(M\lambda\Lambda I)$ $\Gamma, x{:}A \vdash_\Sigma \lambda y{:}B.M : \Lambda y{:}B.C$ because $\Gamma, x{:}A, y{:}B \vdash_\Sigma M{:}C$. There are two sub-cases, depending on how the linear variables $x{:}A$ is consumed.

1. $x \notin FV(B)$. That is, x has a linear occurrence in $M{:}C$. An Exchange puts the judgement in the form where we can apply the induction hypothesis. So we get $\Gamma', y{:}B, x!A \vdash_\Sigma M{:}C$, where Γ' is Γ with the $z \in LFV(A)$ are derelicted too. Then we apply Exchange and $(M\lambda\Lambda I)$ to get $\Gamma', x!A \vdash_\Sigma \lambda y{:}B.M : \Lambda y{:}B.C$.
2. $x \in FV(B)$. This case is argued similarly to the one before.

Admissibility of Contraction:

(Mc) Suppose $!\Gamma, x!A, y!A \vdash_\Sigma c{:}A$ because $\vdash_\Sigma !\Gamma, x!A, y!A$ context, with $c!A \in \Sigma$. The context validity judgement can only have been constructed by a series of applications of $(\Gamma A!)$, the last of which would be

$$\frac{\vdash_\Sigma !\Phi, x!A \text{ context} \quad !\Psi, x!A \vdash_\Sigma A{:}\mathrm{Type}}{\vdash_\Sigma !\Gamma, x!Ay!A \text{ context}}\,[!\Gamma; !\Phi; !\Psi]$$

where the most general (JOIN-!) clause has been used to join the contexts $!\Phi$ and $!\Psi$. The first premiss can be used to apply the (Ac) rule. And the use of as many Weakenings and Exchanges as necessary then allow us to construct

$$\dfrac{\dfrac{\vdash_\Sigma !\Phi, x!A \text{ context}}{!\Phi, x!A \vdash_\Sigma c{:}A}}{!\Gamma, x!A \vdash_\Sigma c{:}A}$$

which is equivalent to $c{:}A[x/y]$, the required result.

$(M\lambda\Lambda I)$ Suppose $\Gamma, x!A, y!A \vdash_\Sigma \lambda z{:}B.M\ \Lambda z{:}.BC$ because $\Gamma, x!A, y!A, z{:}B \vdash_\Sigma M{:}C$. For the case to proceed, we need to make some assumptions regarding the kinds of B and C. For example, consider the intuitionistic variable x: it might not occur at all (so, $B{:}\text{Type}$); it might linearly occur in B (so, $B{:}\Lambda x!A.\text{Type}$); it might linearly occur in $M{:}C$ (so the type of M is really $\Lambda x!A.\text{Type}$; lastly, both of the two previous cases could occur. The same argument applies in the case of y. We basically have to consider 2×4 sub-cases generated by these intuitionistic variables, ranging from the minimum $B{:}\text{Type}, C{:}B \multimap \text{Type}$, to the maximum $B{:}!A \multimap !A \multimap \text{Type}, C{:}!A \multimap !A \multimap (Bxy) \multimap \text{Type}$. We consider both of these cases below.

1. Suppose none of the exponential variables x and y are used. Modulo $(M\equiv)$, the last rule applied could only have been a $(M\Lambda\mathcal{E})$. The second, argument premiss of which could only have been derived by a $(MVar)$:

$$\dfrac{\Phi, x!A, y!A \vdash_\Sigma M'{:}\Lambda z'{:}B.C \qquad \dfrac{\Psi, x!A, y!A \vdash_\Sigma B{:}\text{Type}}{\Psi, x!A, y!A, z{:}B \vdash_\Sigma z{:}B}}{\Gamma, x!A, y!A, z{:}B \vdash_\Sigma M{:}C[z/z']}$$

Both premisses of the above derivation are now in a form to which the induction hypothesis can be applied. Using induction hypothesis twice, the definition of substitution, and the rules $(MVar)$ and $(A\Lambda\mathcal{E})$ allows us to reconstruct this derivation as

$$\dfrac{\Phi, x!A \vdash_\Sigma (M'{:}\Lambda z'{:}B.C)[x/y] \qquad \dfrac{\Psi, x!A \vdash_\Sigma (B{:}\text{Type})[x/y]}{\Psi, x!A, z{:}B \vdash_\Sigma (z{:}B)[x/y]}}{\Gamma, x!A, z{:}B \vdash_\Sigma (M{:}C[z/z'])[x/y]}$$

which is the required result.

2. This sub-case is argued similarly to the one above. We omit the details.

Admissibility of Cut:

($MVar$) $\Delta, x{:}A \vdash_\Sigma x{:}A$ because $\Delta \vdash_\Sigma A{:}\text{Type}$. We have to show that $\Xi \vdash_\Sigma (x{:}A)[M/x]$. This follows from the assumption $\Gamma \vdash_\Sigma M{:}A$ with the $\Xi = \Xi' \backslash \kappa(\Gamma, \Delta)$ side-condition needed to remove the excess occurrences in Γ which type A ;

($M\Lambda\mathcal{E}$) $\Delta, x{:}A, \Delta' \vdash_\Sigma MN : C[N/y]$ because $\Phi \vdash_\Sigma M : \Lambda y{:}B.C$ and $\Psi \vdash_\Sigma N{:}B$, with $[\Delta, x{:}A, \Delta'; \Phi; \Psi]$. There are two sub-cases to consider, depending on whether or not x is a shared variable, as regulated by κ.

1. For the non-sharing case, the proof proceeds according to which context the $x{:}A$ is sent to. So suppose $x{:}A \in \Phi$ (the case for $x{:}A \in \Psi$ is similar). By induction hypothesis we get $\Upsilon \vdash_\Sigma (M : \Lambda y{:}B.C)[M/x]$, where $[\Upsilon; \Phi_0, \Phi_n[M/x]; \Gamma]$ and $\Phi = \Phi_0, x{:}A, \Phi_n$. Then we use ($M\Lambda\mathcal{E}$) to construct $\Xi \vdash_\Sigma (MN : (C[N/z]))[M/x]$, with $[\Xi; \Upsilon; \Psi]$. We have elided the details of substitution.

2. For the sharing case, $x{:}A$ will be sent to both branches. That is, $x{:}A \in \Phi$ and $x{:}A \in \Psi$. The argument then proceeds as above, using the induction hypothesis on each branch.

   □

We now consider some standard type-theoretic properties, concluding with strong normalization and, hence, decidability.

The first part of the following lemma states that whenever we have a judgement of the general form $\Gamma \vdash_\Sigma U{:}V$ then the context Γ is valid. The usual way of stating this is to have the judgement $\vdash_\Sigma \Gamma$ context as the conclusion. But as the λΛ-calculus is weaker than intuitionistic type theories, all we can state is that the constituent parts of the context Γ are valid contexts and that they join together to form the context Γ. Contrast this result with Harper *et al.*'s Lemma A.2.2. The second part of the lemma states that only valid contexts are allowed to be constructed.

Lemma 15 (Subderivation property I)

1. *If* $\Gamma \vdash_\Sigma U{:}V$, *then* $\vdash_\Sigma \Gamma_0$ context, $\cdots \vdash_\Sigma \Gamma_n$ context, *where the* Γ_i *contexts join together, with respect to sharing, to form the context* Γ.

2. *If* $\vdash_\Sigma \Gamma, x{\in}A$ context, *then* $\Delta \vdash_\Sigma A$: Type, *with* $\Delta \subseteq \Gamma$.

Proof By induction on the structure of the proof of the premisses.

1. We do some of the representative cases.

   (Mc) $!\Gamma \vdash_\Sigma c{:}A$ because $\vdash_\Sigma !\Gamma$ context, with $c!A \in \Sigma$.

   $(MVar)$ $\Gamma, x{:}A \vdash_\Sigma x{:}A$ because $\Gamma \vdash_\Sigma A$: Type. Then we use (ΓA) to get

   $$\frac{\vdash_\Sigma \langle\rangle \text{ context} \quad \Gamma \vdash_\Sigma A\text{: Type}}{\vdash_\Sigma \Gamma, x{:}A \text{ context}}$$

   which is the required judgement.

   $(M\lambda\Lambda I)$ $\Gamma \vdash_\Sigma \lambda x{:}A.M\ \Lambda x{:}.AB$ because $\Gamma, x{:}A \vdash_\Sigma M{:}B$. By induction hypothesis, we get that $\vdash_\Sigma \Gamma_0 \cdots \vdash_\Sigma \Gamma_i, x{:}A$ context$\cdots \vdash_\Sigma \Gamma_n$ context. The i^{th} judgement could only have been formed by an application of the (ΓA) rule

   $$\frac{\vdash_\Sigma \Phi \text{ context} \quad \Psi \vdash_\Sigma A\text{:Type}}{\vdash_\Sigma \Gamma_i, x{:}A \text{ context}}$$

   with $[\Gamma_i'; \Phi; \Psi]$ and $\Gamma_i = \Gamma_i' \backslash \kappa(\Phi, \Psi)$. By induction hypothesis again, we have that $\vdash_\Sigma \Psi_0$ context$\cdots \vdash_\Sigma \Psi_m$ context, with the Ψ_j joining together to form the context Ψ.

   $(M\Lambda\mathcal{E})$ $\Xi \vdash_\Sigma (MN)B[N/x]$ because $\Gamma \vdash_\Sigma M\ \Lambda x{:}.AB$ and $\Delta \vdash_\Sigma N{:}A$, with $[\Xi'; \Gamma; \Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma, \Delta)$. By induction hypothesis twice we have that $\vdash_\Sigma \Gamma_i$ context and $\vdash_\Sigma \Delta_j$ context, with the Γ_i joining together to form Γ and the Δ_j joining together to form Δ.

2. There are only the two context-formation rules to consider.

   (ΓA) $\vdash_\Sigma \Xi, x{:}A$ context because $\vdash_\Sigma \Gamma$ context and $\Delta \vdash_\Sigma A$:Type, with $[\Xi; \Gamma; \Delta]$. The second of these premisses provides us with the required judgement.

   $(\Gamma A!)$ Similar to (ΓA) case above. $\vdash_\Sigma \Xi, x!A$ context because $\vdash_\Sigma \Gamma$ context and $\Delta \vdash_\Sigma A$:Type, with $[\Xi; \Gamma; \Delta]$. The second of these premisses provides us with the required judgement. □

The following lemma says that every inhabited term is either a kind or a type. It makes explicit use of the Cut(!) rules for the inductive cases.

Lemma 16 (Subderivation property II)

1. If $\Gamma \vdash_\Sigma A{:}K$*, then* $\Delta \vdash_\Sigma K$ Kind.

2. *If* $\Gamma \vdash_\Sigma M{:}A$, *then* $\Delta \vdash_\Sigma A{:}\mathsf{Type}$.

where $\Delta \subseteq \Gamma$.

Proof By induction on the structure on the proof of the premisses. We do some representative cases for the $M{:}A$-part of the lemma.

(Mc) $!\Gamma \vdash_\Sigma c{:}A$ because $\vdash_\Sigma !\Gamma$ context and $c!A \in \Sigma$. We unfold the context-forming derivation which has $\vdash_\Sigma !\Gamma$ context as its root until we arrive at the judgement $\vdash \Sigma\,\mathsf{sig}$, with $\Sigma = \Sigma_0, c!a, \Sigma_n\,\mathsf{sig}$, for some n. Again, we unfold this signature-forming derivation, until we arrive at

$$\dfrac{\vdash \Sigma_0\ \mathsf{sig} \qquad \vdash_{\Sigma_0} A{:}\mathsf{Type}}{\vdash \Sigma_0, c!A\ \mathsf{sig}} \\ \vdots\ (\Sigma K!),(\Sigma A!) \\ \vdash \Sigma_0, c!A, \Sigma_n\ \mathsf{sig}$$

which is the required result.

$(MVar)$ Immediate: $\Gamma, x{:}A$ because $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$.

$(M\lambda\Lambda I)$ $\Gamma \vdash_\Sigma \lambda x{:}A\,.M : \Lambda x{:}A\,.B$ because $\Gamma, x{:}A \vdash_\Sigma M{:}B$, with $\Gamma' \subseteq \Gamma$. By induction hypothesis we have that $\Gamma', x{:}A \vdash_\Sigma B{:}\mathsf{Type}$. Then we use $(A\Lambda I)$ to construct

$$\dfrac{\Gamma', x{:}A \vdash_\Sigma B{:}\mathsf{Type}}{\Gamma' \vdash_\Sigma \Lambda x{:}A\,.B : \mathsf{Type}}$$

$(M\Lambda\mathcal{E})$ $\Xi \vdash_\Sigma MN{:}B[N/x]$ because $\Gamma \vdash_\Sigma M\ \Lambda x{:}\,.AB$ and $\Delta \vdash_\Sigma N{:}A$, with $[\Xi';\Gamma;\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma,\Delta)$. By induction hypothesis we have that

$$\Gamma' \vdash_\Sigma \Lambda x{:}A\,.B : \mathsf{Type}$$

with $\Gamma' \subseteq \Gamma$. By inversion, we obtain

$$\Gamma', x{:}A \vdash_\Sigma B{:}\mathsf{Type}$$

as its premiss. Then we use the cut rule to construct

$$\dfrac{\Delta \vdash_\Sigma N{:}A \quad \Gamma', x{:}A \vdash_\Sigma B{:}\mathsf{Type}}{\Xi' \vdash_\Sigma B[N/x]{:}\mathsf{Type}}$$

with $[\Xi'';\Delta;\Gamma']$ and $\Xi' = \Xi'' \backslash \kappa(\Delta,\Gamma')$. It should be clear that $\Xi' \subseteq \Xi$. □

A natural algorithm for type-checking proceeds by calculating a type for a term, and then comparing this type to the assigned type. This approach relies on the Unicity of Types and Kinds property of the type system. The following lemma is useful in proving that Unicity.

Lemma 17 (Unicity of Domains) *If* $\Lambda x{\in}A.U \equiv \Lambda x{\in}B.V$, *then* $A \equiv B$ *and* $U \equiv V$.

Proof By the confluence of $\rightarrow^*$, there exists an E such that $\Lambda x{\in}A.U \rightarrow^* E$ and $\Lambda x{\in}B.V \rightarrow^* E$. This is only possible if E is of the form $\Lambda x{\in}C.W$ and $A \rightarrow^* C$, $B \rightarrow^* C$, $U \rightarrow^* W$ and $V \rightarrow^* W$, which implies the claim. □

In [Bar92], Unicity of Types is proved by relying on the PTS to be single-sorted or functional. The λΛ-calculus is single-sorted: we only have Type:Kind. But the following argument for Unicity relies more on the characteristic properties of the reduction relation $\rightarrow$.

Lemma 18 (Unicity of Types and Kinds, UT) *If* $\Gamma \vdash_\Sigma U{:}V$ *and* $\Gamma \vdash_\Sigma U{:}V'$, *then* $V \equiv V'$.

Proof By induction on the structure of the proof of the premisses. We do some representative cases for the $M{:}A$-fragment.

(Mc) Suppose $!\Gamma \vdash_\Sigma c{:}A$ and $!\Gamma \vdash_\Sigma c{:}A'$ because $\vdash_\Sigma !\Gamma$ context, $c!A \in \Sigma$ and $c!A' \in \Sigma$. As there can only be one constant a in Σ, $A = A'$, and so $A \equiv A'$.

$(MVar)$ Suppose $\Gamma, x{:}A \vdash_\Sigma x{:}A$ and $\Gamma, x{:}A \vdash_\Sigma x{:}A'$ because $\Gamma \vdash_\Sigma A{:}$Type. It is immediate that $A = A'$, and so $A \equiv A'$.

$(M\lambda\Lambda I)$ Suppose $\Gamma \vdash_\Sigma \lambda x{:}A.M\ \Lambda x{:}.AB$ and $\Gamma \vdash_\Sigma \lambda x{:}A.M\ \Lambda x{:}.AB'$ because $\Gamma, x{:}A \vdash_\Sigma M{:}B$ and $\Gamma x{:}A \vdash_\Sigma M{:}B'$ respectively. By induction hypothesis we get that $B \equiv B'$. The rule $(\rightarrow A\Lambda)$, applied as many times as necessary, then gives us the required result.

$(M\Lambda\mathcal{E})$ Suppose $\Xi \vdash_\Sigma MN{:}B[N/x]$ and $\Xi \vdash_\Sigma MN{:}B'[N'/x']$ because $\Gamma \vdash_\Sigma M\ \Lambda x{:}.AB$, $\Delta \vdash_\Sigma N{:}A$ $\Gamma \vdash_\Sigma M\ \Lambda x'{:}.A'B'$ and $\Delta \vdash_\Sigma N{:}A'$, where Ξ is the κ-sensitive join of Γ and Δ. By induction hypothesis twice, we get that $A \equiv A'$ and $\Lambda x{:}A.B \equiv \Lambda x'{:}A'.B'$. By Unicity of Domains we get that $A \equiv A'$ (which we knew already) and $B \equiv B'$. Using Lemma 10 then gives us the required result.

$(M\&I)$ Suppose $\Gamma \vdash_\Sigma \langle M,N\rangle{:}A\&B$ and $\Gamma \vdash_\Sigma \langle M,N\rangle{:}A'\&B'$ because $\Gamma \vdash_\Sigma M{:}A, \Gamma \vdash_\Sigma N{:}B$, $\Gamma \vdash_\Sigma M{:}A'$ and $\Gamma \vdash_\Sigma N{:}B'$. By induction hypothesis twice, we get that $A \equiv A'$ and $B \equiv B'$. The rule $(\rightarrow A\&)$, used as many times as necessary, then gives us the required result. □

Unicity of types and kinds, together with the CR property, allow a derivation of

Lemma 19 (Extended Unicity of Domains, EUD) *If $\lambda x{\in}A.U$ inhabits $\Lambda x{\in}B.V$, then $A \equiv B$.*

Proof CR determines, up to definitional equality, the term $\lambda x{\in}A.U$. UT does the same for the type $\Lambda x{\in}B.V$. This is sufficient to allow us to infer the result. □

Our definition (1) of linear occurrence is motivated by the desire for the type theory to have the subject reduction property. However, it is important that no linear variables are lost during reduction. We pause to consider this problem before proceeding to show the property. Consider the following instance of application:

$$\frac{\begin{array}{cc}\vdots & \vdots \\ \Gamma \vdash_\Sigma \lambda x{:}A.y : A \multimap B & z{:}A \vdash_\Sigma z{:}A\end{array}}{\Gamma, z{:}A \vdash_\Sigma (\lambda x{:}A.y)z : B} \tag{2.1}$$

We suppose that the type of the function is $A \multimap B$. After a β-reduction, we have $\Gamma, z{:}A \vdash_\Sigma y{:}B$, which leaves the $z{:}A$ hanging. Now, we supposed that $\Gamma \vdash_\Sigma \lambda x{:}A.y : A \multimap B$ be provable. By inversion, we must then have $\Gamma, x{:}A \vdash_\Sigma y{:}B$ provable. By our definition of linear occurrence, this can be so for one (or both) of the following reasons:

1. $x \in FV(y)$, which is not true in this case (but, in general, in simple types we may have a sufficiently complex M for all to be well);

2. $x \in FV(B)$, so the x is consumed by the B. That is, the type of the function $\lambda x{:}A.y$ is not $A \multimap B$ but rather $\Lambda x{:}A.B(x)$. So, in (2.1), the conclusion of the application is of the form $\Gamma, z{:}A \vdash_\Sigma (\lambda x{:}A.y)z : B[z/x]$ and hence we still have a linear occurrence of z.

So it follows that a situation as simple as (2.1), with the loss of an occurrence of a variable from the succedent, cannot arise in the type theory.

The subject reduction property is proved for $\to_1$, the one-step reduction relation in the basic type theory. It can be checked that $\to^*$ and $\to_1^*$ define the same relation.

Lemma 20 (Subject Reduction) *If $\Gamma \vdash_\Sigma U{:}V$ and $U \to_1 U'$, then $\Gamma \vdash_\Sigma U'{:}V$.*

Proof By simultaneous induction on the structure of the proof of the premisses. The two main cases are when the last step of the typing derivation is either rule $(M\Lambda\mathcal{E})$ or $(M\Lambda!\mathcal{E})$; and the last reduction step is rule $(\to M\beta)$. We consider the first of these cases. So, suppose

$$\Xi \vdash_\Sigma (\lambda x{:}A.M)N : B \quad \text{and} \quad (\lambda x{:}A.M)N \to_1 M[N/x]$$

and the first of these arises because

$$\Gamma \vdash_\Sigma \lambda x{:}A.M : \Lambda x{:}C.D \quad \text{and} \quad \Delta \vdash_\Sigma N{:}C$$

with $[\Xi';\Gamma;\Delta]$, $\Xi = \Xi' \backslash \kappa(\Gamma,\Delta)$ and $B = D[N/x]$.

By Lemma 13, we have that $\Gamma, x{:}A \vdash_\Sigma M{:}D$. By Lemma 19 we have that $A \equiv C$ and by the $(M \equiv)$ rule we have that $\Delta \vdash_\Sigma N{:}A$. Then we use the Cut rule to construct

$$\frac{\Gamma, x{:}A \vdash_\Sigma M{:}D \quad \Delta \vdash_\Sigma N{:}A \quad [\Xi';\Gamma;\Delta] \quad \Xi = \Xi' \backslash \kappa(\Gamma,\Delta)}{\Xi \vdash_\Sigma (M{:}D)[N/x]}$$

The conclusion is $M[N/x]{:}D[N/x]$. □

The type theory extended with $1{:}\top$, the unit of &, does not have the stated subject reduction property. The reason is illustrated by the following derivation, in which we assume that $A!\mathsf{Type} \in \Sigma$:

$$\dfrac{\dfrac{\dfrac{\vdots}{\vdash_\Sigma \Gamma, x{:}A \ \mathsf{context}}}{\Gamma, x{:}A \vdash_\Sigma 1{:}\top}}{\Gamma \vdash_\Sigma \lambda x{:}A.1 : A \multimap \top} \quad \dfrac{\vdots}{z{:}A \vdash_\Sigma z{:}A}$$
$$\overline{\Gamma, z{:}A \vdash_\Sigma (\lambda x{:}A.1)z : \top}$$

After a β-reduction we have $\Gamma, z{:}A \vdash_\Sigma 1{:}\top$, and the z is left hanging. However, we conjecture that such an extended type theory will have a weaker form of subject reduction, in which $\Gamma' \subseteq \Gamma$. The conjecture arises from a consideration of cut-elimination in linear type theory in the presence of $1{:}\top$. The point is that β-reduction in an example such as the one above effects not only terms but also proofs and so should therefore properly be considered an inference rule of the type theory.

All reduction sequences in the type theory terminate:

Theorem 21 (Strong Normalization) *All valid terms are strongly normalizing:*

1. *If* $\Gamma \vdash_\Sigma K$ Kind*, then* K *is strongly normalizing;*

2. *If* $\Gamma \vdash_\Sigma U{:}V$*, then* U *is strongly normalizing.*

The proof idea, again, a variation on an argument by Harper *et al.* [HHP93], is to define a faithful "dependency- and linearity-less" translation τ of kinds and type families to S, the set of simple types constructed by $\times$ and $\to$ over a given base type ω, and $|.|$, of type families and objects to $\Lambda(K)$, the set of untyped λ-terms over a set of constants $K = \{\pi_\sigma | \sigma \in S\}$. Let $\vdash^\lambda$ denote type assignment following Curry (with products) together with the infinite set of rules for K

$$\vdash^\lambda \pi_\sigma{:}\omega \to (\sigma \to \omega) \to \omega$$

for each $\sigma \in \Sigma$.

Definition 22 (Translation to simple types)

$$\begin{array}{lll}
\tau : K \to S & & \\
\tau(\mathrm{Type}) & = & \omega \\
\tau(\Lambda x{\in}A\,.K) & = & \tau(A) \to \tau(K) \\
 & & \\
\tau : A \to S & & \\
\tau(a) & = & a \\
\tau(\Lambda x{\in}A\,.B) & = & \tau(A) \to \tau(B) \\
\tau(\lambda x{\in}A\,.B) & = & \tau(B) \\
\tau(AM) & = & \tau(A) \\
\tau(A\&B) & = & \tau(A) \times \tau(B) \\
 & & \\
|.| : A \to \Lambda(K) & & \\
|a| & = & a \\
|\Lambda x{\in}A\,.B| & = & \pi_{\tau(A)}|A|(\lambda x.|B|) \\
|\lambda x{\in}A\,.B| & = & (\lambda y.\lambda x.|B|)|A| \qquad y \notin FV(B) \\
|AM| & = & |A||M| \\
|A\&B| & = & |A| \times |B| \\
 & & \\
|.| : M \to \Lambda(K) & & \\
|c| & = & c \\
|x| & = & x \\
|\lambda x{\in}A\,.M| & = & (\lambda y.\lambda x.|M|)|A| \qquad y \notin FV(M) \\
|MN| & = & |M||N| \\
|\langle M,N\rangle| & = & \langle |M|,|N|\rangle \\
|\pi_i M| & = & \pi_i|M| \qquad\qquad i \in \{0,1\}
\end{array}$$

The translation embeds the λΛ-calculus into such Curry-typable terms of the untyped λ-calculus in a structure preserving way. The dependency aspect is lost by, for instance, forgetting about the variable x in the term $\Lambda x{:}A\,.B$. The linear aspect is lost by translating linear and intuitionistic variables in exactly the same manner.

We refer also to Troelstra and Schwichtenberg for the technique of strong normalization by translation [TS96]. We will use this technique again when we remark on extending Church-Rosser to η-conversion, in Section 2.3.6 later.

We note some minor technicalities to do with the translation. The translation of $\Gamma \vdash_\Sigma U{:}V$ is given by $\tau(\Sigma), \tau(\Gamma) \vdash^\lambda |U|{:}\tau(V)$. The translation of the signature and context is the obvious one: $\tau(\langle\rangle) = \langle\rangle$; $\tau(\Gamma, x{\in}A) = \tau(\Gamma), x{:}\tau(A)$ and $\tau(\Sigma, c!A) = \tau(\Sigma), c{:}\tau(A)$. The binding strategy is utilized to give occurrences unique names.

The next two lemmata show that the translation is sufficiently faithful. We will abuse notation somewhat and take the symbols such $\equiv$, $\rightarrow$, $[M/x]$, *etc.* to mean similar relations in the simply-typed λ-calculus.

Lemma 23

1. *If* $A \equiv A'$, *then* $\tau(A) = \tau(A')$.

2. *If* $K \equiv K'$, *then* $\tau(K) = \tau(K')$.

Proof By induction on the structure of the proofs that if $A \rightarrow A'$, then $\tau(A) = \tau(A')$ and that if $K \rightarrow K'$ then, $\tau(K) = \tau(K')$. The lemma follows from the fact that $\equiv$ is defined to be the symmetric and transitive closure of $\rightarrow$. We do some representative cases.

$(\rightarrow refl)$ If $a \rightarrow a$, then $\tau(a) = \tau(a')$ immediately.

$(\rightarrow A\Lambda)$ $\Lambda x{\in}A\,.B \rightarrow \Lambda x{\in}A'\,.B'$ because $A \rightarrow A'$ and $B \rightarrow B'$. By induction hypothesis twice we have $\tau(A) = \tau(A')$ and $\tau(B) = \tau(B')$. Then

$$\begin{array}{rcll} \tau(\Lambda x{\in}A\,.B) & = & \tau(A) \rightarrow \tau(B) & \tau \\ & = & \tau(A') \rightarrow \tau(B') & IH \times 2 \\ & = & \tau(\Lambda x{\in}A'\,.B') & \tau \end{array}$$

$(\rightarrow A\lambda)$ $\lambda x{\in}A\,.B \rightarrow \lambda x{\in}A'\,.B'$ because $A \rightarrow A'$ and $B \rightarrow B'$. By induction hypothesis twice we have $\tau(A) = \tau(A')$ and $\tau(B) = \tau(B')$. Then

$$\begin{array}{rcll} \tau(\lambda x{\in}A\,.B) & = & \tau(B) & \tau \\ & = & \tau(B') & IH \\ & = & \tau(\lambda x{\in}A'\,.B') & \tau \end{array}$$

($\rightarrow Aapp$) $AM \rightarrow A'M'$ because $A \rightarrow A'$ and $M \rightarrow M'$. By induction hypothesis twice we have $\tau(A) = \tau(A')$ and $\tau(M) = \tau(M')$. Then

$$\begin{array}{rcll} \tau(AM) & = & \tau(A) & \tau \\ & = & \tau(A') & IH \\ & = & \tau(A'M') & \tau \end{array}$$

($\rightarrow A\beta$) $(\lambda x{\in}A.B)N \rightarrow B'[N'/x]$ because $B \rightarrow B'$ and $N \rightarrow N'$. By induction hypothesis twice we have $\tau(B) = \tau(B')$ and $\tau(N) = \tau(N')$. Then

$$\begin{array}{rcll} \tau((\lambda x{\in}A.B)N) & = & \tau(\lambda x{\in}A.B) & \tau \\ & = & \tau(B) & \tau \\ & = & \tau(B') & IH \\ & = & \tau((\lambda x{\in}A'.B')) & \tau \\ & = & \tau((\lambda x{\in}A'.B')N') & \tau \\ & = & \tau(B'[N'/x]) & \rightarrow \end{array}$$

($\rightarrow A\&$) $A\&B \rightarrow A'\&B'$ because $A \rightarrow A'$ and $B \rightarrow B'$. By induction hypothesis twice we have $\tau(A) = \tau(A')$ and $\tau(B) = \tau(B')$. Then

$$\begin{array}{rcll} \tau(A\&B) & = & \tau(A) \times \tau(B) & \tau \\ & = & \tau(A') \times \tau(B') & IH \times 2 \\ & = & \tau(A'\&B') & \tau \end{array}$$

□

Lemma 24

1. $|M[N/x]| = |M|[|N|/x]$.

2. $|B[N/x]| = |B|[|N|/x]$.

Proof By induction on the structure of M and B respectively. We do some representative cases.
$M = \lambda y{:}A.M$

$$\begin{array}{rcll} |(\lambda y{:}A.M)[N/x]| & = & |\lambda y{:}A[N/x].M[N/x]| & subs \\ & = & (\lambda z.\lambda y.|M[N/x]|)|A[N/x]| & |.| \\ & = & (\lambda z.\lambda y.|M|[|N|/x])|A|[|N|/x] & IH \times 2 \\ & = & (\lambda z.\lambda y.|M|)[|N|/x]|A|[|N|/x] & subs \\ & = & ((\lambda z.\lambda y.|M|)|A|)[|N|/x] & subs \\ & = & |\lambda y{:}A.M|[|N|/x] & |.| \end{array}$$

$M = PQ$

$$\begin{array}{rcll} |(PQ)[N/x]| & = & |P[N/x]Q[N/x]| & \textit{subs} \\ & = & |P[N/x]||Q[N/x]| & |.| \\ & = & |P|[|N|/x]|Q|[|N|/x] & IH \times 2 \\ & = & (|P||Q|)[|N|/x] & \textit{subs} \\ & = & |PQ|[|N|/x] & \tau \end{array}$$

$M = \langle P, Q\rangle$

$$\begin{array}{rcll} |(\langle P,Q\rangle[N/x]| & = & |\langle P[N/x], Q[N/x]\rangle| & \textit{subs} \\ & = & \langle |P[N/x]|, |Q[N/x]|\rangle & |.| \\ & = & \langle |P|[|N|/x], |Q|[|N|/x]\rangle & IH \times 2 \\ & = & \langle |P|, |Q|\rangle[|N|/x] & \textit{subs} \\ & = & |\langle P,Q\rangle|[|N|/x] & |.| \end{array}$$

□

The next lemma shows the consistency of the translation.

Lemma 25

1. *If* $\Gamma \vdash_\Sigma A{:}K$, *then* $\tau(\Sigma), \tau(\Gamma) \vdash^\lambda |A|{:}\tau(K)$.
2. *If* $\Gamma \vdash_\Sigma M{:}A$ *then*, $\tau(\Sigma), \tau(\Gamma) \vdash^\lambda |M|{:}\tau(A)$.

Proof By induction on the structure of the proof of the premisses. We illustrate the argument with a few representative cases.

(Mc) $!\Gamma \vdash_\Sigma c{:}A$ because $\vdash_\Sigma !\Gamma$ context, with $c!A \in \Sigma$. Trivial, as $\tau(A)$ is always a well-formed type.

$(M\lambda\Lambda I)$ $\Gamma \vdash_\Sigma \lambda x{:}A\,.M : \Lambda x{:}A\,.B$ because $\Gamma, x{:}A \vdash_\Sigma M{:}B$. By induction hypothesis we have that

$$\tau(\Sigma), \tau(\Gamma), x{:}\tau(A) \vdash^\lambda |M| : \tau(B)$$

Therefore

$$\tau(\Sigma), \tau(\Gamma) \vdash^\lambda \lambda x.|M| : \tau(A) \to \tau(B)$$

and

$$\tau(\Sigma), \tau(\Gamma) \vdash^\lambda (\lambda y.\lambda x.|M|)|A| : \tau(A) \to \tau(B)$$

which is $\tau(\Sigma), \tau(\Gamma) \vdash^\lambda |\lambda x{:}A\,.M| : \tau(\Lambda x{:}A\,.B)$.

($M\Lambda\mathcal{E}$) $\Xi \vdash_\Sigma MN : B[N/x]$ because $\Gamma \vdash_\Sigma M : \Lambda x{:}A.K$ and $\Delta \vdash_\Sigma N{:}A$, with $[\Xi';\Gamma;\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma,\Delta)$. By induction hypothesis twice we have

$$\tau(\Sigma),\tau(\Gamma) \vdash^\lambda |M| : \tau(A) \to \tau(B)$$

and

$$\tau(\Sigma),\tau(\Delta) \vdash^\lambda |N| : \tau(A)$$

Then we construct

$$\dfrac{\dfrac{\dfrac{\tau(\Sigma),\tau(\Gamma) \vdash^\lambda |M| : \tau(A) \to \tau(B)}{\tau(\Sigma),\tau(\Gamma),\tau(\Delta) \vdash^\lambda |M| : \tau(A) \to \tau(B)}\,W}{\tau(\Sigma),\tau(\Xi) \vdash^\lambda |M| : \tau(A) \to \tau(B)}\,X,C \qquad \dfrac{\dfrac{\tau(\Sigma),\tau(\Delta) \vdash^\lambda |N| : \tau(A)}{\tau(\Sigma),\tau(\Delta),\tau(\Gamma) \vdash^\lambda |N| : \tau(A)}\,W}{\tau(\Sigma),\tau(\Xi) \vdash^\lambda |N| : \tau(A)}\,X,C}{\tau(\Sigma),\tau(\Xi) \vdash^\lambda (|M||N|) : \tau(B)}\,\text{APP}$$

where the double line indicates a series of applications of the indicated rule. The weakenings (W) introduce $\tau(\Delta)$ and $\tau(\Gamma)$ into the left and right proofs respectively. The exchanges (X) and contractions (C) are used to eliminate duplicate (intuitionistic, in the original type theory) variables. These are necessary so as to get the premisses of the $\to$-elimination rule into additive form. The conclusion of the proof tree is $\tau(\Sigma),\tau(\Xi) \vdash^\lambda |MN| : \tau(B[N/x])$, as required. And $\tau(B) = \tau(B[N/x])$ as there is no type dependency in the simply-typed λ-calculus.

($M\&I$) $\Gamma \vdash_\Sigma \langle M,N \rangle : A\&B$ because $\Gamma \vdash_\Sigma M{:}A$ and $\Gamma \vdash_\Sigma N{:}B$. By induction hypothesis twice we have

$$\tau(\Sigma),\tau(\Gamma) \vdash^\lambda |M| : \tau(A)$$

and

$$\tau(\Sigma),\tau(\Gamma) \vdash^\lambda |N| : \tau(B)$$

The rule for $\times$-introduction then gives us that $\tau(\Sigma),\Gamma \vdash^\lambda \langle |M|,|N| \rangle : \tau(A) \times \tau(B)$, which is $\tau(\Sigma),\tau(\Sigma),\tau(\Gamma) \vdash^\lambda |\langle M,N \rangle| : \tau(A\&B)$. □

The extra combinatorial complexity of λΛ-calculus terms owing to the possibility of reductions within type labels is not lost by the translation.

Lemma 26

1. *If* $A \to_1 A'$, *then* $|A| \to_1^+ |A'|$.

2. *If* $M \to_1 M'$, *then* $|M| \to_1^+ |M'|$,

where $\rightarrow_1^+$ is the transitive closure of $\rightarrow_1$ for the untyped λ-calculus.

Proof By induction on the proof of $A \rightarrow_1 A'$ and $M \rightarrow_1 M'$. The only non-trivial cases arise when the last rule applied is one of the β-rules, or one of the Λ-rules. In the first case we have, for example,

$$|(\lambda x{:}A\,.M)N| \rightarrow_1^+ (\lambda x.|M|)|N| \rightarrow_1^+ |M|[|N|/x]$$

which is $|M[N/x]|$, by Lemma 24. In the second case, Lemma 23 suffices for the result. □

We can now give the proof of strong normalization. Suppose there was an infinite reduction in the λΛ-calculus. Then this would be translated into a reduction in the simply-typed λ-calculus. As the translation is faithful, the reduction in the simply-typed λ-calculus would be infinite too. But this cannot be so, as the simply-typed λ-calculus with pairing is known to be strongly normalizing [Gan80]. So there cannot be an infinite reduction in the λΛ-calculus.

Predicativity arises as a corollary of Theorem 21. Finally, we have:

Theorem 27 (Decidability) *All assertions of the λΛ-calculus are decidable.*

Proof The argument is the same as for the λΠ-calculus. We observe that, firstly, the complexity of the proof of a judgement is determined by proofs of strictly smaller measure; and, secondly, the form of a judgement completely determines its proof. The main method underlying this argument involves replacing the conversion rules with a (better behaved) normal-order reduction strategy. □

2.3.6 A remark on η-conversion

We remark that we can extend the definitional equality of the λΛ-calculus with η-conversion and show that decidability still holds for the resulting type theory.

Recall that decidability relies on Strong Normalization and Church-Rosser. The previous proof of Strong Normalization is modular, so can be retained. Church-Rosser, however, only holds for well-typed terms. In the λΠ-calculus case, establishing Church-Rosser involves a complicated argument assuming strengthening in order to prove a substantial amount of the metatheory before Church-Rosser for typed terms can be shown [HHP93, Sal90]. In our case, we can exploit the λΠ-calculus result to show Church-Rosser for the λΛ-calculus. The technique, inspired by the method to prove Strong Normalization, is to define a faithful and consistent translation of the λΛ-calculus into the λΠ-calculus and then reflect confluency for reduction in the λΠ-calculus back to reduction in the λΛ-calculus. (A minor technical issue in this translation is the translation of the & additive conjunction, which is not a λΠ-calculus connective. However, & can be added to the λΠ-calculus in the obvious way without effecting any of its meta-theoretic properties. We shall call the resulting system the λΠ&-calculus.)

Using the translation and its properties, we are able to show Church-Rosser for η-conversion. The argument is as follows. Suppose, for a contradiction, that we had a non-confluent term in the λΛ-calculus. We use the translation to take this term over to the λΠ&-calculus. Now as the translation is sound, we must have a non-confluent λΠ&-calculus term. But this cannot be, as reduction in the λΠ&-calculus is known to be confluent [Coq91, Sal90]. We conclude that there can be no such term in the λΛ-calculus.

It may also be possible to give a more direct proof of the decidability of the λΛ-calculus extended with η-conversion. This can be done following Salvesen's [Sal90] implementation of Harper *el al.*'s [HHP93] proof idea, as strengthening trivially holds for the linear parts of the context. However, the argument is complicated and relies on the equivalence between several different type theories of van Daalen [vD80]. The translation argument above is more than adequate for our purposes.

2.3.7 Related systems

In this section, we briefly compare the λΛ-calculus to the appropriate fragments of other linear type theories. Abramsky's [Abr93] and Benton's [Ben94] linear type theories are in propositions-as-types correspondence with a propositional ILL. Our concern is with a predicate ILL. Consider a linear version of the Barendregt cube, displayed in so-called standard orientation. Then Abramsky's and Benton's type theories correspond to the $\lambda\!\rightarrow$ and $\lambda 2$ nodes; our type theory corresponds to the λP node.

Another difference between Abramsky's and Benton's studies and ours is one of motivation; we study the λΛ-calculus as the language of a logical framework. A comparison with Cervesato and Pfenning's work [CP96] is, perhaps, more appropriate in this case. Their work claims to be inspired by our study's origins [Pym92]. We remark that the description of the LLF framework lacks an account of a notion of representation and that the $\lambda^{\Pi\multimap\&\top}$ type theory is a fragment of the λΛ-calculus lacking, *inter alia*, linear dependent function types. To be precise, the $\{\Pi, \multimap, \&\}$-fragment of $\lambda^{\Pi\multimap\&\top}$ can be recovered by restricting type-formation to be intuitionistic, with the consequence that the use of κ is vacuous. We have noted this restricted type theory in Section 2.3.3.

The key point to make in these comparisons is as follows. It is the construction of the linear dependent function space that necessitates an investigation into various structural properties. These are then explicated by the technical device of multiple occurrences. If our concern were non-dependent ($\multimap, \rightarrow$) or intuitionistic dependent (Π) function spaces, then we could do without such analyses.

2.4 Conservativity

In this section we show that RLF is a conservative extension of LF. We will need the following translation between the $\lambda\Pi$- and $\lambda\Lambda$-calculi. This is reminiscent of the translation of IL into ILL which maps $\phi \rightarrow \psi$ to $!\phi \multimap \psi$ [Gir87].

Definition 28 ($\ulcorner - \urcorner : \lambda\Pi \rightarrow \lambda\Lambda$) *We first define a translation for signatures and contexts. The clauses capture the intuitionistic–linear distinction between the two languages; the image is always of an intuitionistic type.*

$$\ulcorner\langle\rangle\urcorner = \langle\rangle \qquad \begin{aligned} \ulcorner\Sigma, c{:}U\urcorner &= \ulcorner\Sigma\urcorner, c!U \\ \ulcorner\Gamma, x{:}A\urcorner &= \ulcorner\Gamma\urcorner, x!A \end{aligned}$$

For the succedent of the typing judgement, $\ulcorner - \urcorner$ is defined by induction on the structure of the conclusion. We give only the cases for typed objects, $M{:}A$; the other cases are similar.

$$\begin{aligned} \ulcorner c{:}A\urcorner &= c{:}A \\ \ulcorner x{:}A\urcorner &= x{:}A \end{aligned} \qquad \begin{aligned} \ulcorner\lambda x{:}A\,.M : \Pi x{:}A\,.B\urcorner &= \lambda x!A\,.\ulcorner M\urcorner : \Lambda x!A\,.\ulcorner B\urcorner \\ \ulcorner MN : B\urcorner &= \ulcorner M\urcorner\ulcorner N\urcorner : \ulcorner B\urcorner \end{aligned}$$

The abstraction clause deals with the fact that the binding $x{:}A$ is a negative occurrence of a variable.

Now, our argument must capture the property of conservative extension not only at the level of the type theory but also at the level of a framework.[5] That is, we need to consider, for an arbitrary object-logic L, a translation from its definition in LF, via an encoding $\mathcal{E}$ and signature

[5]Conservativity at the level of the type theory is an immediate consequence of Definition 28.

Σ_L, to its definition in RLF, via an encoding $\mathcal{E}'$ and signature Σ'_L, where both $\mathcal{E}$ and $\mathcal{E}'$ are standard judgements-as-types encodings.

Lemma 29 (Conservativity) *Let L be an object-logic. Let $\mathcal{E}$ be a uniform encoding of L in LF. For every provable L-consequence $(X)\Delta \vdash_L \phi$, if*

$$\mathcal{E}(X), \mathcal{E}(\Delta) \vdash^{\lambda\Pi}_{\Sigma_L} M{:}\mathcal{E}(\phi) \tag{2.2}$$

then there is a uniform encoding $\mathcal{E}'$

$$\mathcal{E}'(X), \mathcal{E}'(\Delta) \vdash^{\lambda\Lambda}_{\Sigma'_L} M'{:}\mathcal{E}'(\phi).$$

Proof We define $\mathcal{E}'$ as follows:

$$\begin{array}{rcl} \mathcal{E}'(X) & = & \ulcorner\mathcal{E}(X)\urcorner \\ \mathcal{E}'(M') & = & \ulcorner\mathcal{E}(M)\urcorner \end{array} \qquad \begin{array}{rcl} \mathcal{E}'(\Delta) & = & \ulcorner\mathcal{E}(\Delta)\urcorner \\ \mathcal{E}'(\phi) & = & \ulcorner\mathcal{E}(\phi)\urcorner \end{array},$$

where M and M' are proof-realizers for the proposition ϕ in assumption $(X)\Delta$. We can see immediately that the diagram

$$\begin{tikzcd}
\vdash_L \arrow[d, "\mathcal{E}"'] \arrow[dr, dotted, "\mathcal{E}'"] & \\
\vdash^{\lambda\Pi}_{\Sigma_L} \arrow[r, "\ulcorner - \urcorner"'] & \vdash^{\lambda\Lambda}_{\Sigma'_L}
\end{tikzcd}$$

commutes. We now have to show that $\mathcal{E}'$ is a uniform encoding. The proof is by induction on the structure of (2.2). An interesting case is weakening. So suppose, $\Gamma, \Delta \vdash^{\lambda\Pi}_{\Sigma_L} M{:}A$ because $\Gamma \vdash^{\lambda\Pi}_{\Sigma_L} M{:}A$. Translating the latter consequence into the $\lambda\Lambda$-calculus gives us $!\Gamma \vdash^{\lambda\Lambda}_{\Sigma'_L} M'{:}A$. This can be weakened to get $!(\Gamma, \Delta) \vdash^{\lambda\Lambda}_{\Sigma'_L} M'{:}A$. From the definition of $\ulcorner - \urcorner$, this is the image of $\Gamma, \Delta \vdash^{\lambda\Pi}_{\Sigma_L} M{:}A$. Now, by assumption, $\mathcal{E}$ is a uniform encoding, so $\Gamma, \Delta \vdash^{\lambda\Pi}_{\Sigma_L} M{:}A$ is an image of some object-consequence. □

2.5 Example encodings

In this section, we illustrate several encodings in RLF. The intention is to bring out the essential characteristics of the $\lambda\Lambda$-calculus language — the weak structural properties, linear dependent function space and variable sharing — which allow these encodings to be undertaken uniformly via the judgements-as-types mechanism. The object-logic syntax and inference rules are not considered to be consumable resources and are encoded as (intuitionistic) constants in the signature.

We state representation theorems for each of the three encodings we undertake. In order to do this, we need a notion of *canonical* (essentially, long $\beta\eta$-normal) form. The definitions and lemmata needed for the characterization of canonical forms in the $\lambda\Lambda$-calculus are similar to that for the $\lambda\Pi$-calculus; we omit them from this presentation. In the following, we will often say that a function is a "compositional bijection"; this simply means that it is a bijection and commutes with substitution.

2.5.1 ILL

Our first encoding is that of the $\{\otimes, \&\}$-fragment of propositional intuitionistic linear logic (ILL). We will work through the ILL object-logic in slightly more detail than the others. In this encoding, we work with a restricted type theory in which type formation is entirely intuitionistic; we have discussed such a type theory in Section 2.3.3 previously. Such a restriction picks out the system of Cervesato and Pfenning [CP96] from amongst the others.

The natural deduction style rules for this logic are given in Table 2.5 below and are taken from Troelstra [Tro92]. The lower-case Greek letters ϕ, ψ, χ range over propositions of the ILL object-logic. For the rest of this sub-section, $i \in \{0,1\}$.

$$\dfrac{\begin{matrix}\Gamma & \Delta\\ \vdots & \vdots\\ \phi & \psi\end{matrix}}{\phi\otimes\psi}\,(\text{TENSOR-I}) \qquad \dfrac{\begin{matrix}\Gamma & \Delta,[\phi,\psi]\\ \vdots & \vdots\\ \phi\otimes\psi & \chi\end{matrix}}{\chi}\,(\text{TENSOR-E})$$

$$\dfrac{\begin{matrix}\Gamma & \Gamma\\ \vdots & \vdots\\ \phi & \psi\end{matrix}}{\phi\&\psi}\,(\text{WITH-I}) \qquad \dfrac{\begin{matrix}\Gamma\\ \vdots\\ \phi_0\&\phi_1\end{matrix}}{\phi_i}\,(\text{WITH-E}_i)$$

Table 2.5: A fragment of ILL

The signature Σ_{ILL} begins with the declarations $\iota!\text{Type}$ and $o!\text{Type}$ to represent the syntactic categories of individuals and propositions of ILL. Next, each of the two formula-constructors are declared as constants in the signature Σ_{ILL}:

$$\otimes ! o \multimap o \multimap o \qquad\qquad \& ! o \multimap o \multimap o\ .$$

Terms (formulae) are encoded by a function $\mathcal{E}_X$ which maps terms (formulae) with free variables in X to terms of type ι (o) in Σ_{ILL}, Γ_X:

$$\mathcal{E}_X(\phi \otimes \psi) \;=\; \otimes\ \mathcal{E}_X(\phi)\ \mathcal{E}_X(\psi) \qquad\qquad \mathcal{E}_X(\phi \& \psi) \;=\; \&\ \mathcal{E}_X(\phi)\ \mathcal{E}_X(\psi)\ .$$

There is one basic judgement, the judgement that the formula ϕ has a proof, $\vdash_{ILL} \phi\ \mathit{proof}$. This is represented by declaring the constant $\mathit{proof}!o \multimap \mathrm{Type}$ in the signature. A proof of a formula ϕ is represented by a term of type $\mathit{proof}(\mathcal{E}(\phi))$.

The multiplicative operator $\multimap$ is used to represent the inference in the object-logic. It is also used – as a curried version of a meta-logical multiplicative conjunction – to combine the representation of the premisses of the $\otimes$ rules, which are represented by the following declarations in the signature Σ_{ILL}:

$$\begin{array}{lll}
\text{TENSOR-I} & ! & \Lambda\phi,\psi!o.\mathit{proof}(\phi) \multimap \mathit{proof}(\psi) \multimap \mathit{proof}(\otimes(\phi,\psi)) \\
\text{TENSOR-E} & ! & \Lambda\phi,\psi,\chi!o.\mathit{proof}(\otimes(\phi,\psi)) \multimap \\
 & & \quad (\mathit{proof}(\phi) \multimap \mathit{proof}(\psi) \multimap \mathit{proof}(\chi)) \multimap \mathit{proof}(\chi)\ .
\end{array}$$

We need the distinguished additive operator & to represent the additive rules. An alternative might be to use an additive function space but such an investigation is beyond the scope of our current study. Recall, $i \in \{0,1\}$ in this sub-section.

$$\begin{array}{lll}
\text{WITH-I} & ! & \Lambda\phi,\psi!o.\mathit{proof}(\phi)\,\&\,\mathit{proof}(\psi) \multimap \mathit{proof}(\&(\phi,\psi)) \\
\text{WITH-E}_i & ! & \Lambda\phi_0,\phi_1!o.\mathit{proof}(\&(\phi_0,\phi_1)) \multimap \mathit{proof}(\phi_i)
\end{array}$$

Valid proofs of ILL are labelled trees with the constraint that assumption packets contain exactly one proposition and all such packets are uniquely labelled [Bie94]. A valid proof Π of ϕ, with respect to a proof context $(X)\Delta$,[6] is denoted by the assertion $(X)\Delta \vdash \Pi{:}\phi$, where X is a finite set of variables of first-order logic, Δ is the list of uniquely labelled assumptions $\{\xi_1{:}\phi_1,\ldots,\xi_n{:}\phi_n\}$, and $FV(dom(\Delta)) \subseteq X$. We remark that the treatment of the ILL quantifiers in RLF is essentially the same as that in LF. The rules for proving assertions of the form $(X)\Delta \vdash \Pi{:}\phi$ are given in Table 2.6 below.

The encoding function $\mathcal{E}_{(X)\Delta}$ can be defined to encode proofs of ILL. The two $\otimes$ cases, for instance, are as follows:

[6] As usual, linearity is at the level of *propositions*-as-types; the set of variables X is implicitly !-ed.

$$\frac{}{(X)\xi{:}\phi \vdash \mathrm{HYP}_{\phi}(\xi){:}\phi}$$

$$\frac{(X)\Delta \vdash \Pi{:}\phi \quad (X)\Delta' \vdash \Pi'{:}\psi}{(X)\Delta,\Delta' \vdash \text{TENSOR-I}_{\phi,\psi}(\Pi,\Pi'){:}\phi\otimes\psi} \qquad \frac{(X)\Delta \vdash \Pi{:}\phi\otimes\psi \quad (X)\Delta',\xi{:}\phi,\xi'{:}\psi \vdash \Pi'{:}\chi}{(X)\Delta,\Delta' \vdash \text{TENSOR-E}_{\phi,\psi,\chi}(\Pi,\xi,\xi'{:}\Pi'){:}\chi}$$

$$\frac{(X)\Delta \vdash \Pi{:}\phi \quad (X)\Delta \vdash \Pi'{:}\psi}{(X)\Delta \vdash \text{WITH-I}_{\phi,\psi}(\Pi,\Pi'){:}\phi\&\psi} \qquad \frac{(X)\Delta \vdash \Pi{:}\phi_0\&\phi_1}{(X)\Delta \vdash \text{WITH-E}_{i\,\phi_0,\phi_1}(\Pi){:}\phi_i}$$

Table 2.6: Some valid proof expressions of ILL

$$\begin{aligned}
\mathcal{E}_{(X)\Delta,\Delta'}(\text{TENSOR-I}_{\phi,\psi}(\Pi,\Pi')) &= \text{TENSOR-I}\ \mathcal{E}_X(\phi)\ \mathcal{E}_X(\psi)\ \mathcal{E}_{(X)\Delta}(\Pi) \\
&\quad\ \mathcal{E}_{(X)\Delta'}(\Pi') \\
\mathcal{E}_{(X)\Delta,\Delta'}(\text{TENSOR-E}_{\phi,\psi,\chi}(\Pi,\xi,\xi'{:}\Pi')) &= \text{TENSOR-E}\ \mathcal{E}_X(\phi)\ \mathcal{E}_X(\psi)\ \mathcal{E}_X(\chi) \\
&\quad\ \mathcal{E}_{(X)\Delta}(\Pi) \\
&\quad\ \lambda\xi{:}\mathit{proof}(\mathcal{E}_X(\phi))\,. \\
&\quad\ \lambda\xi'{:}\mathit{proof}(\mathcal{E}_X(\psi))\,.\mathcal{E}_{(X)\Delta'}(\Pi')
\end{aligned}$$

A proof context $(X)\Delta$, with $X = \{x_1,\ldots,x_m\}$ and $\Delta = \{\xi_1{:}\phi_1,\ldots,\xi_n{:}\phi_n\}$, is mapped to $\Gamma_{X,\Delta} = x_1{!}\iota,\ldots,x_m{!}\iota,\xi_1{:}\mathit{proof}(\mathcal{E}(\phi_1)),\ldots,\xi_n{:}\mathit{proof}(\mathcal{E}(\phi_n))$.

The encoding basically illustrates the propositions-as-types correspondence for a $\{\otimes,\&\}$-fragment of ILL. So we can expect a strong representation theorem.

Theorem 30 (Representation for ILL) *The encoding functions are compositional bijections. That is, for every ILL-formula* ϕ*:*

1. $X \vdash_{ILL} \phi$ *if and only if* $\Gamma_X \vdash_{\Sigma_{ILL}} \mathcal{E}_X(\phi){:}\iota$ *;*

2. $(X)\Delta \vdash_{ILL} \Pi{:}\phi$ *if and only if* $\Gamma_{X,\Delta} \vdash_{\Sigma_{ILL}} M_\Pi{:}\mathit{proof}(\mathcal{E}_X(\phi))$ *,*

where Π *is an ILL proof-object and* M_Π *is a canonical object of the* λΛ*-calculus.*

Proof The encoding functions $\mathcal{E}_X$ and $\mathcal{E}_{(X)\Delta}$ are clearly injective. Surjectivity is established by defining decoding functions $\mathcal{D}_X$ and $\mathcal{D}_{(X)\Delta}$ which are left-inverse to $\mathcal{E}_X$ and $\mathcal{E}_{(X)\Delta}$:

$$\begin{aligned}\mathcal{D}_X(\otimes\ M_1\ M_2) &= \mathcal{D}_X(M_1)\ \otimes\ \mathcal{D}_X(M_2)\\ \mathcal{D}_X(\&\ M_1\ M_2) &= \mathcal{D}_X(M_1)\ \&\ \mathcal{D}_X(M_2)\end{aligned}$$

$$\mathcal{D}_{(X)\Delta,\Delta'}\begin{pmatrix}\text{TENSOR-I}\ M_1\ M_2\\ P_1\ P_2\end{pmatrix} = \text{TENSOR-I}_{\mathcal{D}_{(X)\Delta}(M_1),\mathcal{D}_{(X)\Delta'}(M_2)}(\Pi_1,\Pi_2)$$

$$\mathcal{D}_{(X)\Delta,\Delta'}\begin{pmatrix}\text{TENSOR-E}\ M_1\ M_2\\ M_3\ P_1\\ \lambda\xi{:}\mathit{proof}(M_1).\\ \lambda\xi'{:}\mathit{proof}(M_2).P_2\end{pmatrix} = \begin{array}{l}\text{TENSOR-E}_{\mathcal{D}_{(X)\Delta}(M_1),\mathcal{D}_{(X)\Delta}(M_2),\mathcal{D}_{(X)\Delta'}(M_3)}\\ (\Pi_1,\xi,\xi'{:}\Pi_2)\end{array}$$

$$\begin{array}{l}\textit{where}\ \Pi_1 = \mathcal{D}_{(X)\Delta}(P_1)\\ \textit{and}\ \Pi_2 = \mathcal{D}_{(X)\Delta'}(P_2)\end{array}$$

That the decoding functions are total and well-defined follows from the definition of canonical forms and the signature. By induction on formulae and proof expressions, respectively, we get $\mathcal{D}_X(\mathcal{E}_X(\phi)) = \phi$ and $\mathcal{D}_{(X)\Delta}(\mathcal{E}_{(X)\Delta}(\Pi)) = \Pi$. Again, by a similar induction, we get that the encoding commutes with substitution. □

The encoding can be extended to deal with a $\{\top,\otimes,\oplus,\multimap,1\}$-fragment of propositional ILL. The representation of the ILL units forces the design of the type theory. A meta-logical $\top$ is required to directly represent the object-logic $\top$; for otherwise linearity constraints in the type theory would mean that an encoding of $\Gamma \vdash_{ILL} \top$ would not be a valid $\lambda\Lambda$-calculus judgement. The case for $\perp$ would be similar.

2.5.2 ML with references

Our second encoding is that of the programming language ML extended with references (MLR), a reworking of an example in Cervesato and Pfenning [Cer96, CP96]. In our reworking, we exploit the use of the Λ which is not available to Cervesato and Pfenning. Consequently, we are in the full $\lambda\Lambda$-calculus type theory, in which κ's action is non-trivial.

The basic MLR logic judgement is of the form $S \triangleright K \vdash_{MLR} i \longrightarrow a$ which means: the program i is evaluated with the store S and continuation K and leaves an answer (a store–expression pair) a. The signature Σ_{MLR} begins with the declarations *store*!Type, *cont*!Type, *instr*!Type and *ans*!Type to represent the syntactic categories of store, continuations, expressions and answers.

Evaluation is represented by the following declaration:

$$ev\,!\,cont \multimap instr \multimap answer \multimap \mathsf{Type}\,.$$

We are really only interested in the rule for evaluating re-assignment. This can be stated as follows:

$$\frac{S, c = v', S' \rhd K \vdash_{MLR} \bullet \longrightarrow A}{S, c = v, S' \rhd K \vdash_{MLR} \mathtt{ref}\, c := v' \longrightarrow A},$$

where • is the MLR unit expression.

The ML memory is modelled by a set of (cell,expression)-pairs. Each such pair is represented by a linear hypothesis of type *contains* which holds a lvalue (the cell) and its rvalue (the expression).

$$cell\,!\,\mathsf{Type} \qquad exp\,!\,\mathsf{Type} \qquad contains\,!\,cell \multimap exp \multimap \mathsf{Type}$$

The rule for re-assignment evaluation is encoded as follows:

$$\begin{aligned} \text{EV-REASS} \quad ! \quad & \Lambda c!cell\,.\,\Lambda v, v'!exp\,. \\ & ((contains\ c\ v') \multimap (ev\ K \bullet A)) \multimap \\ & (\Lambda v{:}exp\,.\,(contains\ c\ v)) \multimap (ev\ K\ (c := v')\ A) \end{aligned}$$

where the assignment instruction $c := v$ is shown in the usual (infix) form for reasons of readability. The rule can also be encoded in such a fashion that the linear property of the memory is formalized via the Λ quantifier. We will illustrate this idea soon. For now, based on our re-working of the MLR example, we can state the following by referring to [CP96].

Theorem 31 (Representation for MLR) *The encoding functions are compositional bijections. That is, for all stores S of shape $\langle c_1, v_1\rangle, \ldots, \langle c_n, v_n\rangle$, continuations K, instructions i and answers A (which are closed except for possible occurrences of free cells),*

$$S \rhd K \vdash_{MLR} \Pi{:}i \longrightarrow a \ \ \textit{if and only if}$$
$$\begin{bmatrix} c_1!cell, \ldots, c_n!cell, p_1{:}(contains\ c_1\ \mathcal{E}(v_1)), \\ \ldots, p_m{:}(contains\ c_m\ \mathcal{E}(v_m)) \end{bmatrix} \vdash_{\Sigma_{MLR}} M_\Pi{:}(ev\ \mathcal{E}(K)\ \mathcal{E}(i)\ \mathcal{E}(a)),$$

where Π is a proof object of MLR and M_Π is a canonical object of the λΛ-calculus. □

One property that it is desirable to show for the MLR logic is type preservation; in the context of a store Ω, if $S \triangleright K \vdash_{MLR} i \longrightarrow a$, i is a valid instruction of type τ, K is a valid continuation of type $\tau \to \tau'$ and S is a valid store, then a is a valid answer of type τ. The main difference in our reworking of this example is how the proof of type preservation for the EV-REASS rule, prEV-REASS, is encoded.

$$\begin{aligned}\text{prEV-REASS} \;\; ! \;\; & \Lambda c!cell\,.\,\Lambda v, v'!exp\,.\,\Lambda p{:}(contains\; c\; v)\,. \\ & (\Lambda p'{:}(contains\; c\; v')\,.\,(prCell\; p'\; c\; v') \multimap (ev\; K \bullet A)) \multimap \\ & (prCell\; p\; c\; v) \multimap (prEv\; K\; (x := v')\; A)\end{aligned}$$

In the above type, *prCell* and *prEv* are the proofs of type preservation over cells and for evaluations, respectively. We note that the types of p and p' have no linear free variables in them. That is, the type theory we have employed in the encoding does not involve the notion of sharing.

Now, the cells could have been quantified intuitionistically (as they are in [CP96]) instead of linearly. In that case, a sub-proof of $\Gamma \vdash_{\Sigma_{MLR}}$ prEV-REASS:U, where U is the above type of prEV-REASS, would consist of an instance of Π-introduction. But this would allow us to admit garbage: (cell,expression)-pairs which are occupying memory space but not being used. The linear quantification gives us a better representation of memory management. The above encoding realizes the intuition that we are making general statements about linear variables, so the Λ and not the Π quantifier should be used.

The encoded version of MLR type preservation can be stated and shown as in [CP96]. We omit the details.

2.5.3 A λ_I-calculus

Our last example is that of the equational theory of a type theory similar to Church's λ_I-calculus, in which abstraction is only allowed if the abstracted variable is free in the body of the function. We use the full expressiveness of the $\lambda\Lambda$-calculus type theory, with the crucial notion of variable sharing. This allows the Λ quantifier to capture the (traditional) notion of relevance. By contrast, in the encoding of the λ_I-calculus in Avron *et al.* [AHMP92], the relevance constraint is enforced by introducing extraneous language to axiomatize relevance in domain theory.

The signature Σ_{λ_I} begins with the declaration o!Type to represent the syntactic category of terms. The next three constants represent the object-logic abstraction and application operations, and the equality judgement:

$$\lambda_I ! (o \multimap o) \multimap o \qquad app ! o \multimap o \multimap o \qquad = ! o \multimap o \multimap \mathsf{Type} \;.$$

The axioms and rules of the equational theory of the relevant λ-calculus are encoded as follows:

$$\begin{array}{lll} E_0 & ! & \Lambda x{:}o\,.\,x = x \\ E_1 & ! & \Lambda x{:}o\,.\,\Lambda y{:}o\,.\,x = y \multimap y = x \\ E_2 & ! & \Lambda x{:}o\,.\,\Lambda y{:}o\,.\,\Lambda z{:}o\,.\,x = y \multimap y = z \multimap x = z \\ E_3 & ! & \Lambda x{:}o\,.\,\Lambda x'{:}o\,.\,\Lambda y{:}o\,.\,\Lambda y'{:}o\,.\,x = x' \multimap y = y' \multimap \\ & & \quad app(x,y) = app(x',y') \\ \beta & ! & \Lambda x{:}o \multimap o\,.\,\Lambda y{:}o\,.\,app(\lambda_I(x),y) = xy \end{array}$$

The first three constant declarations, E_0 to E_2, encode the reflexivity, symmetry and transitivity properties of the object-logic judgement, $=$. The constant declaration E_3 encodes the object-logic rule of congruence with respect to application. Finally, the constant declaration β encodes application.

Now, the definition of κ means that the $\Lambda x{:}o$ quantifies over all occurrences of x in its body. Like the ILL example before, the encoding is illustrating a propositions-as-types correspondence. This allows us to state a stronger representation theorem than that given in Avron *et al.* [AHMP92].

Theorem 32 (Representation for λ_I) *The encoding functions $\mathcal{E}$ are compositional bijections:*

1. $X \vdash_{\lambda_I} M$ *if and only if* $x_1{:}o,\ldots,x_n{:}o \vdash_{\Sigma_{\lambda_I}} \mathcal{E}_X(M){:}o$ *for* $x_i \in FV(M)$*; and*

2. $(M_1 = N_1),\ldots,(M_n = N_n) \vdash_{\lambda_I} \Pi{:}(M = N)$ *if and only if*

$$x_1{:}(\mathcal{E}(M_1) = \mathcal{E}(N_1)),\ldots,x_n{:}(\mathcal{E}(M_n) = \mathcal{E}(N_n)) \vdash_{\Sigma_{\lambda_I}} M_\Pi{:}(\mathcal{E}(M) = \mathcal{E}(N)) \;,$$

where Π is a proof object of λ_I and M_Π is a canonical object of the λΛ-calculus. □

2.6 Summary

In this chapter, we have studied a logical framework, RLF, for uniformly encoding natural deduction presentations of weak logics. We also studied the proof-theoretic meta-theory of the language of RLF, the λΛ-calculus, in some detail. Further work along these lines includes undertaking more object-logic encodings in RLF. We can also extend our study to the Barendregt-like cube of such type theories [Bar92], and to the hyper-cube of intuitionistic and relevant λ-cubes, with "diagonal" edges determined by a translation of the form considered in Definition 28.

Chapter 3

The propositions-as-types correspondence

3.1 Introduction

There is a conceptual similarity between the $\lambda\Lambda$-calculus type theory and the logic **BI**, particularly in the way that the two function spaces are formed. In this chapter, we set up a propositions-as-types correspondence between the type theory and a structural fragment of the logic. We follow Barendregt's [Bar92] treatment, where a propositions-as-types correspondence is set up between predicate logic and the λP type theory.

This chapter is organized as follows. In Section 3.2, we give Pym and O'Hearn's presentation of the logic **BI**. Then, in Section 3.3, we motivate and state certain restrictions on **BI**. It is with this fragment of the logic that the correspondence is set up; this is done in Section 3.4. In Section 3.5, we discuss some other correspondences of relevant logics.

3.2 The logic BI

We give a presentation of Pym and O'Hearn's logic **BI** of bunched implications [OP99]. The $\lambda\Lambda$-calculus is in propositions-as-types correspondence with a structural (we will say what this means in the next section) fragment of **BI**. We present the original logic for several reasons. Firstly, for reasons of completeness and interest. And, secondly, in order to clarify the exact nature of the restriction with which the $\lambda\Lambda$-calculus is in correspondence with.

BI restricts the structural rules of intuitionistic logic by decomposing implication into multiplicative and additive parts. The two implications, the linear one $A \multimap B$ and the intuitionistic one $A \to B$, arise from extra structure, called bunches following Read [Rea88], in the context. In Girard's linear logic, the structurals are restricted via the ! modality, so that only !-ed formulae may be weakened or contracted. In contrast, in **BI** there are two context-formation operators, one of which, the ";", admits the structural rules of weakening and contraction; the other, the ",", doesn't. In this scheme, contexts are not lists but trees with nodes labelled either by a "," or a ";" and the last leaf formation determines which kind of function space is formed:

$$(\multimap I)\,\frac{\Gamma,\phi \vdash_{\Sigma,\Xi} \psi}{\Gamma \vdash_{\Sigma,\Xi} \phi \multimap \psi} \qquad (\to I)\,\frac{\Gamma;\phi \vdash_{\Sigma,\Xi} \psi}{\Gamma \vdash_{\Sigma,\Xi} \phi \to \psi}$$

If the variable context is a bunch too, then we obtain two kinds of universal quantifiers:

$$(\forall_{\mathrm{new}} I)\,\frac{(X,x{:}C)\Gamma \vdash_{\Sigma,\Xi} \phi}{(X)\Gamma \vdash_{\Sigma,\Xi} \forall_{\mathrm{new}} x{:}C.\phi} \qquad (\forall I)\,\frac{(X;x{:}C)\Gamma \vdash_{\Sigma,\Xi} \phi}{(X)\Gamma \vdash_{\Sigma,\Xi} \forall x{:}C.\phi}$$

We now give Pym and O'Hearn's presentation of **BI**. Here, we restrict attention to the $\{\multimap\ ,\to\ ,\&\ ,\forall_{\mathrm{new}}\ ,\forall\}$-subset of the connectives as these have been our primary concern in the analysis of relevant natural deduction. **BI** itself has quite a full set of connectives, including disjunctions and existential quantifiers.

The language of the logic **BI** is defined over a structure. The following definition is a slight generalization of van Dalen's [vD94] and of Barendregt's [Bar92].

Definition 33 *A structure is a tuple consisting of non-empty sets called sorts, relations, typed functions and constants.*

Barendregt works exclusively with the example structure $\mathcal{A} = \langle A,B,P,Q,f,g,c\rangle$, where A and B are sets, $P \subseteq A$, $Q \subseteq A \times B$, $f{:}A \to A \to B$, $g{:}A \to B$ and $c \in A$. We will work more generally, over an arbitrary structure; it is easy to see how the language and logic can be made more particular to deal with pre-defined relations and functions.

The language of the logic begins by giving names to each component of the structure. We will omit this formality, confusing the sort, relation, function and constant symbols of the language with the components of the structure. That is, we will say "there is a relation in the language" for "there is a symbol in the language for a relation in the structure", *etc.*

The logic is defined over a term signature Σ and a predicate signature Ξ. These are essentially the set of functions (including 0-ary functions or constants) and the set of relations, respectively, which are components of the particular structure we are working with.

Let *x*, *y*, *etc.* range over a set of sorted variables. The set of **BI** terms, *Term*, is given by the following inductive definition:

$$t \quad ::= \quad x \quad \textit{variables}$$

If we had predefined functions in the language, then the set *Term* would be extended by $\ldots c \mid f(t) \mid g(t,t')$, where t and t' are terms of the appropriate sorts.

Let L denote a set of atomic propositional letters and let α, β, *etc.* range over L. The set of **BI** formulae over L, *Form*, is given by the following inductive definition:

$$\begin{array}{lcll} \phi & ::= & \alpha & \textit{atoms} \\ & \mid & \phi \multimap \psi & \textit{multiplicative implication} \\ & \mid & \forall_{\mathrm{new}} x{:}C.\phi & \textit{multiplicative quantification} \\ & \mid & \phi \& \psi & \textit{additive conjunction} \\ & \mid & \phi \to \psi & \textit{additive implication} \\ & \mid & \forall x{:}C.\phi & \textit{additive quantification} \end{array}$$

As for terms, if we had relations in the language, then the set *Form* would be extended by $\ldots \mid P(t) \mid Q(s,t)$, where s and t are terms of the appropriate sorts.

Bunches of variables, X, and of propositions, Γ, are given by the inductive definition:

$$\begin{array}{lcll} X & ::= & x{:}A & \textit{variable assumption} \\ & \mid & I & \textit{multiplicative unit} \\ & \mid & 1 & \textit{additive unit} \\ & \mid & X,Y & \textit{multiplicative combination} \\ & \mid & X;Y & \textit{additive combination} \end{array}$$

$$\begin{array}{lcll} \Gamma & ::= & \phi & \textit{propositional assumption} \\ & \mid & I & \textit{multiplicative unit} \\ & \mid & 1 & \textit{additive unit} \\ & \mid & \Gamma,\Delta & \textit{multiplicative combination} \\ & \mid & \Gamma;\Delta & \textit{additive combination} \end{array}$$

The main point regarding bunches is that ";" behaves additively, and admits weakening and contraction, whereas "," behaves multiplicatively. Bunches are structured as trees, with nodes labelled by ";" and "," and leaves labelled with propositions. We write $\Delta(\Phi)$ to refer to a subtree

Φ in Δ and $\Delta([\Phi'/\Phi]$ for Δ with Φ replaced by Φ'. We write $\Delta(-)$ to denote a bunch with a hole in it. We write $\Delta \cong \Delta'$ to denote the isomorphism under leaf re-labelling. We require that $(\Gamma, I, \text{“},\text{''})$ and $(\Gamma, 1, \text{“};\text{''})$ be commutative monoids. The isomorphism and commutative monoid equations form the coherence equivalence $\Delta \equiv \Delta'$ on bunches.

The main judgement of **BI** is $(X)\Delta \vdash_{\Sigma,\Xi} \phi$, which is read as "$\phi$ is a proposition in the variable context X and the propositional context Δ, with respect to the term signature Σ and to the propositional signature Ξ".

There are two auxiliary judgements, which are used to establish the well-formedness of terms and propositions. These two judgements are $X \vdash_\Sigma t{:}C$, read as "t is a term of type C in the variable context X with respect to the term signature Σ", and $X \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}$, read as "$\phi$ is a well-formed proposition in the variable context X with respect to the term signature Σ and to the propositional signature Ξ". We will omit the rules for these judgements, they being largely (though not exclusively; one must take care in combining contexts) apparent from the grammar of terms and formulae.

We remark that in the Axiom rule, the symmetric relation $Axiom(X,Y)$ records that X and Y are the same. This form of the rule is necessary for the maintenance of bunched structure in the variable context. The more familiar Axiom $(X)\phi \vdash_{\Sigma,\Xi} \phi$, with the side-condition that $(X) \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}$, is immediately derivable via an application of Substitution (Cut on terms).

Definition 34 *The judgement $(X)\Delta \vdash_{\Sigma,\Xi} \phi$ is defined by the following rules:*

Identity and structurals

$$Axiom\,\frac{(X) \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop} \quad (Y) \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}}{(Z)\phi \vdash_{\Sigma,\Xi} \phi}\; Z = X,Y \; or \; Z = X;Y \;,\; Axiom(X,Y)$$

$$Substitution\,\frac{(X(x{:}A))\Gamma \vdash_{\Sigma,\Xi} \phi \quad (Y) \vdash_{\Sigma,\Xi} t{:}A}{(X(Y)[u/x'])\Gamma[t/x][t/x'] \vdash_{\Sigma,\Xi} \phi[t/x][t/x']}$$

for all x' such that Axiom(x,x'), where u denotes I or 1 (here we abuse notation and write just Axiom(x,x') rather than pick out x and x' from arbitrary bunches).

$$W\,\frac{(X)\Gamma(\Delta) \vdash_{\Sigma,\Xi} \phi}{(X)\Gamma(\Delta;\Delta') \vdash_{\Sigma,\Xi} \phi} \qquad C\,\frac{(X)\Gamma(\Delta;\Delta) \vdash_{\Sigma,\Xi} \phi}{(X)\Gamma(\Delta) \vdash_{\Sigma,\Xi} \phi}$$

$$E\,\frac{(X)\Delta \vdash_{\Sigma,\Xi} \phi}{(X)\Delta' \vdash_{\Sigma,\Xi} \phi}\;(\Delta \equiv \Delta')$$

Multiplicatives

$$\multimap I\,\frac{(X)\Delta,\phi \vdash_{\Sigma,\Xi} \psi}{(X)\Delta \vdash_{\Sigma,\Xi} \phi \multimap \psi} \qquad \multimap E\,\frac{(X,Z)\Gamma \vdash_{\Sigma,\Xi} \phi \multimap \psi \quad (Z,Y)\Delta \vdash_{\Sigma,\Xi} \phi}{(X,Y)\Gamma,\Delta \vdash_{\Sigma,\Xi} \psi}\, X \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}$$

$$\forall_{\text{new}} I\,\frac{(X,x{:}C)\Delta \vdash_{\Sigma,\Xi} \phi}{(X)\Delta \vdash_{\Sigma,\Xi} \forall_{\text{new}} x{:}C.\phi} \qquad \forall_{\text{new}} E\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \forall_{\text{new}} x{:}C.\phi \quad (Y) \vdash_{\Sigma,\Xi} t{:}C}{(X,Y)\Gamma \vdash_{\Sigma,\Xi} \phi[t/x]}$$

Additives

$$\rightarrow I\,\frac{(X)\Delta;\phi \vdash_{\Sigma,\Xi} \psi}{(X)\Delta \vdash_{\Sigma,\Xi} \phi \rightarrow \psi} \qquad \rightarrow E\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \phi \rightarrow \psi \quad (Y)\Delta \vdash_{\Sigma,\Xi} \phi}{(X;Y)\Gamma;\Delta \vdash_{\Sigma,\Xi} \psi}$$

$$\&I\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \phi \quad (X)\Gamma \vdash_{\Sigma,\Xi} \psi}{(X)\Gamma \vdash_{\Sigma,\Xi} \phi\&\psi} \qquad \&E_i\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \phi_0\&\phi_1}{(X)\Gamma \vdash_{\Sigma,\Xi} \phi_i}\,(i = 1,2)$$

$$\forall I\,\frac{(X;x{:}C)\Delta \vdash_{\Sigma,\Xi} \phi}{(X)\Delta \vdash_{\Sigma,\Xi} \forall x{:}C.\phi} \qquad \forall E\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \forall x{:}C.\phi \quad (Y) \vdash_{\Sigma,\Xi} t{:}C}{(X;Y)\Gamma \vdash_{\Sigma,\Xi} \phi[t/x]}$$

The original introduction rule for & is given as

$$\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \phi \quad (Y)\Delta \vdash_{\Sigma,\Xi} \psi}{(X;Y)\Gamma;\Delta \vdash_{\Sigma,\Xi} \phi\&\psi}$$

But as ";" admits weakening and contraction, we take the simpler, more familiar rule in the definition. The actual equivalence of the two systems is a delicate issue.

The version of **BI** that we have presented here is the linear one, in which "," admits neither weakening nor contraction. One can define an affine version of **BI**, in which weakening is permitted for ",". Our concern is with the weakest, linear system.

A very interesting reading, a "sharing interpretation", of the two kinds of function spaces and quantifiers is given by O'Hearn and Pym [OP99, Pym98, O'H99b]. One of the domains of interpretation is in Reynold's Syntactic Control of Interference and Idealized Algol [Rey78, Rey81], where $\phi \multimap \psi$ is the type of a procedure which does not share store with arguments and $\phi \rightarrow \psi$ is the type of a procedure which might. Alternatively, the context Γ,Δ can be seen as partitioning the store into two separate components, while the context $\Gamma;\Delta$ allows some shared store. The other domain of interpretation is that of proof-search (the basis of logic programming) in the clausal hereditary Harrop fragment of **BI**. Resolution for the program clause $\forall_{\text{new}} x.G \multimap A$ creates a local answer substitution for the atom A. In particular, the subsequent proof-search for the goal G does not depend on this answer substitution. Compared with this, the (usual) Prolog-style resolution for the program clause $\forall x.G \rightarrow A$ creates global answer substitutions. Thus the two kinds of function spaces and quantifiers allow us to reason about sharing and interference in imperative and logic programming languages.

The reader is referred to the papers of Pym and O'Hearn for a more complete discussion of the logic.

3.3 A fragment of the logic BI

The first point to note is that both the logic and the type theory are parameterized over a signature, a set of constants of closed type. We can easily see how to translate in-between these signatures.

The main consideration in determining a correspondence between the λΛ-calculus and **BI** is to relate their respective contexts. This is an obvious first step to undertake as much of the structure of both the type theory and the logic arises from contexts. In the original presentation of **BI**, contexts are trees with internal nodes labelled by either "," or ";". In the type theory, on the other hand, a context can be extended in two ways, by a $x{:}A$ or a $x!A$. The basic idea for the correspondence is to consider, in some informal sense, ";" as intuitionistic extension and "," as linear extension:

λΛ-calculus	**BI**
$\Gamma, x{:}A$	Γ, A
$\Gamma, x!A$	$\Gamma; A$

The above is not quite the full story, as **BI** allows arbitrary context combination and not just context extension. But the idea will be to see, in a context Γ, Δ, the Δ as some singular proposition ϕ_Δ. This "packing" of Δ as ϕ_Δ is the restriction of **BI** that is the internal logic of the λΛ-calculus type theory, and it brings the internal logic closer to Linear Logic. The packing, together with currying, allows us to show the correspondence of derivations and proofs. We will consider this issue again when we discuss soundness.

There are two other points to consider. The first one is that due to the restricted nature of first order logic, in which term variables are separated from propositional variables, multiple occurrences and the requirement for the κ function are not logical issues. However, well-formedness of types is still an issue, as a consideration of the $\multimap E$ rule indicates:

$$\multimap E\,\frac{(X,Z)\Gamma \vdash_{\Sigma,\Xi} \phi \multimap \psi \quad (Z,Y)\Delta \vdash_{\Sigma,\Xi} \phi}{(X,Y)\Gamma,\Delta \vdash_{\Sigma,\Xi} \psi}\; Z \vdash_{\Sigma,\Xi} \phi$$

Consider the simplest situation corresponding to the above proof in the type theory. Suppose $\phi{:}A \multimap \mathsf{Type}, \psi{:}B \multimap \mathsf{Type} \in \Sigma$. Then we can construct the following:

$$\dfrac{\dfrac{\dfrac{\vdots}{x{:}A, y{:}B \vdash_\Sigma (\phi x) \multimap (\psi y){:}\mathsf{Type}}}{x{:}A, y{:}B, z{:}(\phi x) \multimap (\psi y) \vdash_\Sigma z{:}(\phi x) \multimap (\psi y)}\,(MVar) \quad \dfrac{\dfrac{\vdots}{x{:}A \vdash_\Sigma \phi x{:}\mathsf{Type}}}{x{:}A, w{:}(\phi x) \vdash_\Sigma w{:}(\phi x)}\,(MVar)}{x{:}A, y{:}B, z{:}(\phi x) \multimap (\psi y), x{:}A, w{:}\phi x \vdash_\Sigma zw{:}(\psi y)}\,(M\Lambda\mathcal{E})$$

The presence of the proof realizers z and w means that the variables corresponding to X in the logic are still needed. This exercise illustrates the difference between a predicate logic and its corresponding dependent type theory, in that the well-formedness side-condition of the logic judgement is already included in the proof of the type theoretic judgement.

The second point to consider is that the introduction rules for function spaces in the $\lambda\Lambda$-calculus correspond exactly to introduction rules for implication and universal quantification formation in **BI**. However, for the additive or intuitionistic elimination rules, we must impose the restriction that the context of the minor premiss be entirely extended with ";"s. We will use $!\Delta$ to denote such a context. Following along this line of argument, we also take two Substitution rules, one for cutting a linear proposition and one for cutting an intuitionistic proposition.

In the sequel, we use the following notation to identify the type, whether linear or intuitionistic, of a variable. We use $\Gamma(I,\phi)$ to identify a linear leaf of a bunch and $\Gamma(1;\phi)$ to identify an intuitionistic leaf of a bunch. More formally, we should say that $\Gamma = \Psi(\Phi \bullet \phi)$, where $\bullet$ is either "," or ";". The former is a slightly better means of "pattern-matching" a linear leaf and is equivalent, using the coherence isomorphism, to the more formal description.

We are now ready to present the restricted logic.

Definition 35 *The formula judgement is defined by the following rules.*

Identity and structurals

$$Axiom\ \dfrac{X \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop} \quad Y \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}}{(Z)\phi \vdash_{\Sigma,\Xi} \phi}\ Z = X,Y \ or\ Z = X;Y\ ,\ Axiom(X,Y)$$

$$MultSubstitution\ \dfrac{(X(I,x{:}A))\Gamma \vdash_{\Sigma,\Xi} \phi \quad (Y) \vdash_{\Sigma,\Xi} t{:}A}{(X(Y)[I/x'])\Gamma[t/x][t/x'] \vdash_{\Sigma,\Xi} \phi[t/x][t/x']}\ Axiom(x,x')$$

$$AddSubstitution\ \dfrac{(X(1;x{:}A))\Gamma \vdash_{\Sigma,\Xi} \phi \quad (!Y) \vdash_{\Sigma,\Xi} t{:}A}{(X(!Y)[1/x'])\Gamma[t/x][t/x'] \vdash_{\Sigma,\Xi} \phi[t/x][t/x']}\ Axiom(x,x')$$

$$W\,\frac{(X)\Delta \vdash_{\Sigma,\Xi} \phi}{(X;x{:}A)\Delta;\phi \vdash_{\Sigma,\Xi} \phi} \qquad C\,\frac{(X)\Delta(1;\phi;\phi) \vdash_{\Sigma,\Xi} \phi}{(X)\Delta(1;\phi) \vdash_{\Sigma,\Xi} \phi}$$

$$E\,\frac{(X)\Delta \vdash_{\Sigma,\Xi} \phi}{(X')\Delta' \vdash_{\Sigma,\Xi} \phi}\,(\Delta \equiv \Delta')$$

Multiplicatives

$$-\!\!\circ I\,\frac{(X)\Delta,\phi \vdash_{\Sigma,\Xi} \psi}{(X)\Delta \vdash_{\Sigma,\Xi} \phi \multimap \psi} \qquad -\!\!\circ E\,\frac{(X,Z)\Gamma \vdash_{\Sigma,\Xi} \phi \multimap \psi \quad (Z,Y)\Delta \vdash_{\Sigma,\Xi} \phi}{(X,Y)\Gamma,\Delta \vdash_{\Sigma,\Xi} \psi}\,Z \vdash_{\Sigma,\Xi} \phi{:}\mathrm{Prop}$$

$$\forall_{\mathrm{new}} I\,\frac{(X,x{:}C)\Delta \vdash_{\Sigma,\Xi} \phi}{(X)\Delta \vdash_{\Sigma,\Xi} \forall_{\mathrm{new}} x{:}C.\phi} \qquad \forall_{\mathrm{new}} E\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \forall_{\mathrm{new}} x{:}C.\phi \quad (Y) \vdash_{\Sigma,\Xi} t{:}C}{(X,Y)\Gamma \vdash_{\Sigma,\Xi} \phi[t/x]}$$

Additives

$$\&I\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \phi \quad (X)\Gamma \vdash_{\Sigma,\Xi} \psi}{(X)\Gamma \vdash_{\Sigma,\Xi} \phi\&\psi} \qquad \&E_i\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \phi_0\&\phi_1}{(X)\Gamma \vdash_{\Sigma,\Xi} \phi_i}\,(i=1,2)$$

$$\to I\,\frac{(X)\Delta;\phi \vdash_{\Sigma,\Xi} \psi}{(X)\Delta \vdash_{\Sigma,\Xi} \phi \to \psi} \qquad \to E\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \phi \to \psi \quad (!Y)!\Delta \vdash_{\Sigma,\Xi} \phi}{(X;!Y)\Gamma;!\Delta \vdash_{\Sigma,\Xi} \psi}$$

$$\forall I\,\frac{(X;x{:}C)\Delta \vdash_{\Sigma,\Xi} \phi}{(X)\Delta \vdash_{\Sigma,\Xi} \forall x{:}C.\phi} \qquad \forall E\,\frac{(X)\Gamma \vdash_{\Sigma,\Xi} \forall x{:}C.\phi \quad (!Y) \vdash_{\Sigma,\Xi} t{:}C}{(X;!Y)\Gamma \vdash_{\Sigma,\Xi} \phi[t/x]}$$

We formalize the concept of deduction in this fragment **BI**. This will allow us to show a very precise correspondence between formulae and types, and between terms and proofs. Proof expressions of the logic are given by the following grammar:

$$\begin{aligned}\Pi \ ::=\ & \mathrm{LAX}_{\phi}(\xi) \mid \mathrm{IAX}_{\phi}(\xi) \mid \mathrm{LIMP\text{-}I}_{\phi,\psi}(\xi{:}\Pi) \mid \mathrm{LIMP\text{-}E}_{\phi,\psi}(\Pi,\Pi') \mid \\ & \mathrm{IIMP\text{-}I}_{\phi,\psi}(\xi{:}\Pi) \mid \mathrm{IIMP\text{-}E}_{\phi,\psi}(\Pi,\Pi') \mid \mathrm{WITH\text{-}I}_{\phi,\psi}(\Pi,\Pi') \mid \mathrm{WITH\text{-}E}^{i}_{\phi,\psi}(\Pi) \mid \\ & \mathrm{LALL\text{-}I}_{x,\phi}(\Pi) \mid \mathrm{LALL\text{-}E}_{x,\phi,t}(\Pi) \mid \mathrm{IALL\text{-}I}_{x,\phi}(\Pi) \mid \mathrm{IALL\text{-}E}_{x,\phi,t}(\Pi)\end{aligned}$$

In the above, ξ ranges over a countably infinite set of occurrence markers. The idea will be to label each discharged occurrence uniquely with an occurrence marker.

Proof expressions are directly related both to Barendregt's derivation combinators [Bar92] and to O'Hearn–Pym's proof-objects [OP99]. For instance, the $\mathrm{IALL\text{-}I}_{x,\phi}(\psi)$ proof expression corresponds to Barendregt's G combinator and to O'Hearn–Pym's λ-abstraction operator.

The judgement $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi$ is read as "Π is a derivation of ϕ with respect to the context $(X)\Delta$". The rules for valid proof expressions are given below.

Definition 36 *BI derivations are defined as follows:*

Identities and structurals

$$MultId\ \frac{X \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop} \quad Y \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}}{(Z)I, x{:}\phi \vdash_{\Sigma,\Xi} \mathit{LAx}_{\phi}(x){:}\phi}\ Z = X,Y \text{ or } Z = X;Y\ ,\ Axiom(X,Y)$$

$$AddId\ \frac{X \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop} \quad Y \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}}{(Z)1; x{:}\phi \vdash_{\Sigma,\Xi} \mathit{IAx}_{\phi}(x){:}\phi}\ Z = X,Y \text{ or } Z = X;Y\ ,\ Axiom(X,Y)$$

$$MultSubstitution\ \frac{(X(I,x{:}A))\Gamma \vdash_{\Sigma,\Xi} \Phi{:}\phi \quad (Y) \vdash_{\Sigma,\Xi} t{:}A}{(X(Y)[I/x'])\Gamma[t/x][t/x'] \vdash_{\Sigma,\Xi} (\Phi{:}\phi)[t/x][t/x']}\ Axiom(x,x')$$

$$AddSubstitution\ \frac{(X(1;x{:}A))\Gamma \vdash_{\Sigma,\Xi} \Phi{:}\phi \quad (!Y) \vdash_{\Sigma,\Xi} t{:}A}{(X(!Y)[1/x'])\Gamma[t/x][t/x'] \vdash_{\Sigma,\Xi} (\Phi{:}\phi)[t/x][t/x']}\ Axiom(x,x')$$

$$W\ \frac{(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi}{(X;x{:}A)\Delta; y{:}\phi \vdash_{\Sigma,\Xi} \Pi{:}\phi} \qquad C\ \frac{(X)\Delta(1;x{:}\phi;y{:}\phi) \vdash_{\Sigma,\Xi} \Pi{:}\phi}{(X)\Delta(1;x{:}\phi) \vdash_{\Sigma,\Xi} (\Pi{:}\phi)[x/y]}$$

$$E\ \frac{(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi}{(X')\Delta' \vdash_{\Sigma,\Xi} \Pi{:}\phi}\ (\Delta \equiv \Delta')$$

Multiplicatives

$$\multimap I\ \frac{(X)\Delta, x{:}\phi \vdash_{\Sigma,\Xi} \Pi{:}\psi}{(X)\Delta \vdash_{\Sigma,\Xi} \mathit{LIMP\text{-}I}_{\phi,\psi}(\Pi){:}\phi \multimap \psi}$$

$$\multimap E\ \frac{(X,Z)\Gamma \vdash_{\Sigma,\Xi} \Pi{:}\phi \multimap \psi \quad (Z,Y)\Delta \vdash_{\Sigma,\Xi} \Pi'{:}\phi}{(X,Y)\Gamma,\Delta \vdash_{\Sigma,\Xi} \mathit{LIMP\text{-}E}_{\psi,\phi}(\Pi,\Pi'){:}\psi}\ Z \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}$$

$$\forall_{\text{new}} I\ \frac{(X,x{:}C)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi}{(X)\Delta \vdash_{\Sigma,\Xi} \mathit{LALL\text{-}I}_{x,\phi}(\Pi){:}\forall_{\text{new}} x{:}C\,.\,\phi} \qquad \forall_{\text{new}} E\ \frac{(X)\Gamma \vdash_{\Sigma,\Xi} \Pi{:}\forall_{\text{new}} x{:}C\,.\,\phi \quad (Y) \vdash_{\Sigma,\Xi} t{:}C}{(X,Y)\Gamma \vdash_{\Sigma,\Xi} \mathit{LALL\text{-}E}_{x,\phi,t}(\Pi){:}\phi[t/x]}$$

Additives

$$\&I\ \frac{(X)\Gamma \vdash_{\Sigma,\Xi} \Pi{:}\phi \quad (X)\Gamma \vdash_{\Sigma,\Xi} \Pi'{:}\psi}{(X)\Delta \vdash_{\Sigma,\Xi} \mathit{WITH\text{-}I}_{\phi,\psi}(\Pi,\Pi'){:}\phi \& \psi} \qquad \&E_i\ \frac{(X)\Gamma \vdash_{\Sigma,\Xi} \Pi{:}\phi_0 \& \phi_1}{(X)\Gamma \vdash_{\Sigma,\Xi} \mathit{WITH\text{-}E}^{i}_{\phi_0,\phi_1}(\Pi){:}\phi_i}\ (i = 1,2)$$

$$\rightarrow I\ \frac{(X)\Delta;\phi \vdash_{\Sigma,\Xi} \Pi{:}\psi}{(X)\Delta \vdash_{\Sigma,\Xi} \mathit{IIMP\text{-}I}_{\phi,\psi}(\Pi){:}\phi \rightarrow \psi} \qquad \rightarrow E\ \frac{(X)\Gamma \vdash_{\Sigma,\Xi} \Pi{:}\phi \rightarrow \psi \quad (!Y)!\Delta \vdash_{\Sigma,\Xi} \Pi'{:}\phi}{(X;!Y)\Gamma;!\Delta \vdash_{\Sigma,\Xi} \mathit{IIMP\text{-}E}_{\phi,\psi}(\Pi,\Pi'){:}\psi}$$

$$\forall I\ \frac{(X;x{:}C)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi}{(X)\Delta \vdash_{\Sigma,\Xi} \mathit{IALL\text{-}I}_{x,\phi}(\Pi){:}\forall x{:}C\,.\,\phi} \qquad \forall E\ \frac{(X)\Gamma \vdash_{\Sigma,\Xi} \Pi{:}\forall x{:}C\,.\,\phi \quad (!Y) \vdash_{\Sigma,\Xi} t{:}C}{(X;!Y)\Gamma \vdash_{\Sigma,\Xi} \mathit{IALL\text{-}E}_{x,\phi,t}(\Pi){:}\phi[t/x]}$$

The above proof rules are subject to the same side-conditions as the previous stated logical rules.

3.4 The correspondence

In this section, we will set up the correspondence between **BI** and the $\lambda\Lambda$-calculus. The basic idea is as follows: terms of the logic correspond to elements of the type theory; formulae of the logic correspond to types of the type theory; and a deduction of a formula ϕ in the logic corresponds to an element of the type corresponding to ϕ in the type theory. We will define two functions; $\mathcal{T}$, from the logic to the $\mathcal{T}$ype theory and $\mathcal{L}$, from the type theory to the $\mathcal{L}$ogic. These functions will, in fact, be inverses.

Before we give the translations, we need to say how components of the particular structure we are working with are dealt with within the type theory. Barendregt's method is to define a so-called canonical context corresponding to the particular structure. For instance, for $\mathcal{A} = \langle A, B, P, Q, f, g, c\rangle$, the canonical context is

$$\Gamma_{\mathcal{A}} = A{:}\mathsf{Type}, B{:}\mathsf{Type}, P{:}A \to \mathsf{Type}, Q{:}A \to B \to \mathsf{Type}, f{:}A \to (A \to A), g{:}A \to B, c{:}A.$$

But in both the logic and the type theory we distinguish between a signature and a context, and place the declarations corresponding to the components of the structure as constants in the signature. The translations between the logic signature Σ, Ξ and the type theory signature Σ are easy to see.

We now define translations from terms, formulae and derivations of **BI** to elements, types and terms of the $\lambda\Lambda$-calculus, over a given context. This will enable us to show soundness.

We consider the translation of contexts first. This is quite a crucial matter, and it can't be done as loosely as Barendregt's treatment for intuitionistic logic [Bar92]. The translation allows us to see, in a certain way, the second sub-bunch as a series of propositional extensions.

Definition 37 *Let $\Delta = \delta_1 \bullet \ldots \bullet \delta_n$ be a bunch. The function $\mathcal{T}$ is defined by induction on the structure of bunches as follows:*

$$\begin{array}{rcl} \mathcal{T}(I) & = & \langle\rangle \\ \mathcal{T}(1) & = & \langle\rangle \\ \mathcal{T}(\Gamma, \Delta) & = & \Gamma, x_1{:}\delta_1, \ldots x_n{:}\delta_n \\ \mathcal{T}(\Gamma; \Delta) & = & \Gamma, x_1!\delta_1, \ldots x_n!\delta_n \end{array}$$

where the x_i are new variables.

The translation of bunches forces us to consider a particular fragment of **BI**. This is the fragment in which the bunch Γ, Δ is replaced by the bunch $\Gamma, \phi^{\otimes}_{\Delta}$, where $\phi^{\otimes}_{\Delta}$ is Δ with both the "," and ";" substituted by $\otimes$. And, similarly, the bunch $\Gamma; \Delta$ is replaced by the bunch $\Gamma; \phi^{\&}_{\Delta}$, where $\phi^{\&}_{\Delta}$ is Δ with both the "," and ";" substituted by &. Now the corresponding fragment of the type theory we are working in doesn't have $\otimes$. But if we abstract the $\phi^{\otimes}_{\Delta}$ over and then curry the $\otimes$ away, all we need is $\multimap$. This is the idea at work inside the proof of soundness.

We remark that a finer translation of contexts, which can be thought of as relying less on dereliction, is possible, though we leave this to another occasion.

This context translation is used explicitly in the following.

Definition 38 *The function $\mathcal{T}$ is defined on terms, formulae and derivations of **BI** as follows.*

1. *Given a term t, the context for t, denoted by Γ_t, and the translation of t, denoted by $\mathcal{T}(t)$, are inductively defined as follows:*

t	Γ_t	$\mathcal{T}(t)$
$c{:}C$	$\langle\rangle$	$c{:}C$
$x{:}C$	$x{:}C$	$x{:}C$

If the language had functions in it, then we should extend the above translation in the obvious way;

2. *Given a formula ϕ, the context for ϕ, denoted by Γ_ϕ, and the translation of ϕ, denoted by $\mathcal{T}(\phi)$, are inductively defined as follows:*

ϕ	Γ_ϕ	$\mathcal{T}(\phi)$
$\phi \multimap \psi$	$\mathcal{T}(\Gamma_\phi), \mathcal{T}(\Gamma_\psi)$	$\mathcal{T}(\phi) \multimap \mathcal{T}(\psi)$
$\forall_{\mathrm{new}} x{:}C.\phi$	$\mathcal{T}(\Gamma_\phi) - \{x{:}C\}$	$\Lambda x{:}C.\mathcal{T}(\phi)$
$\phi \& \psi$	$\mathcal{T}(\Gamma_\phi)$	$\mathcal{T}(\phi) \& \mathcal{T}(\psi)$
$\phi \to \psi$	$\mathcal{T}(\Gamma_\phi), \mathcal{T}(\Gamma_\psi)$	$\mathcal{T}(\phi) \to \mathcal{T}(\psi)$
$\forall x{:}C.\phi$	$\mathcal{T}(\Gamma_\phi) - \{x!C\}$	$\Lambda x!C.\mathcal{T}(\phi)$

If the language had relations in it, then, again, we should extend the above translation in the obvious way. Note that in the case of the translation of $\phi \& \psi$, $\Gamma_\phi = \Gamma_\psi$, so that we could have stated either;

3. *Given a derivation* $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi$, *the context for* Π, *denoted by* Γ_Π, *and the translation of* Π, *denoted by* $\mathcal{T}(\Pi)$, *are inductively defined as follows:*

Π	Γ_Π	$\mathcal{T}(\Pi)$
$LAx_\phi(\xi)$	$\mathcal{T}(X),\xi{:}\mathcal{T}(\phi)$	$\xi{:}\mathcal{T}(\phi)$
$IAx_\phi(\xi)$	$\mathcal{T}(X),\xi!\mathcal{T}(\phi)$	$\xi{:}\mathcal{T}(\phi)$
$LIMP\text{-}I_{\phi,\psi}(\xi{:}\Pi)$	$\mathcal{T}(\Gamma_\Pi)-\{\xi{:}\mathcal{T}(\phi)\}$	$\lambda\xi{:}\mathcal{T}(\phi).\mathcal{T}(\Pi)$
$LIMP\text{-}E_{\phi,\psi}(\Pi,\Pi')$	Θ	$\mathcal{T}(\Pi)\ \mathcal{T}(\Pi')$
$LALL\text{-}I_{x,\phi}(\Pi)$	$\mathcal{T}(\Gamma_\Pi)-\{x{:}C\}$	$\lambda x{:}C.\mathcal{T}(\Pi)$
$LALL\text{-}E_{x,\phi,t}(\Pi)$	Ξ	$\mathcal{T}(\Pi)\ \mathcal{T}(t)$
$WITH\text{-}I_{\phi,\psi}(\Pi,\Pi')$	$\mathcal{T}(\Gamma_\Pi)$	$\langle\mathcal{T}(\Pi),\mathcal{T}(\Pi')\rangle$
$WITH\text{-}E^i_{\phi,\psi}(\Pi)$	$\mathcal{T}(\Gamma_\Pi)$	$\pi_i(\mathcal{T}(\Pi))$
$IIMP\text{-}I_{\phi,\psi}(\xi{:}\Pi)$	$\mathcal{T}(\Gamma_\Pi)-\{\xi!\mathcal{T}(\phi)\}$	$\lambda\xi!\mathcal{T}(\phi).\mathcal{T}(\Pi)$
$IIMP\text{-}E_{\phi,\psi}(\Pi,\Pi')$	Φ	$\mathcal{T}(\Pi)\ \mathcal{T}(\Pi')$
$IALL\text{-}I_{x,\phi}(\Pi)$	$\mathcal{T}(\Gamma_\Pi)-\{x!C\}$	$\lambda x!C.\mathcal{T}(\Pi)$
$IALL\text{-}E_{x,\phi,t}(\Pi)$	Ψ	$\mathcal{T}(\Pi)\ \mathcal{T}(t)$

where $[\Theta;\mathcal{T}(\Gamma_\Pi);\mathcal{T}(\Gamma_{\Pi'})]$ *(that is,* Θ *is the join of the translations of* Γ_Π *and* $\Gamma_{\Pi'}$*),* $[\Xi;\mathcal{T}(\Gamma_\Pi);\mathcal{T}(\Gamma_t)]$, $[\Phi;\mathcal{T}(\Gamma_\Pi);\mathcal{T}(\Gamma_{\Pi'})]$ *and* $[\Psi;\mathcal{T}(\Gamma_\Pi);\mathcal{T}(\Gamma_t)]$.

We can be more specific about the form of the context for some of the above cases.

Lemma 39 *In the intuitionistic elimination,* $\multimap$*E and* $\forall$*E, cases, the translation of the minor premiss* $\mathcal{T}(\Gamma_{\Pi'})$ *is entirely intuitionistic.*

Proof The form of the rules $\multimap$E and $\forall$E dictates that the context of the minor premiss be entirely formed with ";". The translation of such a bunch is, by definition, an entirely intuitionistic one. The effect of the structural rules is argued for on a similar basis. □

The next lemma says that the translation of terms and formulae is indeed sound. *i.e.*, that terms do correspond to elements and formulae do correspond to types.

Lemma 40

1. *If* $X \vdash_\Sigma t{:}C$, *then* $\Gamma_t \vdash_\Sigma \mathcal{T}(t){:}C$.

2. *If* $(X)\Delta \vdash_{\Sigma,\Xi} \phi{:}\mathsf{Prop}$, *then* $\Gamma_\phi \vdash_\Sigma \mathcal{T}(\phi){:}\mathsf{Type}$.

Proof : By induction on the structure of t and ϕ respectively.

1. Both the constant and variable cases are dealt with straight-forwardly by the (Mc) and $(MVar)$ rules. The inductive hypothesis is used for any functions that we might have, where we use the $(M\Lambda\mathcal{E})$ rule.

2. If there are any predicates in the language, then they are treated as base cases; we use $(A\Lambda\mathcal{E})$ and $(A\Lambda!\mathcal{E})$ to construct the types. For the inductive cases, we consider just the linear ones. For $\phi \multimap \psi$, by induction hypothesis twice, we have that $\Gamma_\phi \vdash_\Sigma \mathcal{T}(\phi){:}\mathsf{Type}$ and $\Gamma_\psi \vdash_\Sigma \mathcal{T}(\psi){:}\mathsf{Type}$. Then we just use $(A\Lambda I2)$ to get $\Gamma_\phi, \Gamma_\psi \vdash_\Sigma \mathcal{T}(\phi) \multimap \mathcal{T}(\psi){:}\mathsf{Type}$, which is the required $\mathcal{T}$-translation of $\phi \multimap \psi$. For $\forall_{\mathrm{new}} x{:}C.\phi$, by induction hypothesis, we have that $\Gamma_\phi, x{:}C \vdash_\Sigma \mathcal{T}(\phi){:}\mathsf{Type}$. Then we use $(A\Lambda I1)$ to get $\Gamma_\phi \vdash_\Sigma \Lambda x{:}C.\mathcal{T}(\phi){:}\mathsf{Type}$, which is the required $\mathcal{T}$-translation of $\forall_{\mathrm{new}} x{:}C.\phi$. The intuitionistic cases are dealt with similarly. $\square$

The next theorem relates derivations in **BI** to proofs in the $\lambda\Lambda$-calculus.

Theorem 41 (Soundness) *If* $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi$, *then* $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi)$.

Proof By induction on the structure of the derivation of the hypothesis.

$\mathrm{LAx}_\phi(\xi)$ It is convenient to work with the usual form of the axiom, which can be obtained by an application of Substitution. By Lemma 40, we know that $\Gamma_\phi \vdash_\Sigma \mathcal{T}(\phi){:}\mathsf{Type}$, where Γ_ϕ are the declarations of the free variables in ϕ. Then we use $(MVar)$ to get $\Gamma_\phi, \xi{:}\mathcal{T}(\phi) \vdash_\Sigma \xi{:}\mathcal{T}(\phi)$, which is the required $\mathcal{T}$-translation of $(X)\xi{:}\phi \vdash_{\Sigma,\Xi} \mathrm{LAx}_\phi(\xi){:}\phi$.

$\mathrm{IAx}_\phi(\xi)$ This is similar to the previous case and is omitted.

$\mathrm{LIMP\text{-}I}_{\phi,\psi}(\Pi)$ $(X)\Delta \vdash_{\Sigma,\Xi} \mathrm{LIMP\text{-}I}_{\phi,\psi}(\Pi){:}\phi \multimap \psi$ because $(X)\Delta, \xi{:}\phi \vdash_{\Sigma,\Xi} \Pi{:}\psi$. By induction hypothesis we have that $\Gamma_\Pi, \xi{:}\mathcal{T}(\phi) \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\psi)$. By an application of $(M\lambda\Lambda I)$ we get $\Gamma_\Pi \vdash_\Sigma \lambda\xi{:}\mathcal{T}(\phi).\mathcal{T}(\Pi) : \mathcal{T}(\phi) \multimap \mathcal{T}(\psi)$, which is the required conclusion.

$\mathrm{LIMP\text{-}E}_{\phi,\psi}(\Pi,\Pi')$ $(X,X')\Delta,\Delta' \vdash_{\Sigma,\Xi} \mathrm{LIMP\text{-}E}_{\phi,\psi}(\Pi,\Pi'){:}\psi$ because $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi \multimap \psi$ and $(X')\Delta' \vdash_{\Sigma,\Xi} \Pi'{:}\phi$. By induction hypothesis twice we have that $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi) \multimap \mathcal{T}(\psi)$ and $\Gamma_{\Pi'} \vdash_\Sigma \mathcal{T}(\Pi'){:}\mathcal{T}(\phi)$. By an application of $(M\Lambda\mathcal{E})$ and (possibly) weakening, we get

$$\Xi \vdash_\Sigma \mathcal{T}(\Pi)\mathcal{T}(\Pi'){:}\psi$$

where $[\Xi; \Gamma_\Pi; \Gamma_{\Pi'}]$. This is the required conclusion.

LALL-I$_{x,\phi}(\Pi)$ $(X)\Delta \vdash_{\Sigma,\Xi}$ LALL-I$_{x,\phi}(\Pi)$:$\forall_{\text{new}} x{:}C.\phi$ because $(X,x{:}C)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi$. By induction hypothesis we have that $\Gamma_\Pi, x{:}C \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi)$. By an application of $(M\lambda\Lambda I)$ we get $\Gamma_\Pi \vdash_\Sigma \lambda x{:}C.\mathcal{T}(\Pi) : \Lambda x{:}C.\mathcal{T}(\phi)$, which is the required conclusion.

LALL-E$_{x,\phi,t}(\Pi)$ $(X,X')\Delta \vdash_{\Sigma,\Xi}$ LALL-E$_{x,\phi,t}(\Pi)$:$\phi[t/x]$ because $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\forall_{\text{new}} x{:}C.\phi$ and $(X') \vdash_{\Sigma,\Xi} t{:}A$. By induction hypothesis twice we have that $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\Lambda x{:}A.\mathcal{T}(\phi)$ and that $\Gamma_{X'} \vdash_\Sigma \mathcal{T}(t){:}C$. Then we use $(M\Lambda\mathcal{E})$ and (possibly) weakening, to get $\Theta \vdash_\Sigma \mathcal{T}(\Pi)\mathcal{T}(t){:}\mathcal{T}(\phi)[t/x]$, where $[\Theta; \Gamma_\Pi; \Gamma_X]$. This is the required conclusion.

WITH-I$_{\phi,\psi}(\Pi,\Pi')$ $(X)\Delta \vdash_{\Sigma,\Xi}$ WITH-I$_{\phi,\psi}(\Pi,\Pi')$:$\phi\&\psi$ because $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi$ and $(X)\Delta \vdash_{\Sigma,\Xi} \Pi'{:}\psi$. By induction hypothesis twice we have that $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi)$ and $\Gamma_{\Pi'} \vdash_\Sigma \mathcal{T}(\Pi'){:}\mathcal{T}(\psi)$. Recall that $\Gamma_\Pi = \Gamma_{\Pi'}$. Then we use $(M\&I)$ to get $\Gamma_\Pi \vdash_\Sigma \langle \mathcal{T}(\Pi), \mathcal{T}(\Pi')\rangle{:}\mathcal{T}(\phi)\&\mathcal{T}(\psi)$, which is the required conclusion.

WITH-E$^i_{\phi_0,\phi_1}(\Pi)$ $(X)\Delta \vdash_{\Sigma,\Xi}$ WITH-E$^i_{\phi_0,\phi_1}(\Pi)$:ϕ_i because $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi_0\&\phi_1$. By induction hypothesis we have that $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi_0)\&\mathcal{T}(\phi_1)$. Then we use $(M\&\mathcal{E}_i)$ to get

$$\Gamma_\Pi \vdash_\Sigma \pi_i(\mathcal{T}(\Pi)){:}\mathcal{T}(\phi_i)$$

which is the required conclusion.

IIMP-I$_{\phi,\psi}(\Pi)$ $(X)\Delta \vdash_{\Sigma,\Xi}$ IIMP-I$_{\phi,\psi}(\Pi)$:$\phi \to \psi$ because $(X)\Delta;\xi{:}\phi \vdash_{\Sigma,\Xi} \Pi{:}\psi$. By induction hypothesis we have that $\Gamma_\Pi, \xi!\mathcal{T}(\phi) \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\psi)$. By an application of $(M\lambda\Lambda!I)$ we get $\Gamma_\Pi \vdash_\Sigma \lambda\xi!\mathcal{T}(\phi).\mathcal{T}(\Pi) : \mathcal{T}(\phi) \to \mathcal{T}(\psi)$, which is the required conclusion.

IIMP-E$_{\phi,\psi}(\Pi,\Pi')$ $(X;!X')\Delta;!\Delta' \vdash_{\Sigma,\Xi}$ IIMP-E$_{\phi,\psi}(\Pi,\Pi')$:ψ because $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi \to \psi$ and $(!X')!\Delta' \vdash_{\Sigma,\Xi} \Pi'{:}\phi$. By induction hypothesis twice we have that $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi) \to \mathcal{T}(\psi)$ and $!\Gamma_{\Pi'} \vdash_\Sigma \mathcal{T}(\Pi'){:}\mathcal{T}(\phi)$. We weaken the first of these judgements to get $\Gamma_\Pi, !\Gamma_{\Pi'} \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi) \to \mathcal{T}(\psi)$. Then, an application of $(M\Lambda!\mathcal{E})$ gives us $\Phi \vdash_\Sigma \mathcal{T}(\Pi)\mathcal{T}(\Pi'){:}\psi$, where $[\Phi; \Gamma_\Pi, !\Gamma_{\Pi'}; !\Gamma_{\Pi'}]$. This is the required conclusion.

IALL-I$_{x,\phi}(\Pi)$ $(X)\Delta \vdash_{\Sigma,\Xi}$ LALL-I$_{x,\phi}(\Pi)$:$\forall x{:}C.\phi$ because $(X;x{:}C)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi$. By induction hypothesis we have that $\Gamma_\Pi, x!C \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi)$. By an application of $(M\lambda\Lambda!I)$ we get $\Gamma_\Pi \vdash_\Sigma \lambda x!C.\mathcal{T}(\Pi) : \Lambda x!C.\mathcal{T}(\phi)$, which is the required conclusion.

IALL-E$_{x,\phi,t}(\Pi)$ $(X;!X')\Delta \vdash_{\Sigma,\Xi}$ LALL-E$_{x,\phi,t}(\Pi)$:$\phi[t/x]$ because $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\forall x{:}C.\phi$ and $(!X') \vdash_{\Sigma,\Xi} t{:}C$. By induction hypothesis twice we have that $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\Lambda x!A.\mathcal{T}(\phi)$ and that $!\Gamma_{X'} \vdash_\Sigma$

$\mathcal{T}(t){:}C$. We weaken the first of these judgements to get $\Gamma_\Pi, !\Gamma_{!X'} \vdash_\Sigma \mathcal{T}(\Pi){:}\Lambda x!A\,.\mathcal{T}(\phi)$. Then, application of $(M\Lambda!\mathcal{E})$ gives us $\Psi \vdash_\Sigma \mathcal{T}(\Pi)\mathcal{T}(t){:}\mathcal{T}(\phi)[t/x]$, where $[\Psi;\Gamma_\Phi, !\Gamma_{!X'}; !\Gamma_{!X'}]$. This is the required conclusion. □

We now give a translation from the type theory to the logic. Basically, the above technique is done in reverse. That is, we consider the signature of the $\lambda\Lambda$-calculus and construct the structure and language out of it. Then we give mappings of elements, types and elements of types of the $\lambda\Lambda$-calculus to terms, formulae and deductions of **BI**.

We now define the translation from the type theory to the logic. The $\lambda\Lambda$-calculus' contexts are translated as follows: $\langle\rangle$ is translated to $I;1$; the singleton contexts $x{:}A$ and $x!A$ are translated to $I,x{:}A$ and $1;x{:}A$; and other contexts $\Gamma,x{:}A$ and $\Gamma,x!A$ are translated as $\Gamma',x{:}A$ and $\Gamma';x{:}A$, where Γ' is the translation, inductively, of Γ.

Definition 42 *The function $\mathcal{L}$ is defined on elements, types and terms of the $\lambda\Lambda$-calculus as follows.*

1. *Given an element t, the assumptions for t, denoted by (X_t), and the translation of t, denoted by $\mathcal{L}(t)$, is inductively defined as follows:*

t	(X_t)	$\mathcal{L}(t)$
$c{:}A$	$\langle\rangle$	$c{:}A$
$x{:}A$	$x{:}A$	$x{:}A$

If the signature contained functions, then we extend the above translation in the obvious way;

2. *Given a type ϕ, the assumptions for ϕ, denoted by $(X_\phi)\Delta_\phi$, and the translation of ϕ, denoted by $\mathcal{L}(\phi)$, are inductively defined as follows:*

ϕ	$(X_\phi)\Delta_\phi$	$\mathcal{L}(\phi)$
a	$\langle\rangle$	a
$\phi \multimap \psi$	$(X_\phi, X_\psi)\Delta_\phi, \Delta_\psi$	$\mathcal{L}(\phi) \multimap \mathcal{L}(\psi)$
$\Lambda x{:}A\,.\phi$	$(X_\phi - (I,x{:}A)\Delta_\phi$	$\forall_{\text{new}} x{:}A\,.\mathcal{L}(\phi)$
$\phi\&\psi$	$(X_\phi)\Delta_\phi$	$\mathcal{L}(\phi)\&\mathcal{L}(\psi)$
$\phi \to \psi$	$(X_\phi;X_\psi)\Delta_\phi;\Delta_\psi$	$\mathcal{L}(\phi) \to \mathcal{L}(\psi)$
$\Lambda x!A\,.\phi$	$(X_\phi - (1;x{:}A)\Delta_\phi$	$\forall x{:}A\,.\mathcal{L}(\phi)$

We note that in the case of the additive conjunction, $\Delta_\phi = \Delta_\psi$, so we could have stated either. We also note that if the language had relations in it, then, as in the translation for terms, the above translation would be extended in the obvious way;

3. *Given a derivation $\Gamma \vdash_\Sigma M{:}A$, the proof context for M, denoted by $(X_M)\Delta_M$, and the translation of M, denoted by $\mathcal{L}(M)$, are inductively defined as follows:*

$\Gamma \vdash_\Sigma M{:}A$	$(X_M)\Delta_M$	$\mathcal{L}(M)$
$x{:}A$	$(X_\Gamma)I, x{:}A$	$\mathit{LAx}_{\mathcal{L}(A)}(x)$
$x{:}A$	$(X_\Gamma)1; x{:}A$	$\mathit{IAx}_{\mathcal{L}(A)}(x)$
$\lambda x{:}A.M : A \multimap B$	$(X_M)\Delta_M - (I, x{:}\mathcal{L}(A))$	$\mathit{LIMP\text{-}I}_{\mathcal{L}(A),\mathcal{L}(B)}(x{:}\mathcal{L}(M))$
$MN{:}B$	$(X_M, X_N)\Delta_M, \Delta_N$	$\mathit{LIMP\text{-}E}_{\mathcal{L}(A),\mathcal{L}(B)}(\mathcal{L}(M), \mathcal{L}(N))$
$\lambda x{:}C.M{:}\Lambda x{:}C.A$	$(X_M - (I, x{:}C))\Delta_M$	$\mathit{LALL\text{-}I}_{x,\mathcal{L}(A)}(\mathcal{L}(M))$
$Mt{:}A[t/x]$	$(X_M, X_t)\Delta_M$	$\mathit{LALL\text{-}E}_{x,\mathcal{L}(A),t}(\mathcal{L}(M))$
$\langle M, N\rangle{:}A\&B$	$(X_M)\Delta_M$	$\mathit{WITH\text{-}I}_{\mathcal{L}(A),\mathcal{L}(B)}(\mathcal{L}(M), \mathcal{L}(N))$
$\pi_i(M){:}A_i$	$(X_M)\Delta_M$	$\mathit{WITH\text{-}E}^i_{\mathcal{L}(A_0),\mathcal{L}(A_1)}(\mathcal{L}(M))$
$\lambda x!A.M : A \to B$	$(X_M)\Delta_M - (1; x{:}\mathcal{L}(A))$	$\mathit{IIMP\text{-}I}_{\mathcal{L}(A),\mathcal{L}(B)}(x{:}\mathcal{L}(M))$
$MN{:}B$	$(X_M; X_N)\Delta_M; \Delta_N$	$\mathit{IIMP\text{-}E}_{\mathcal{L}(A),\mathcal{L}(B)}(\mathcal{L}(M), \mathcal{L}(N))$
$\lambda x!C.M{:}\Lambda x!C.A$	$(X_M - (1; x{:}C))\Delta_M$	$\mathit{LALL\text{-}I}_{x,\mathcal{L}(A)}(\mathcal{L}(M))$
$Mt{:}A[t/x]$	$(X_M; X_t)\Delta_M$	$\mathit{IALL\text{-}E}_{x,\mathcal{L}(A),\mathcal{L}(t)}(\mathcal{L}(M))$

We note that in the translation of identities and applications, we really need realizers to indicate what kind of identity or application was realized. Usually this will be clear from the context, so we shall not be too formal about it.

The following lemma says that elements do indeed become terms and types do indeed become formulae. As with all completeness arguments, it requires that the type-theoretic judgement resemble the logic judgement in that only types can depend on terms, and not the other way around.

Lemma 43

1. *If $\Gamma \vdash_\Sigma t{:}C$ and Γ is just a list of variable declarations, then $\mathcal{L}(\Gamma) \vdash_\Sigma \mathcal{L}(t){:}C$.*

2. *If $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$ and Γ is of the form Γ_X, Γ_Δ (a list of variable declarations followed by a list of type declarations), then $(\mathcal{L}(\Gamma_X))\mathcal{L}(\Gamma_\Delta) \vdash_{\Sigma,\Xi} \mathcal{L}(A){:}\mathsf{Prop}$.*

Proof By induction on the structure of t and A respectively.

1. The base cases are when t is a constant or a variable. In both these cases, $\mathcal{L}(t) = t \in Term_C$ by virtue of C being declared in the signature and so being a sort symbol in the structure. The inductive hypothesis and the application rules are used for any functions that might be declared in the signature.

2. The base case is when A is a constant type, a. In this case, a is translated to a sort a in the signature. We consider just one inductive case. Suppose $\Gamma \vdash_\Sigma \Lambda x{:}C.\phi : \mathsf{Type}$ because $\Gamma, x{:}C \vdash_\Sigma \phi{:}\mathsf{Type}$. By induction hypothesis, we have that $(\mathcal{L}(\Gamma_X), x{:}C)\mathcal{L}(\Gamma_\Delta) \vdash_{\Sigma,\Xi} \mathcal{L}(\phi){:}\mathsf{Prop}$. Then we use $(\forall_{\text{new}} I)$ to get that $\mathcal{L}(\Gamma) \vdash_{\Sigma,\Xi} \forall_{\text{new}} x{:}C.\mathcal{L}(\phi) : \mathsf{Prop}$. The other inductive cases are done similarly. □

The following theorem relates realizers of the λΛ-calculus to proof expressions of **BI**.

Theorem 44 (Completeness) *If $\Gamma \vdash_\Sigma M{:}A$ and Γ is of the form Γ_X, Γ_Δ (a list of variable declarations followed by a list of type declarations), then $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(A)$.*

Proof By induction on the proof of the hypothesis.

$(MVar)$ Suppose $\Gamma, x{:}A \vdash_\Sigma x{:}A$, which implies that $dom(\Gamma) = FV(A)$. By Lemma 43, we know that $\mathcal{L}(A) \in Form_A$. Then we use $(MultId)$ to construct $(X_\Gamma)x{:}\mathcal{L}(A) \vdash_{\Sigma,\Xi} \textsc{LAx}_{\mathcal{L}(A)}(x){:}\mathcal{L}(A)$, where X_Γ is the **BI** version of the λΛ-calculus context Γ.

$(MVar!)$ This case is done similarly to the one above and we omit the details.

$(M\lambda\Lambda I)$ Suppose $\Gamma \vdash_\Sigma \lambda x{:}A.M{:}A \multimap B$ because $\Gamma, x{:}A \vdash_\Sigma M{:}B$. By induction hypothesis we have that $(X_M)\Delta_M, x{:}\mathcal{L}(A) \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(B)$. Then we use $(\multimap I)$ to get $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \textsc{limp-i}_{A,B}(\mathcal{L}(M)){:}\mathcal{L}(A) \multimap \mathcal{L}(B)$, which is the required conclusion.

$(M\Lambda\mathcal{E})$ Suppose $\Xi \vdash_\Sigma MN{:}B$ because $\Gamma \vdash_\Sigma M{:}A \multimap B$ and $\Delta \vdash_\Sigma N{:}A$, where Ξ is the context-sensitive join of Γ and Δ. By induction hypothesis twice we have that $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(A) \multimap \mathcal{L}(B)$ and $(X_N)\Delta_N \vdash_{\Sigma,\Xi} \mathcal{L}(N){:}\mathcal{L}(A)$. Then we use $(\multimap E)$ to get

$$(X_M, X_N)\Delta_M, \Delta_N \vdash_{\Sigma,\Xi} \textsc{limp-e}_{A,B}(\mathcal{L}(M), \mathcal{L}(N)){:}\mathcal{L}(B)$$

which is the required conclusion.

($M\lambda\Lambda I$) Suppose $\Gamma \vdash_\Sigma \lambda x{:}C.M{:}\Lambda x{:}C.A$ because $\Gamma, x{:}C \vdash_\Sigma M{:}A$. By induction hypothesis we have that $(X_M, x{:}C)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(A)$. Then we use ($\forall_{\text{new}}I$) to get

$$(X_M)\Delta_M \vdash_{\Sigma,\Xi} \text{LALL-I}_{x,A}(\mathcal{L}(M)){:}\forall_{\text{new}}x{:}C.\mathcal{L}(A)$$

which is the required conclusion.

($M\Lambda\mathcal{E}$) Suppose $\Xi \vdash_\Sigma Mt{:}A[t/x]$ because $\Gamma \vdash_\Sigma M{:}\forall_{\text{new}}x{:}C.A$ and $\Delta \vdash_\Sigma t{:}C$, where Ξ is the context-sensitive join of Γ and Δ. By induction hypothesis twice we have that $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\forall_{\text{new}}x{:}C.\mathcal{L}(A)$ and that $(X_t) \vdash_{\Sigma,\Xi} \mathcal{L}(t){:}C$. Then we use ($\forall_{\text{new}}E$) to get $(X_M, X_t)\Delta_M \vdash_{\Sigma,\Xi}$ $\text{LIMP-E}_{x,\phi}(\mathcal{L}(t), \mathcal{L}(M)){:}\mathcal{L}(\phi)[\mathcal{L}(t)/x]$, which is the required conclusion, again up to structural equivalence.

($M\&I$) Suppose $\Gamma \vdash_\Sigma \langle M, N\rangle{:}A\&B$ because $\Gamma \vdash_\Sigma M{:}A$ and $\Gamma \vdash_\Sigma N{:}B$. By induction hypothesis twice we have that $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(A)$ and $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(N){:}\mathcal{L}(B)$. Then we use ($\&I$) to get $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \text{WITH-I}_{\mathcal{L}(A),\mathcal{L}(B)}(\mathcal{L}(M), \mathcal{L}(N)) : \mathcal{L}(A)\&\mathcal{L}(B)$, which is the required conclusion.

($M\&\mathcal{E}_i$) Suppose $\Gamma \vdash_\Sigma \pi_i(M){:}A_i$ because $\Gamma \vdash_\Sigma M{:}A_0\&A_1$. By induction hypothesis we have that $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(A_0)\&\mathcal{L}(A_1)$. Then we use ($\&E$) to get

$$(X_M)\Delta_M \vdash_{\Sigma,\Xi} \text{WITH-E}^i_{\mathcal{L}(A_0),\mathcal{L}(A_1)}(\mathcal{L}(M)){:}\mathcal{L}(A_i)$$

which is the required conclusion.

($M\lambda\Lambda!I$) Suppose $\Gamma \vdash_\Sigma \lambda x!A.M{:}A \to B$ because $\Gamma, x!A \vdash_\Sigma M{:}B$. By induction hypothesis we have that $(X_M)\Delta_M; x{:}\mathcal{L}(A) \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(B)$. Then we use ($\to I$) to get $(X_M)\Delta_M \vdash_{\Sigma,\Xi}$ $\text{IIMP-I}_{A,B}(\mathcal{L}(M)){:}\mathcal{L}(A) \to \mathcal{L}(B)$, which is the required conclusion.

($M\Lambda!\mathcal{E}$) Suppose $\Xi \vdash_\Sigma MN{:}B$ because $\Gamma \vdash_\Sigma M{:}A \to B$ and $!\Delta \vdash_\Sigma N{:}A$, where Ξ is the context-sensitive join of Γ and $!\Delta$. By induction hypothesis twice we have that $(X_M)\Delta_M \vdash_{\Sigma,\Xi}$ $\mathcal{L}(M){:}\mathcal{L}(A) \to \mathcal{L}(B)$ and $(!X_N)!\Delta_N \vdash_{\Sigma,\Xi} \mathcal{L}(N){:}\mathcal{L}(A)$. Then we use ($\to E$) to get

$$(X_M; !X_N)\Delta_M; !\Delta_N \vdash_{\Sigma,\Xi} \text{IIMP-E}_{A,B}(\mathcal{L}(M), \mathcal{L}(N)){:}\mathcal{L}(B)$$

which is the required conclusion.

($M\lambda\Lambda!I$) Suppose $\Gamma \vdash_\Sigma \lambda x!C.M{:}\Lambda x!C.A$ because $\Gamma, x!C \vdash_\Sigma M{:}A$. By induction hypothesis we have that $(X_M;x{:}C)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(A)$. Then we use ($\forall I$) to get

$$(X_M)\Delta_M \vdash_{\Sigma,\Xi} \text{IALL-I}_{x,A}(\mathcal{L}(M)){:}\forall x{:}C.\mathcal{L}(A)$$

which is the required conclusion.

($M\Lambda!\mathcal{E}$) Suppose $\Xi \vdash_\Sigma Mt{:}A[t/x]$ because $\Gamma \vdash_\Sigma M{:}\forall x{:}C.A$ and $!\Delta \vdash_\Sigma t{:}C$, where Ξ is the context-sensitive join of Γ and $!\Delta$. By induction hypothesis twice we have that $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\forall x{:}C.\mathcal{L}(A)$ and that $(!X_t) \vdash_{\Sigma,\Xi} \mathcal{L}(t){:}C$. Then we use ($\forall E$) to get

$$(X_M, !X_t)\Delta_M \vdash_{\Sigma,\Xi} \text{IIMP-E}_{x,\phi}(t, \mathcal{L}(M)){:}\mathcal{L}(\phi)[\mathcal{L}(t)/x]$$

which is the required conclusion. □

So far we have set up maps between the logic and the type theory. The next theorem shows that the maps are inverses of each other.

Theorem 45 (Correspondence) *Let* $(X)\Delta \vdash_{\Sigma,\Xi} \Pi{:}\phi$ *and let* $\Gamma \vdash_\Sigma M{:}A$. *Then*

1. $\mathcal{L}(\mathcal{T}(\Pi)) = \Pi$ *; and*
2. $\mathcal{T}(\mathcal{L}(M)) = M$.

Proof

1. By induction on the structure of Π.

$\text{LAx}_\phi(\xi)$ Suppose $(X)\xi{:}\phi \vdash_{\Sigma,\Xi} \text{LAx}_\phi(\xi){:}\phi$. Then we calculate

$$\begin{array}{rcll} \mathcal{L}(\mathcal{T}(\text{LAx}_\phi(\xi))) & = & \mathcal{L}(\xi) & \text{with } \Gamma = \mathcal{T}(X), \xi{:}\phi \\ & = & \text{LAx}_\phi(\xi) & \text{definition} \end{array}$$

$\text{IAx}_\phi(\xi)$ This is dealt with similarly to the above one and we omit the details.

$\text{LIMP-I}_{\phi,\psi}(\xi{:}\Pi)$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{LIMP-I}_{\phi,\psi}(\xi{:}\Pi){:}\phi \multimap \psi$. Then we calculate

$$\begin{array}{lll} \mathcal{L}(\mathcal{T}(\text{LIMP-I}_{\phi,\psi}(\xi{:}\Pi))) & & \\ \quad = & \mathcal{L}(\lambda\xi{:}\mathcal{T}(\phi).\mathcal{T}(\Pi)) & \text{with } \Gamma = \mathcal{T}(\Gamma_\Pi) - \{\xi{:}\phi\} \\ \quad = & \text{LIMP-I}_{\mathcal{L}(\mathcal{T}(\phi)),\mathcal{L}(\mathcal{T}(\psi))}(\xi{:}\mathcal{L}(\mathcal{T}(\Pi))) & \text{definition} \\ \quad = & \text{LIMP-I}_{\phi,\psi}(\xi{:}\Pi) & \text{IH} \end{array}$$

$\text{LIMP-E}_{\phi,\psi}(\Pi,\Pi')$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{LIMP-E}_{\phi,\psi}(\Pi,\Pi'){:}\psi$. Before giving the calculation, we make a brief note regarding the translation of contexts. The premisses of the judgement $(X)\Delta \vdash_{\Sigma,\Xi} \text{LIMP-E}_{\phi,\psi}(\Pi,\Pi'){:}\psi$ are $(X_\Pi)\Delta_\Pi \vdash_{\Sigma,\Xi} \Pi{:}\phi \multimap \psi$ and $(X_{\Pi'})\Delta_{\Pi'} \vdash_{\Sigma,\Xi} \Pi'{:}\phi$, where $X = X_\Pi, X_{\Pi'}$ and $\Delta = \Delta_\Pi, \Delta_{\Pi'}$. By induction, these premisses will be translated to the $\lambda\Lambda$-calculus judgements $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi) \multimap \mathcal{T}(\psi)$ and $\Gamma_{\Pi'} \vdash_\Sigma \mathcal{T}(\Pi'){:}\mathcal{T}(\phi)$. Now, the contexts Γ_Π and $\Gamma_{\Pi'}$ might not share the same intuitionistic variables (which arise from the ; variables of the **BI** context). So, in order for the type theory join to work, it might be necessary to weaken in the appropriate intuitionistic variables in each of the contexts Γ_Π and $\Gamma_{\Pi'}$. The join then contracts the copy away and leaves a context which is the correct **BI** translation. Subject to this explanation, which applies in several other cases of this proof too, we now have the following:

$$\begin{array}{lll}
\mathcal{L}(\mathcal{T}(\text{LIMP-E}_{\phi,\psi}(\Pi,\Pi'))) & & \\
\quad = & \mathcal{L}(\mathcal{T}(\Pi)\mathcal{T}(\Pi')) & \text{with } [\Gamma;\mathcal{T}(\Gamma_\Pi);\mathcal{T}(\Gamma_{\Pi'})] \\
\quad = & \text{LIMP-E}_{\phi,\psi}(\mathcal{L}(\mathcal{T}(\Pi)),\mathcal{L}(\mathcal{T}(\Pi'))) & \text{definition} \\
\quad = & \text{LIMP-E}_{\phi,\psi}(\Pi,\Pi') & \text{IH}
\end{array}$$

$\text{LALL-I}_{x,\phi}(\Pi)$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{LALL-I}_{x,\phi}(\Pi){:}\forall_{\text{new}} x{:}C.\phi$. Then we calculate

$$\begin{array}{lll}
\mathcal{L}(\mathcal{T}(\text{LALL-I}_{x,\phi}(\Pi))) & & \\
\quad = & \mathcal{L}(\lambda x{:}C.\mathcal{T}(\Pi)) & \text{with } \Gamma = \mathcal{T}(\Gamma_\Pi) - \{x{:}C\} \\
\quad = & \text{LALL-I}_{x,\mathcal{L}(\mathcal{T}(\phi))}(\mathcal{L}(\mathcal{T}(\Pi))) & \text{definition} \\
\quad = & \text{LALL-I}_{x,\phi}(\Pi) & \text{IH}
\end{array}$$

$\text{LALL-E}_{x,\phi,t}(\Pi)$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{LALL-E}_{x,\phi,t}(\Pi){:}\phi[t/x]$. Then we calculate

$$\begin{array}{lll}
\mathcal{L}(\mathcal{T}(\text{LALL-E}_{x,\phi,t}(\Pi))) & & \\
\quad = & \mathcal{L}(\mathcal{T}(\Pi)\mathcal{T}(t)) & \text{with } [\Gamma;\mathcal{T}(\Gamma_\Pi);\mathcal{T}(\Gamma_t)] \\
\quad = & \text{LALL-E}_{x,\mathcal{L}(\mathcal{T}(\phi)),\mathcal{L}(\mathcal{T}(t))}(\mathcal{L}(\mathcal{T}(\Pi))) & \text{definition} \\
\quad = & \text{LALL-E}_{x,\phi,t}(\Pi) & \text{IH}
\end{array}$$

The above calculation is done subject to the same explanation regarding the $(\multimap E)$ calculation above.

$\text{WITH-I}_{\phi,\psi}(\Pi,\Pi')$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{WITH-I}_{\phi,\psi}(\Pi,\Pi'){:}\phi\&\psi$. Then we calculate

$$\begin{array}{lll} \mathcal{L}(\mathcal{T}(\text{WITH-I}_{\phi,\psi}(\Pi,\Pi'))) & & \\ \quad = \mathcal{L}(\langle\mathcal{T}(\Pi),\mathcal{T}(\Pi')\rangle) & & \text{with } \Gamma = \mathcal{T}(\Gamma_\Pi)(= \mathcal{T}(\Gamma_{\Pi'})) \\ \quad = \text{WITH-I}_{\phi,\psi}(\mathcal{L}(\mathcal{T}(\Pi)),\mathcal{L}(\mathcal{T}(\Pi'))) & & \text{IH} \end{array}$$

WITH-E$^i_{\phi_0,\phi_1}(\Pi)$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{WITH-E}^i_{\phi_0,\phi_1}(\Pi){:}\phi_i$. Then we calculate

$$\begin{array}{lll} \mathcal{L}(\mathcal{T}(\text{WITH-E}^i_{\phi_0,\phi_1}(\Pi))) & & \\ \quad = \mathcal{L}(\pi_i(\mathcal{T}(\Pi))) & & \text{with } \Gamma = \mathcal{T}(\Gamma_\Pi) \\ \quad = \text{WITH-E}^i_{\phi_0,\phi_1}(\mathcal{L}(\mathcal{T}(\Pi))) & & \text{IH} \end{array}$$

IIMP-I$_{\phi,\psi}(\xi{:}\Pi)$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{IIMP-I}_{\phi,\psi}(\xi{:}\Pi){:}\phi \to \psi$. Then we calculate

$$\begin{array}{lll} \mathcal{L}(\mathcal{T}(\text{IIMP-I}_{\phi,\psi}(\xi{:}\Pi))) & & \\ \quad = \mathcal{L}(\lambda\xi!\mathcal{T}(\phi).\mathcal{T}(\Pi)) & & \text{with } \Gamma = \mathcal{T}(\Gamma_\Pi) - \{\xi!\phi\} \\ \quad = \text{IIMP-I}_{\mathcal{L}(\mathcal{T}(\phi)),\mathcal{L}(\mathcal{T}(\psi))}(\xi{:}\mathcal{L}(\mathcal{T}(\Pi))) & & \text{definition} \\ \quad = \text{IIMP-I}_{\phi,\psi}(\xi{:}\Pi) & & \text{IH} \end{array}$$

IIMP-E$_{\phi,\psi}(\Pi,\Pi')$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{IIMP-E}_{\phi,\psi}(\Pi,\Pi'){:}\psi$. Before giving the calculation, we make a brief note regarding the translation of contexts. The premisses of the judgement $(X)\Delta \vdash_{\Sigma,\Xi} \text{IIMP-E}_{\phi,\psi}(\Pi,\Pi'){:}\psi$ are $(X_\Pi)\Delta_\Pi \vdash_{\Sigma,\Xi} \Pi{:}\phi \to \psi$ and $(!X_{\Pi'})!\Delta_{\Pi'} \vdash_{\Sigma,\Xi} \Pi'{:}\phi$, where $X = X_\Pi; !X_{\Pi'}$ and $\Delta = \Delta_{Pi}; !\Delta_{\Pi'}$. By induction, these premisses will be translated to the $\lambda\Lambda$-calculus judgements $\Gamma_\Pi \vdash_\Sigma \mathcal{T}(\Pi){:}\mathcal{T}(\phi) \to \mathcal{T}(\psi)$ and $!\Gamma_{\Pi'} \vdash_\Sigma \mathcal{T}(\Pi'){:}\mathcal{T}(\phi)$. Now, the contexts Γ_Π and $!\Gamma_{\Pi'}$ might not share the same intuitionistic variables (which arise from the ; variables of the **BI** context). So, in order for the type theory join to work, it might be necessary to weaken in the appropriate intuitionistic variables in each of the contexts Γ_Π and $!\Gamma_{\Pi'}$. The join then contracts the copy away and leaves a context which is the correct **BI** translation. Subject to this explanation, we now have the following:

$$\begin{array}{lll} \mathcal{L}(\mathcal{T}(\text{IIMP-E}_{\phi,\psi}(\Pi,\Pi'))) & & \\ \quad = \mathcal{L}(\mathcal{T}(\Pi)\mathcal{T}(\Pi')) & & \text{with } [\Gamma;\mathcal{T}(\Gamma_\Pi);\mathcal{T}(\Gamma_{\Pi'})] \\ \quad = \text{LIMP-E}_{\phi,\psi}(\mathcal{L}(\mathcal{T}(\Pi)),\mathcal{L}(\mathcal{T}(\Pi'))) & & \text{definition} \\ \quad = \text{LIMP-E}_{\phi,\psi}(\Pi,\Pi') & & \text{IH} \end{array}$$

$\text{IALL-I}_{x,\phi}(\Pi)$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{IALL-I}_{x,\phi}(\Pi){:}\forall x{:}C.\phi$. Then we calculate

$$\begin{array}{lll} \mathcal{L}(\mathcal{T}(\text{IALL-I}_{x,\phi}(\Pi))) & & \\ = & \mathcal{L}(\lambda x!C.\mathcal{T}(\Pi)) & \text{with } \Gamma = \mathcal{T}(\Gamma_\Pi) - \{x!C\} \\ = & \text{IALL-I}_{x,\mathcal{L}(\mathcal{T}(\phi))}(\mathcal{L}(\mathcal{T}(\Pi))) & \text{definition} \\ = & \text{IALL-I}_{x,\phi}(\Pi) & \text{IH} \end{array}$$

$\text{IALL-E}_{x,\phi,t}(\Pi)$ Suppose $(X)\Delta \vdash_{\Sigma,\Xi} \text{IALL-E}_{x,\phi,t}(\Pi){:}\phi[t/x]$. Then we calculate

$$\begin{array}{lll} \mathcal{L}(\mathcal{T}(\text{IALL-E}_{x,\phi,t}(\Pi))) & & \\ = & \mathcal{L}(\mathcal{T}(\Pi)\mathcal{T}(t)) & \text{with } [\Gamma;\mathcal{T}(\Gamma_\Pi);\mathcal{T}(\Gamma_t)] \\ = & \text{IALL-E}_{x,\mathcal{L}(\mathcal{T}(\phi)),\mathcal{L}(\mathcal{T}(t))}(\mathcal{L}(\mathcal{T}(\Pi))) & \text{definition} \\ = & \text{IALL-E}_{x,\phi,t}(\Pi) & \text{IH} \end{array}$$

The above calculation is done subject to the same explanation regarding the $(\to E)$ calculation above.

2. By induction on the structure of M, where $\Gamma \vdash_\Sigma M{:}A$.

$(MVar)$ Suppose $\Gamma, x{:}A \vdash_\Sigma x{:}A$. Then we calculate

$$\begin{array}{rcl} \mathcal{T}(\mathcal{L}(x)) & = & \mathcal{T}(\text{LAX}_\phi(x)) \\ & = & x \end{array}$$

$(MVar!)$ This is done similarly to the above and we omit the details.

$(M\lambda\Lambda I)$ Suppose $\Gamma \vdash_\Sigma \lambda x{:}\phi.M{:}\phi \multimap \psi$. Then we calculate

$$\begin{array}{lll} \mathcal{T}(\mathcal{L}(\lambda x{:}\phi.M)) & & \\ = & \mathcal{T}(\text{LIMP-I}_{\mathcal{L}(\phi),\mathcal{L}(\psi)}(x{:}\mathcal{L}(M))) & \text{with } (X)\Delta = (X_M)\Delta_M - (I, x{:}\mathcal{L}(\phi)) \\ = & \lambda x{:}\mathcal{T}(\mathcal{L}(\phi)).\mathcal{T}(\mathcal{L}(M)) & \text{definition} \\ = & \lambda x{:}\phi.M & \text{IH} \end{array}$$

In fact, the $x{:}\mathcal{L}(\phi)$ will be the last leaf of the bunch $(X_M)\Delta_M$.

$(M\Lambda\mathcal{E})$ Suppose $\Xi \vdash_\Sigma MN{:}\psi$. Before we give the calculation, it is worth making a little note regarding the correctness of the translation of the context. Now the judgement $\Xi \vdash_\Sigma MN{:}\psi$ holds because $\Gamma \vdash_\Sigma M{:}\phi \multimap \psi$ and $\Delta \vdash_\Sigma N{:}\phi$, where Ξ is the sharing-sensitive join of Γ and Δ. By induction, these premiss judgements are translated as

$(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(\phi) \multimap \mathcal{L}(\psi)$ and $(X_N)\Delta_N \vdash_{\Sigma,\Xi} \mathcal{L}(N){:}\mathcal{L}(\phi)$. An application of $(\multimap E)$ then gives us the judgement $(X_M,X_N)\Delta_M,\Delta_N \vdash_{\Sigma,\Xi} \mathcal{L}(M)\mathcal{L}(N){:}\mathcal{L}(\psi)$. Now the contexts $(X_M)\Delta_M$ and $(X_N)\Delta_N$ might share intuitionistic variables. So it might be necessary to weaken Ξ in order to get the translation correct. Subject to this explanation, we now calculate:

$$\begin{array}{rcll} \mathcal{T}(\mathcal{L}(MN)) & = & \mathcal{T}(\text{LIMP-E}_{\mathcal{L}(\phi),\mathcal{L}(\psi)}(\mathcal{L}(M),\mathcal{L}(N)) & \text{definition} \\ & = & \mathcal{T}(\mathcal{L}(M))\mathcal{T}(\mathcal{L}(N)) & \text{definition} \\ & = & MN & \text{IH} \end{array}$$

$(M\lambda\Lambda I)$ Suppose $\Gamma \vdash_\Sigma \lambda x{:}C.M{:}\Lambda x{:}C.\phi$. Then we calculate

$$\begin{array}{rcll} \mathcal{T}(\mathcal{L}(\lambda x{:}C.M)) & & & \\ & = & \mathcal{T}(\text{LALL-I}_{x,\mathcal{L}(\phi)}(\mathcal{L}(M))) & \text{with } (X)\Delta = (X_M - (I,x{:}C))\Delta_M \\ & = & \lambda x{:}C.\mathcal{T}(\mathcal{L}(M)) & \text{definition} \\ & = & \lambda x{:}C.M & \text{IH} \end{array}$$

In fact, the $x{:}\mathcal{L}(\phi)$ will be the last leaf of the bunch $(X_M)\Delta_M$.

$(M\Lambda\mathcal{E})$ Suppose $\Xi \vdash_\Sigma Mt{:}\phi[t/x]$. Then we calculate

$$\begin{array}{rcll} \mathcal{T}(\mathcal{L}(Mt)) & = & \mathcal{T}(\text{LALL-E}_{x,\mathcal{L}(\phi),\mathcal{L}(t)}(\mathcal{L}(M))) & \text{definition} \\ & = & \mathcal{T}(\mathcal{L}(M))\mathcal{T}(\mathcal{L}(t)) & \text{definition} \\ & = & Mt & \text{IH} \end{array}$$

The above calculation is done subject to the same explanation regarding the $(\multimap E)$ calculation above.

$(M\&I)$ Suppose $\Gamma \vdash_\Sigma \langle M,N\rangle{:}A\&B$. Then we calculate

$$\begin{array}{rcll} \mathcal{T}(\mathcal{L}(\langle M,N\rangle)) & = & \mathcal{T}(\text{WITH-I}_{\mathcal{L}(A),\mathcal{L}(B)}(\mathcal{L}(M),\mathcal{L}(N))) & \text{definition} \\ & = & \langle \mathcal{T}(\mathcal{L}(A)),\mathcal{T}(\mathcal{L}(B))\rangle & \text{definition} \\ & = & \langle M,N\rangle & \text{IH} \end{array}$$

$(M\&\mathcal{E}_i)$ Suppose $\Gamma \vdash_\Sigma \pi_i(M){:}A_i$. Then we calculate

$$\begin{array}{rcll} \mathcal{T}(\mathcal{L}(\pi_i(M))) & = & \mathcal{T}(\text{WITH-E}^i_{\mathcal{L}(A_0),\mathcal{L}(A_1)}(\mathcal{L}(M))) & \text{definition} \\ & = & \pi_i(\mathcal{T}(\mathcal{L}(M))) & \text{definition} \\ & = & \pi_i(M) & \text{IH} \end{array}$$

$(M\lambda\Lambda!I)$ Suppose $\Gamma \vdash_\Sigma \lambda x!\phi.M{:}\phi \to \psi$. Then we calculate

$$\begin{array}{lll} \mathcal{T}(\mathcal{L}(\lambda x!\phi.M)) & & \\ = & \mathcal{T}(\text{IIMP-I}_{\mathcal{L}(\phi),\mathcal{L}(\psi)}(x{:}\mathcal{L}(M))) & \text{with } (X)\Delta = (X_M)\Delta_M - (1;x{:}\mathcal{L}(\phi)) \\ = & \lambda x!\mathcal{T}(\mathcal{L}(\phi)).\mathcal{T}(\mathcal{L}(M)) & \text{definition} \\ = & \lambda x!\phi.M & \text{IH} \end{array}$$

In fact, the $x{:}\mathcal{L}(\phi)$ will be the last leaf of the bunch $(X_M)\Delta_M$.

$(M\Lambda!\mathcal{E})$ Suppose $\Xi \vdash_\Sigma MN{:}\psi$. Before we give the calculation, it is worth making a little note regarding the correctness of the translation of the context. Now the judgement $\Xi \vdash_\Sigma MN{:}\psi$ holds because $\Gamma \vdash_\Sigma M{:}\phi \to \psi$ and $!\Delta \vdash_\Sigma N{:}\phi$, where Ξ is the sharing-sensitive join of Γ and $!\Delta$. By induction, these premiss judgements are translated as $(X_M)\Delta_M \vdash_{\Sigma,\Xi} \mathcal{L}(M){:}\mathcal{L}(\phi) \to \mathcal{L}(\psi)$ and $(!X_N)!\Delta_N \vdash_{\Sigma,\Xi} \mathcal{L}(N){:}\mathcal{L}(\phi)$. An application of $(\to E)$ then gives us the judgement $(X_M;!X_N)\Delta_M;!\Delta_N \vdash_{\Sigma,\Xi} \mathcal{L}(M)\mathcal{L}(N){:}\mathcal{L}(\psi)$. Now the contexts $(X_M)\Delta_M$ and $(!X_N)!\Delta_N$ might share intuitionistic variables. So it might be necessary to weaken Ξ in order to get the translation correct. Subject to this explanation, we now calculate:

$$\begin{array}{rcll} \mathcal{T}(\mathcal{L}(MN)) & = & \mathcal{T}(\text{IIMP-E}_{\mathcal{L}(\phi),\mathcal{L}(\psi)}(\mathcal{L}(M),\mathcal{L}(N)) & \text{definition} \\ & = & \mathcal{T}(\mathcal{L}(M))\mathcal{T}(\mathcal{L}(N)) & \text{definition} \\ & = & MN & \text{IH} \end{array}$$

$(M\lambda\Lambda!I)$ Suppose $\Gamma \vdash_\Sigma \lambda x!C.M{:}\Lambda x!C.\phi$. Then we calculate

$$\begin{array}{lll} \mathcal{T}(\mathcal{L}(\lambda x!C.M)) & & \\ = & \mathcal{T}(\text{IALL-I}_{x,\mathcal{L}(\phi)}(\mathcal{L}(M))) & \text{with } (X)\Delta = (X_M - (1;x{:}C))\Delta_M \\ = & \lambda x!C.\mathcal{T}(\mathcal{L}(M)) & \text{definition} \\ = & \lambda x!C.M & \text{IH} \end{array}$$

In fact, the $x{:}\mathcal{L}(\phi)$ will be the last leaf of the bunch $(X_M)\Delta_M$.

$(M\Lambda\mathcal{E})$ Suppose $\Xi \vdash_\Sigma Mt{:}\phi[t/x]$. Then we calculate

$$\begin{array}{rcll} \mathcal{T}(\mathcal{L}(Mt)) & = & \mathcal{T}(\text{IALL-E}_{x,\mathcal{L}(\phi),\mathcal{L}(t)}(\mathcal{L}(M))) & \text{definition} \\ & = & \mathcal{T}(\mathcal{L}(M))\mathcal{T}(\mathcal{L}(t)) & \text{definition} \\ & = & Mt & \text{IH} \end{array}$$

The above calculation is done subject to the same explanation regarding the $(\to E)$ calculation above. $\square$

3.5 Other correspondences

The first description of such a correspondence, between intuitionistic logic and simply-typed λ-calculus, was given by Howard in around 1969 [How80]. Howard refers to Tait for the one-to-one correspondence between cut-elimination in the logic and normalization in the term calculus. Independently, de Bruijn interpreted propositions as types and proofs as terms while encoding (classical) mathematics as part of the Automath project. Hence, the propositions-as-types correspondence is also referred to as the Curry-Howard-de Bruijn correspondence. We have already commented that our approach follows that of Barendregt's, where the correspondence between λP, a $\lambda\Pi$-calculus-like first-order minimal dependent type theory, and *PRED*, a first-order minimal predicate logic, is shown [Bar92].

Barendregt deals with minimal predicate logic. Several other correspondences that we mention below only deal with the propositional part of the correspondence. Barendregt refers to Guevers thesis [Geu93], where the proposition-as-types embedding is extended to the entire λ-cube. Guevers also shows that completeness fails at the $\lambda P\omega$–$PRED\omega$ node of the cube. Barendregt's completeness theorem refers to Tonino and Fujita [TF92], who show the correspondence between higher order predicate logic and type theory (Tonino and Fujita's type theory is a subsystem of *CC*).

Abramsky [Abr93], Benton [Ben94] and Barber [Bar97] show the propositions-as-types correspondence for intuitionistic linear logic. Their techniques generally seem to be as follows: define the propositional logic and then decorate it with terms of the type theory. Our approach follows Barendregt in that we define two separate systems and then show the correspondence.

Gabbay and de Queiroz show the the Curry-Howard correspondence for a family of propositional "resource logics" [GdQ92]. Each such natural deduction-style logic and type theory is defined by, firstly, constraining which axioms are allowed in the system and, secondly, by imposing an "abstraction discipline" which is basically a (meta-logical) side-condition on the $(\rightarrow I)$ rule. For instance, linear logic is defined by the combinators I (reflexivity) , B (left transitivity or prefixing), B' (right transitivity) and C (permutation), and abstraction is allowed as long as variables are bound "one by one" (as opposed to several, or none, bound in one action). Relevance logic is defined by adding the combinator S (distribution) to the above set and abstraction is allowed as long as it is "non-vacuous".

Independently, Wansing [Wan93] covers the same subject as Gabbay and de Queiroz. Wansing's logic is presented in a Hilbert-style and the analysis is more in the spirit of Howard in that it covers the correspondence between cut-elimination and normalization. Wansing does this for a minimal Lambek-style categorical logic and a λ-calculus which has two abstractions (a left one and a right one), and then presents methods to extend the system under consideration to a family of sub-structural logics included in propositional IL.

Of course, the main difference between our and these other treatments is of content; the logic we consider controls structurals in quite a different manner from, say, linear logic. As O'Hearn and Pym point out [OP99], **BI** is very likely a folklore relevant logic. But it appears never to have been explicitly written down or studied before.

3.6 Summary

In this chapter, we have set up the Curry-Howard-de Bruijn correspondence between the λΛ-calculus and a structural fragment of the logic **BI**. The key characteristic allowing the correspondence is to be able to extend, both in the type theory and the logic, the context in two distinct ways.

Chapter 4

Kripke resource semantics

4.1 Introduction

The internal logic of the $\lambda\Lambda$-calculus corresponds, in a certain way detailed in the previous chapter, to a fragment of the logic **BI**. A key characteristic of **BI** is the extra structure on the context — specifically, the two kinds of context extension operations — which allows the formation of the two kinds of function spaces. This can be understood, categorically, by a single category which has two kinds of monoidal structure on it. It can also be understood, model-theoretically, by a unique combination of two familiar ideas: a Kripke-style world semantics and an Urquhart-style resource semantics. In this chapter we will use the internal logic and its semantics to motivate an indexed categorical semantics for the type theory.

The categorical semantics of the $\lambda\Lambda$-calculus is given by a Kripke resource model, which is a monoid-indexed set of Kripke functors $\{\mathcal{J}_r{:}[\mathcal{W},[\mathcal{C}^{op},\mathbf{Cat}]] \mid r \in R\}$. The indexing monoid R is seen as providing an account of resource consumption. The main component of the Kripke functor is an indexed category $[\mathcal{C}^{op},\mathbf{Cat}]$, the base of which has two kinds of monoidal structure on it. The extra structure on the indexed category includes a natural isomorphism which allows the formation of the function spaces. The model can be seen as providing an indexed view appropriate to a notion of dependency in Read's bunches [Rea88].

This chapter is organized as follows. In § 4.2, we summarize the presentation of the type theory in both syntactic and algebraic forms. In § 4.3, we motivate the categorical semantics of the type theory. § 4.3 and § 4.4 form the main technical part of the chapter, the latter ending with a completeness theorem. We construct two example Kripke resource models. The first of these is the term model, whose construction is mainly to prove completeness. The second example model is a construction in which **Fam**, the category of families of sets, is interpreted not in **Set** but in a presheaf $\mathbf{Set}^{C^{op}}$, where C is a small monoidal category. This gives us enough structure to define two kinds of indexed products.

4.2 The λΛ-calculus

We briefly remind the reader of the syntactic presentation and internal logic of the λΛ-calculus. In the sequel, we shall refer to the natural deduction presentation of the λΛ-calculus as System **N**.

4.2.1 The syntactic presentation

The λΛ-calculus is a first-order dependent type theory with two kinds of function spaces, a linear one and an intuitionistic one. The calculus is used for deriving type judgements, the main one of which is $\Gamma \vdash_{\Sigma} M{:}A$, that M is a term of type A in context Γ and signature Σ.

The definition of the type theory depends crucially on several notions to do with the joining and maintenance of contexts. The notion of context joining is implemented by a ternary relation $[\Xi;\Gamma;\Delta]$, read as "Γ and Δ join to form Ξ". The notion of sharing, which deals with the well-formedness of types in binary, multiplicative rules and utilizes the notion of multiple occurrences, is implemented by the κ function. These notions allow the formation of full linear dependent types.

In Chapter 2, we gave the natural deduction presentation of the type theory and proved Church-Rosser and Strong Normalization, and so decidability, for the judgement $\Gamma \vdash_{\Sigma} M{:}A$.

4.2.2 The internal logic

The internal logic of the type theory corresponds to a structural fragment of the logic **BI** of bunched implications. The main judgement of **BI** is $(X)\Gamma \vdash_{\Sigma,\Xi} \phi$, to be read as "$\phi$ is a proposition in the variable context X and the propositional context Γ with respect to the term signature Σ and the propositional signature Ξ". The structurals are controlled by two distinct context-forming operators, the ";", which admits weakening and contraction, and the ",", which does not. This structure in the context, or bunches, allows the formation of two kinds of functions and quantifiers, linear ones and intuitionistic ones.

The logic **BI** was discussed in Chapter 3 and a propositions-as-types correspondence was shown between the $\lambda\Lambda$-calculus and a structural fragment of it. We will start Section 4.3 by discussing the Kripke resource semantics of **BI**, utilizing that semantics to work out the mathematical structure needed for the semantics of the type theory.

4.2.3 An algebraic presentation

In preparation for our presentation of a categorical semantics of the $\lambda\Lambda$-calculus in general and, in particular, for the completeness argument later, we give an algebraic presentation of the $\lambda\Lambda$-calculus type theory. The idea is to consider provably well-formed syntactic object modulo definitional equality. We let $|U|$ denote the $\alpha\beta\eta$-equivalence class of expressions of the $\lambda\Lambda$-calculus, though we will tend to omit the $|-|$ brackets where no confusion can arise.

Definition 46 *Let Σ be a signature. The base category $C(\Sigma)$ of contexts and realizations is defined as follows:*

- *Objects: contexts Γ such that* $\mathbf{N}$ *proves* $\vdash_\Sigma \Gamma\ \mathsf{context}$*;*
- *Arrows: realizations $\Gamma \xrightarrow{\langle M_1,\ldots,M_n\rangle} \Delta$ such that* $\mathbf{N}$ *proves* $\Gamma \vdash_\Sigma (M_i{:}A_i)[M_j/x_j]_{j=1}^{i-1}$, *where* $\Delta = x_1{\in}A_1,\ldots,x_n{\in}A_n$.
  - *Identities are $x_1{\in}A_1,\ldots,x_n{\in}A_n \xrightarrow{\langle x_1,\ldots,x_n\rangle} x_1{\in}A_1,\ldots,x_n{\in}A_n$. We will write the identity arrow on Γ as 1_Γ;*
  - *Composition is given by substitution. If $f = \Gamma \xrightarrow{\langle M_1,\ldots,M_n\rangle} \Delta$ and $g = \Delta \xrightarrow{\langle N_1,\ldots,N_p\rangle} \Theta$, then $f;g = \Gamma \xrightarrow{\langle N_1[M_j/y_j]_{j=1}^{n},\ldots,N_p[M_j/y_j]_{j=1}^{n}\rangle} \Theta$.* □

Proposition 47 *$C(\Sigma)$ is a category.*

Proof Let

$$\Gamma = x_1{\in}A_1,\ldots,x_m{\in}A_m$$
$$\Delta = y_1{\in}B_1,\ldots,y_n{\in}B_n$$
$$\Theta = z_1{\in}C_1,\ldots,z_p{\in}C_p$$
$$\Phi = w_1{\in}D_1,\ldots,w_q{\in}D_q$$

be objects and

$$\Gamma \xrightarrow{\langle M_1,\ldots,M_n\rangle} \Delta$$
$$\Delta \xrightarrow{\langle N_1,\ldots,N_p\rangle} \Theta$$
$$\Theta \xrightarrow{\langle P_1,\ldots,P_q\rangle} \Phi$$

be arrows. We check that 1_Δ is an identity morphism

$$\begin{aligned}
1_\Delta;N &= \langle N_1[N_i/y_i]_{i=1}^n,\ldots,N_p[N_i/y_i]_{i=1}^n\rangle \\
&= \langle N_1,\ldots,N_p\rangle \\
&= N
\end{aligned}$$

$$\begin{aligned}
M;1_\Delta &= \langle y_1[M_i/y_i]_{i=1}^n,\ldots,y_n[M_i/y_i]_{i=1}^n\rangle \\
&= \langle M_1,\ldots,M_p\rangle \\
&= M
\end{aligned}$$

and that composition is associative

$$\begin{aligned}
(M;N);P &= \langle N_1[M_i/y_i]_{i=1}^n,\ldots,N_p[M_i/y_i]_{i=1}^n\rangle;P \\
&= \langle P_1[N_j[M_i/y_i]_{i=1}^n/z_j]_{j=1}^p,\ldots,P_q[N_j[M_i/y_i]_{i=1}^n/z_j]_{j=1}^p\rangle \\
&= \langle P_1[N_j/z_j]_{j=1}^p[M_i/y_i]_{i=1}^n,\ldots,P_q[N_j/z_j]_{j=1}^p[M_i/y_i]_{i=1}^n\rangle \\
&= M;\langle P_1[N_j/z_j]_{j=1}^p,\ldots,P_q[N_j/z_j]_{j=1}^p\rangle \\
&= M;(N;P)
\end{aligned}$$

□

In the judgements $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$ and $\Gamma \vdash_\Sigma M{:}A$, the context Γ, which is an object of $C(\Sigma)$ according to Definition 46, can be seen as an index for the type A and the term M. That is, M and A depend on the variables declared in Γ. This can be seen in the internal logic too, where in the judgement $(X)\Gamma \vdash \phi$, X is an index for ϕ. We formalize this for the algebraic presentation of the syntax by taking $C(\Sigma)$ to be the base of a strict indexed category as follows.

Definition 48 *We inductively define a strict indexed category $\mathcal{E}(\Sigma)$ over the base category $C(\Sigma)$*

$$\mathcal{E}(\Sigma) : C(\Sigma)^{op} \to \mathbf{Cat},$$

where **Cat** *is the category of small categories and functors, as follows:*

- *For each Γ in $C(\Sigma)$, the category $\mathcal{E}(\Sigma)(\Gamma)$ is defined as follows:*
  - *Objects: Types A such that* **N** *proves $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$;*
  - *Morphisms: $A \xrightarrow{M} B$ where the object M is such that $\Gamma, x{:}A \xrightarrow{M} y{:}B$ in $C(\Sigma)$. By the classifying category theorem which follows, this amounts to the assertion* **N** *proves $\Gamma, x{:}A \vdash_\Sigma M{:}B$. Composition is given by substitution.*

- *For each $f : \Gamma \to \Delta$ in $C(\Sigma)$, $\mathcal{E}(\Sigma)(f)$ is a functor $f^* : \mathcal{E}(\Sigma)(\Delta) \to \mathcal{E}(\Sigma)(\Gamma)$ given by $f^*(A) \overset{\text{def}}{=} A[f]$ and $f^*(M) \overset{\text{def}}{=} M[f]$.* □

Working in **Cat** might raise some doubts about completeness (in particular, the presence of identity arrows) for the linear case. But we note that the arrow $A \multimap A$ (or $1 \longrightarrow A \multimap A$; the term model considers them identical) over the context Γ is given by the judgement $\Gamma, x{:}A \vdash_\Sigma x{:}A$. This is a valid one, and does not amount to weakening because the variables declared in Γ are used in A. If we had not defined the arrows in each $\mathcal{E}(\Sigma)(\Gamma)$ as above, then we might, in order to achieve completeness, have needed to have worked in a weaker setting, such as a presheaf $\mathcal{E}(\Sigma){:}C(\Sigma)^{op} \to \mathbf{Set}$ [Rd97].

The relation between the type theory and the category defined by the two definitions above is given by the following theorem, which states that the term category defines no more and no less than what can be proved in **N**.

Theorem 49 (Classifying category) *Let Σ be a signature and let Γ, M and A be in $\alpha\beta\eta$-normal form.*

- **N** *proves* $\vdash_\Sigma \Gamma$ context *if and only if Γ is an object of $C(\Sigma)$;*
- **N** *proves* $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$ *if and only if A is an object of $\mathcal{E}(\Sigma)(\Gamma)$;*
- *Let $M_1, \ldots, M_n$ be objects.* **N** *proves $\Gamma \vdash_\Sigma M_i{:}B_i[M_j/x_j]_{j=1}^{i-1}$ if and only if $\Gamma \xrightarrow{\langle M_1,\ldots,M_n\rangle} y_1{\in}B_1, \ldots, y_n{\in}B_n$ is an arrow of $C(\Sigma)$;*
- **N** *proves $\Gamma \vdash_\Sigma M{:}A$ if and only if $\langle\rangle \xrightarrow{M} A$ is an arrow of $\mathcal{E}(\Sigma)(\Gamma)$.*

Proof By induction: in the forward direction on the structure of the proofs in **N**; and the backward direction on the complexity of expressions. The arguments are straight forward and we omit the details. □

4.3 Kripke resource models of the $\lambda\Lambda$-calculus

4.3.1 Kripke resource $\lambda\Lambda$-structure

We motivate the mathematical structure which is used to model the $\lambda\Lambda$-calculus by considering, informally, models of the internal logic. In fact, the structure we motivate will be quite modular;

a sub-structure will model the intuitionistic $\{\rightarrow, \Pi\}$-fragment of the λΛ-calculus (*i.e.*, the λΠ-calculus).

The key issue in the syntax concerns the co-existing linear and intuitionistic function spaces and quantifiers. This distinction can be explained by reference to a *resource semantics*. The notion of resource, such as time and space, is a primitive one in informatics. Essential aspects of a resource include our ability to identify elements (including the null element) of the resource and their combinations. Thus we work with a resource monoid $(R, +, 0)$. We can also imagine a notion of comparison $\sqsubseteq$ between resources, indicating when one resource is better than another, in that it may prove more propositions. Similar ideas can be seen, *post hoc*, in the relevant logic literature [Urq72].

A resource semantics elegantly explains the difference between the linear and intuitionistic connectives in that the action, or computation, of the linear connectives can be seen to consume resources. We consider this, informally, for the internal logic judgement $(X)\Delta \vdash \phi$. Let $\mathcal{M} = (M, \cdot, e, \sqsubseteq)$ be a Kripke resource monoid. A simplified version of the forcing relation for the two implications is defined as follows:

1. $r \models \phi \multimap \psi$ if and only if for all $s \in M$ if $s \models \phi$ then $r \cdot s \models \psi$

2. $r \models \phi \rightarrow \psi$ if and only if for all $s \in M$ if $s \sqsubseteq r$ then $s \models \psi$

A similar pair of clauses defines the forcing relation for the two **BI** quantifiers (here we need an environment $u \in [\![X]\!]r$ appropriate to the bunch variables X at world r, where $[\![X]\!]$ is the interpretation of the bunch of variables X in $\mathbf{Set}^{\mathcal{M}^{op}}$):

1. $(X)u, r \models \forall x.\phi$ if and only if for all $s \sqsubseteq r$. for all $d \in Dn$. $(X;x)([\![X]\!](s \sqsubseteq r)u, d), s \models \phi$

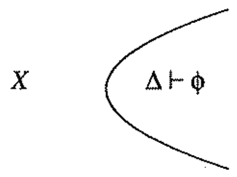
2. $(X)u, r \models \forall_{\text{new}} x.\phi$ if and only if for all s . for all $d \in Dn$. $(X,x)[u,d], r \cdot s \models \phi$

Here $D{:}\mathcal{M}^{op} \rightarrow \mathbf{Set}$ is a domain of individuals, $(-,-)$ is cartesian pairing and $[-,-]$ is the pairing operation defined by Day's tensor product construction in $\mathbf{Set}^{\mathcal{M}^{op}}$. The resource semantics can be seen to combine Kripke's semantics for intuitionistic logic and Urquhart's semantics for relevant logic. Details are in [OP99].

Suppose we have a category $\mathcal{E}$ where the propositions will be interpreted. Then we will index $\mathcal{E}$ in two ways for the purposes of interpreting the type theory. First, we index it by a Kripke world structure $\mathcal{W}$. This is to let the functor category $[\mathcal{W}, \mathcal{E}]$ have enough strength to model the

$\{\multimap, \forall\}$-fragment of the internal logic and so correspond to Kripke-style models for intuitionistic logic. And, second, we index $[\mathcal{W}, \mathcal{E}]$ by a resource monoid R. The structure we obtain is an R-indexed set of Kripke functors $\{\mathcal{J}_r \mid r \in R\}$. We remark that the separation of worlds from resources considered in this structure emphasizes a sort of "phase shift" [Gir87, HM94]. We briefly reconsider this choice in Section 4.5 later.

We now consider how to model the propositions and so explicate the structure of $\mathcal{E}$. The basic judgement of the internal logic is $(X)\Delta \vdash \phi$, that ϕ is a proposition in the context Δ over the context X. One reading of this judgement, and perhaps the most natural, is to see X as an index for the propositional judgement $\Delta \vdash \phi$:



This reading can be extended to the type theory, where, in the basic judgement $\Gamma \vdash_\Sigma M{:}A$, Γ can be seen as an index for $M{:}A$ or that $M{:}A$ depends on Γ for its meaning. Thus we are led to using the technology of indexed category theory [PS78]. More specifically, in the case of the type theory, the judgement $\Gamma \vdash_\Sigma M{:}A$ is modelled as the arrow $1 \xrightarrow{[\![M]\!]} [\![A]\!]$ in the fibre over $[\![\Gamma]\!]$ in the strict indexed category $\mathcal{E}{:}\,C^{op} \to \mathbf{Cat}$.

We remark that this is not the only technique for modelling a typing judgement; Cartmell [Car86], Pitts [Pit92] and several other authors use a more "one-dimensional" structure which relies on the properties of certain classes of maps to model the intuitionistic fragment of the $\lambda\Lambda$-calculus. These are formally equivalent to the indexed approach but the latter is appealing for one main reason: it provides a technical separation of conceptually separate issues. For instance, at a logical level, the base and fibres deal, respectively, with terms and propositions. At the cost of less bureaucracy, these issues would be muddled in the non-indexed approach.

We need the base category C to account for the structural features of the type theory and its internal logic. Recall that, proof-theoretically, the two function spaces and quantifiers arise because of extra structure, *viz.* the two types of context extension operators, in the context. To model the context, we work with a category with two kinds of structure on it. This leads us to the following definition.

Definition 50 (doubly monoidal category) *A doubly monoidal category is a category C equipped with two monoidal structures $(\otimes, I)$ and $(\times, 1)$.* □

The definition requires some comments. Firstly, there is no requirement that the bifunctors $\otimes$ and $\times$ be symmetrical too as the contexts that the objects are intended to model are (ordered) lists. Secondly, the use of the symbol $\times$ as one of the context extension operators suggests that $\times$ is a cartesian product. This is the case when $\{\mathcal{J}_r \mid r \in R\}$ is a model of the internal logic, where there are no dependencies within the variable context X, but not when $\{\mathcal{J}_r \mid r \in R\}$ is a model of the type theory, where there are dependencies within Γ. In the latter case, we have the property that for each object D extended by $\times$, there is a first projection map $p_{D,A}{:}D \times A \to D$. There is no second projection map $q_{D,A}{:}D \times A \to A$ in C, as A by itself may not correspond to a well-formed type.

The interaction between projection and other maps in C is stated by requiring the following pullback in C:

$$\begin{tikzcd}
D \times f^*(A) \arrow[r, "f \times 1_A"] \arrow[d, "p_{D,f^*A}"'] \arrow[dr, phantom, "\lrcorner", very near start] & E \times A \arrow[d, "p_{E,A}"] \\
D \arrow[r, "f"'] & E
\end{tikzcd}$$

The pullback indicates, for the cartesian case, how to interpret realizations as tuples. Suppose $1 \xrightarrow{M} A$ is an arrow in the fibre over D, then there exists a unique arrow $D \xrightarrow{\langle 1_D \times M \rangle} D \times A$. The pullback does not cover the case for the monoidal extension. For that, we must require the existence of the unique arrow $D(= D \otimes I) \xrightarrow{\langle 1_D \otimes M \rangle} D \otimes A$ in C, the tuples being given by the bifunctoriality of $\otimes$.

A doubly monoidal category C with both exponentials or, alternatively, C equipped with two monoidal closed structures $(\times, \to, 1)$ and $(\otimes, \multimap, I)$, is called a doubly closed category (DCC) in O'Hearn and Pym [OP99]. DCCs provide a class of models of **BI** in which both function spaces are modelled within C. We will work with the barer doubly monoidal category, requiring some extra structure on the fibres to model the function space.

Models of intuitionistic linear logic, such as Barber-Plotkin's DILL [Bar97], have to deal with two kinds of context extensions too. The technique used there is to work with a pair of categories, a monoidal one and a cartesian one, together with a monoidal adjunction between them. But this forces too much of a separation between the linear and intuitionistic parts of a context to be useful for modelling dependency. More specifically, suppose we wanted to model $x!A, y{:}Bx$. In the Barber-Plotkin scheme, A would be interpreted in the cartesian category and Bx in the monoidal one. However, Bx by itself is not a valid object in the monoidal category; it needs an x of type A in order to be well-formed.

We now consider how the function spaces are modelled. In the intuitionistic case, the weakening functor $p^*_{D,A}$ has a right adjoint $\Pi_{D,A}$ which satisfies the Beck-Chevalley condition. In fact, this amounts to the existence of a natural isomorphism cur_W

$$\frac{Hom_{\mathcal{J}_r(W)(D\times A)}(p^*_{D,A}(C), B)}{Hom_{\mathcal{J}_r(W)(D)}(C, \Pi_{D,A}(B))}$$

The absence of weakening for the linear context extension operator means that we can't model Λ in the same way. But the structure displayed above suggests a way to proceed. It is sufficient to require the existence of a natural isomorphism $\Lambda_{D,A}$

$$\frac{Hom_{\mathcal{J}_{r+r'}(W)(D\bullet A)}(1, B)}{Hom_{\mathcal{J}_r(W)(D)}(1, \Lambda x\in A\,.B)}$$

in the indexed category. Here, we use $\bullet$ to range over $\otimes$ and $\times$, and $\in$ to range over $:$ and $!$. There are a couple of remarks that need to be made about the isomorphism. Firstly, it refers only to hom-sets in the fibre whose source is 1. This restriction, which avoids the need to establish the well-foundedness of an arbitrary object over both D and $D \bullet A$, suffices to model the judgement $\Gamma \vdash_\Sigma M{:}A$ as an arrow $1 \overset{[\![M]\!]}{\to} [\![A]\!]$ in the fibre over $[\![\Gamma]\!]$: examples are provided by both the term and set-theoretic models that we will present later. The second remark we wish to make is that the extended context is defined in the $r+r'$-indexed functor. The reason for this can be seen by observing the form of the forcing clause for application in **BI**. Given these two remarks, the above isomorphism allows the formation of function spaces.

Finally, the additive conjunction & is modelled by requiring each category $\mathcal{J}_r(W)(D)$ to have products.

This completes our motivation for the categorical semantics of the $\lambda\Lambda$-calculus and its internal logic. To summarize, the structure we use to model the $\lambda\Lambda$-calculus is given by an R-indexed set of functors $\{\mathcal{J}_r{:}[\mathcal{W}, [C^{op}, \mathbf{Cat}]] \mid r \in R\}$. The resource indexing and the structure on C is sufficient to model both the linear and intuitionistic function spaces. The approach is modular enough to also provide a categorical semantics for the intuitionistic fragment of the $\lambda\Lambda$-calculus, the $\lambda\Pi$-calculus. Basically, we work with a single functor $\mathcal{J}_r{:}[\mathcal{W},[\mathcal{D}^{op},\mathbf{Cat}]]$, where $\mathcal{D}$ is a category with only the cartesian structure $(\times,1)$ on it.

We are led to the following definition. We remind the reader of some notational conventions. We use $\bullet$ to range over both linear $\otimes$ and intuitionistic $\times$ context extensions, $\langle f,g\rangle$ to range over both linear $\langle f\otimes g\rangle$ and intuitionistic $\langle f\times g\rangle$ tuples, and, in the syntax, $x{\in}A$ to range over both linear $x{:}A$ and intuitionistic $x!A$ declarations.

Definition 51 *Let $\langle R,+,0\rangle$ be a commutative monoid (of "resources"). A* <u>*Kripke resource $\lambda\Lambda$-structure*</u> *is an R-indexed set of functors*

$$\{\mathcal{J}_r : [\mathcal{W},[C^{op},\mathbf{Cat}]] \mid r\in R\}$$

where $\langle\mathcal{W},\leq\rangle$ is a poset, $C^{op}=\amalg_{W\in\mathcal{W}}C_W^{op}$, where $W\in\mathcal{W}$ and each C_W is a small doubly monoidal category with $1\cong I$, and **Cat** *is the category of small categories and functors such that*

1. *Each $\mathcal{J}_r(W)(D)$ has a terminal object, $1_{\mathcal{J}_r(W)(D)}$, preserved on the nose by each $f^*(=\mathcal{J}_r(W)(f))$, where $f{:}E\to D\in C_W$;*

2. *For each $W\in\mathcal{W}$, $D\in C_W$ and an object $A\in\mathcal{J}_r(W)(D)$, there is a $D\bullet A\in C_W$.*

   For the cartesian extension, there are canonical first projections $D\times A\overset{p_{D,A}}{\to}D$ and canonical pullbacks

$$\begin{tikzcd}
E\times f^*(A) \arrow[r, "f\times 1_A"] \arrow[d, "p_{E,f^*A}"'] \arrow[dr, phantom, "\lrcorner", very near start] & D\times A \arrow[d, "p_{D,A}"] \\
E \arrow[r, "f"'] & D
\end{tikzcd}$$

   The pullback indicates, for the cartesian case, how to interpret realizations as tuples. In particular, for each $1\overset{M}{\to}A\in\mathcal{J}_r(W)(D)$, there exists a unique arrow $D\overset{\langle 1\times M\rangle}{\to}D\times A$ in C_W. It does not cover the case for the monoidal extension. For that, we require there to exist a unique $D(=D\otimes I)\overset{\langle 1\otimes M\rangle}{\to}D\otimes A$, the tuples being given by the bifunctoriality of $\otimes$.

   For both extensions, there is a canonical second projection $1\overset{q_{D,A}}{\to}A$ in the fibre over $D\bullet A$.

   These maps are required to satisfy the strictness conditions that $(1_D)^*(A)=A$ and $1_D\bullet 1_A=1_{D\bullet A}$ for each $A\in\mathcal{J}_r(W)(D)$; $g^*(f^*(A))=(g;f)^*(A)$ and $(g\bullet f^*(A));f\bullet A=(g;f)\bullet A$ for each $F\overset{g}{\to}E$ and $E\overset{f}{\to}D$ in C_W. Moreover, for each W and D, $D\bullet 1_{\mathcal{J}_r(W)(D)}=D$;

3. *For each D, A, there is a natural isomorphism* $\Lambda_{D,A}$

$$\frac{Hom_{\mathcal{J}_{r+r'}(W)(D\bullet A)}(1,B)}{\overline{\overline{Hom_{\mathcal{J}_r(W)(D)}(1,\Lambda x{\in}A\,.B)}}}$$

where the extended context is defined in the $r+r'$*-indexed functor. This natural isomorphism is required to satisfy the Beck-Chevalley condition: For each* $E\xrightarrow{f}D$ *in* $\mathcal{C}_W$ *and each* B *in* $\mathcal{J}_r(W)(D\bullet A)$

$$f^*(\Lambda_{D,A}B)=\Lambda_{E,f^*A}((f\bullet id_A)^*B)\ ;$$

4. *Each category* $\mathcal{J}_r(W)(D)$ *has cartesian products.* □

For the purpose of our current study of modelling the type theory, structures, in which we consider only the arrows of type $1\to A$ in the fibre, suffice.

We conclude this section by showing a kind of a conservativity result. We embed a Kripke resource $\lambda\Lambda$-structure $\{\mathcal{J}_r \mid r\in R\}$ into a Kripke $\lambda\Pi$-structure $\mathcal{J}$ and show that the function space given by $\Lambda_{D,!A}(B)$ in the $\lambda\Lambda$-structure case is the same as that given by $\Pi_{D,A}(B)$ in the $\lambda\Pi$-structure case. This is to be expected, as a $\lambda\Pi$-structure has just the sub-structure of a $\lambda\Lambda$-structure to model the intuitionistic fragment of the $\lambda\Lambda$-calculus. Recall that a Kripke $\lambda\Pi$-structure is a functor $\mathcal{J}:[\mathcal{W},[\mathcal{D}^{op},\mathbf{Cat}]]$, where $\mathcal{D}$ is a category equipped with just the (modified) cartesian closed structure, plus the usual coherence conditions [Pym97].

Lemma 52 *The natural isomorphism*

$$cur_W : Hom_{\mathcal{J}(W)(D\bullet A)}(p^*_{D,A}(1),B)\quad\cong\quad Hom_{\mathcal{J}(W)(D)}(1,\Pi_{D,A}(B)) : cur_W^{-1}$$

in the Kripke $\lambda\Pi$*-structure* $\mathcal{J}$ *is just the* $\Lambda_{D,A}$ *natural isomorphism in the* $D\times A$ *case in the Kripke resource* $\lambda\Lambda$*-structure.*

Proof Fix a $\mathcal{J}_r$ to work in. Then define a translation ! from the $\lambda\Lambda$-structure to the $\lambda\Pi$-structure. Informally, the translation can be seen as follows:

$$\mathcal{J}_r(W)(A\bullet\ldots\bullet B) \ni 1\xrightarrow{f}D \qquad\mapsto\qquad \mathcal{J}(W)(A'\times\ldots\times B') \ni 1\xrightarrow{f'}D'$$

where the primed components are the same as the originals except that, for objects, the λΛ-structures $\{\multimap, \Lambda, \times, \rightarrow, \Pi\}$ operators are translated into the λΠ-structures $\{\rightarrow, \Pi, \times, \rightarrow, \Pi\}$ operators (we add $\times$ to the λΠ-structure in the obvious way) respectively. A similar translation is done for the morphisms. The key point is that !'s action on C is to forget the linear—intuitionistic context extension operator difference, translating both to the $\mathcal{D}$ context extension operator $\times$.

To show that the natural isomorphisms are the same, we start with the conclusion of the natural isomorphism $\Lambda_{D,A}$ and compute:

$$\dfrac{\dfrac{Hom_{\mathcal{J}_r(W)(D)}(1, \Lambda x!A\,.B)}{Hom_{\mathcal{J}(W)(!D)}(1, \Pi_{!D,!A}(!B))}\,!}{\overline{Hom_{\mathcal{J}(!W)(!D\times !A)}(p^*_{!D,!A}(1), (!B))}}\,cur$$

which is the translation of the premiss of the $\Lambda_{D,A}$ natural isomorphism. This is so as the first projection $p_{D,A}{:}D \times A \rightarrow D$ exists in each C. □

Syntactically, the Lemma 52 should be seen as a translation from the λΛ-calculus to the λΠ-calculus (and so the reverse of Definition 28). More semantically, it should perhaps be seen in a 2-categorical setting. The statement of the lemma would be that in some large category which has as objects structures, one should be able to construct an arrow from a λΛ-structure to a λΠ-structure.

4.3.2 Kripke resource Σ-λΛ-model

A Kripke resource model is a Kripke resource structure that has enough points to interpret not only the constants of Σ but also the λΛ-calculus terms defined over Σ and a given context Γ. Formally, a Kripke resource model is made up of five components: a Kripke resource structure that has Σ-operations, an interpretation function, two C-functors, and a satisfaction relation. Except for the structure, the components are defined, due to inter-dependences, simultaneously by induction on raw syntax. This explains the long and complex formal definition of the model (below).

Ignoring these inter-dependencies for a moment, we explain the purpose of each component of the model. First, the Kripke resource structure provides the abstract domain where the type theory is interpreted in. The Σ-operations provide the points to interpret constants in the signature. Second, the interpretation $[\![-]\!]$ is a partial function, mapping raw (that is, not necessary well-formed) contexts Γ to objects of C, types over raw contexts A_Γ to objects in the category indexed by the interpretation of Γ, and terms over raw contexts M_Γ to arrows in the category indexed by the interpretation of Γ. Types and terms are interpreted up to $\beta\eta$-equivalence. Fourth, the C-functors maintain the well-formedness of contexts with regard to joining and sharing. The model also needs to be constrained so that multiple occurrences of variables in the context get the same interpretation. Finally, fifth, satisfaction is a relation on worlds and sequents axiomatizing the desired properties of the model. In stronger logics, such as intuitionistic logic, the abstract definition of the model is sufficient to derive the properties of the satisfaction relation. van Dalen's description of a Kripke model for intuitionistic logic is done this way, for instance [vD94]. In our case, the definition has to be given more directly. Some further remarks on the model's definition are given immediately after its presentation.

We remark that we restrict our discussion of semantics to the $\Gamma \vdash_\Sigma M{:}A{:}\mathsf{Type}$-fragment. The treatment of the $\Gamma \vdash_\Sigma A{:}K$-fragment is undertaken analogously — in a sense, the $A{:}K$-fragment has the same logical structure as the $M{:}A$-fragment. To interpret the kind Type, we must require the existence of a chosen object, call it Ω, in each fibre. The object Ω must obey several equations: it must be preserved on the nose by any f^* and must behave well under quantification. Details of the treatment of the $A{:}K$-fragment in the case of contextual categories are in Streicher's thesis [Str88]. The analogous development in our setting is similar and we omit the details.

Definition 53 *Let Σ be a $\lambda\Lambda$-calculus signature. A Kripke resource Σ-$\lambda\Lambda$ model is a 5-tuple*

$$\langle \{\mathcal{J}_r{:}[\mathcal{W},[\mathcal{C}^{op},\mathbf{Cat}]] \mid r \in R\}, [\![-]\!], join, share, \models_\Sigma \rangle$$

where $\{\mathcal{J}_r{:}[\mathcal{W},[\mathcal{C}^{op},\mathbf{Cat}]] \mid r \in R\}$ is a Kripke resource $\lambda\Lambda$-structure that has Σ-operations, $[\![-]\!]$ is an interpretation from the raw syntax of the $\lambda\Lambda$-calculus to components of $\mathcal{J}_r{:}[\mathcal{W},[\mathcal{C}^{op},\mathbf{Cat}]]$, join and share are C-functors and $\models_\Sigma$ is a satisfaction relation on worlds and sequents, defined by simultaneous induction on the raw structure of the syntax as follows:

1. *The Kripke resource $\lambda\Lambda$-structure has Σ-operations if, for all W in $\mathcal{W}$,*

   (a) *Corresponding to each constant $c!\Lambda x_1{\in}A_1 \ldots \Lambda x_m{\in}A_m.\mathsf{Type} \in \Sigma$ there is in each $\mathcal{J}_r(W)([\![!\Phi]\!]^W_{\mathcal{J}_r})$ an operation op_c such that*

   $$op_c([\![M_{1\Gamma_1}]\!]^W_{\mathcal{J}_1}, \ldots, [\![M_{m\Gamma_m}]\!]^W_{\mathcal{J}_m})$$

   is an object of $\mathcal{J}_r(W)([\![\Xi]\!]^W_{\mathcal{J}_r})$, where

$$\llbracket \Xi \rrbracket^W_{\mathcal{J}_r} = \mathit{share\ join}(\llbracket \Gamma_m \rrbracket^W_{\mathcal{J}_m}, \ldots, \mathit{share\ join}(\llbracket \Gamma_1 \rrbracket^W_{\mathcal{J}_r}, \llbracket !\Phi \rrbracket^W_{\mathcal{J}_r}) \ldots) \ ;$$

(b) *Corresponding to each constant* $c!\Lambda x_1{\in}A_1 \ldots \Lambda x_m{\in}A_m . A \in \Sigma$ *there is in each*

$$\mathcal{J}_r(W)(\llbracket !\Phi, x_1{\in}A_1, \ldots, x_m{\in}A_m \rrbracket^W_{\mathcal{J}_r})$$

an arrow $1_{\mathcal{J}_r(W)(D)} \xrightarrow{op_c} \llbracket A \rrbracket^W_{\mathcal{J}_r}$, *where* $D = \llbracket !\Phi, x_1{\in}A_1, \ldots, x_m{\in}A_m \rrbracket^W_{\mathcal{J}_r}$;

2. *An* <u>*interpretation*</u> $\llbracket - \rrbracket^{-}_{\mathcal{J}_r}$*, in each such* $\mathcal{J}_r$*, satisfies, at each* W*:*

   (a) $\llbracket \langle\rangle \rrbracket^W_{\mathcal{J}_r} \simeq 1_C$;

   (b) $\llbracket \Gamma, x{:}A \rrbracket^W_{\mathcal{J}_{r+r'}} \simeq \llbracket \Gamma \rrbracket^W_{\mathcal{J}_r} \otimes \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_{r'}}$;

   (c) $\llbracket \Gamma, x!A \rrbracket^W_{\mathcal{J}_r} \simeq \llbracket \Gamma \rrbracket^W_{\mathcal{J}_r} \times \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_r}$;

   (d) $\llbracket \Gamma \xrightarrow{\langle M_1, \ldots, M_n \rangle} \Delta \rrbracket^W_{\mathcal{J}_r} \simeq \llbracket \Gamma \rrbracket^W_{\mathcal{J}_r} \xrightarrow{\langle \llbracket M_{1\Gamma_1} \rrbracket^W_{\mathcal{J}_1}, \ldots, \llbracket M_{n\Gamma_n} \rrbracket^W_{\mathcal{J}_n} \rangle} \llbracket \Delta \rrbracket^W_{\mathcal{J}_r}$,
   where $\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r} = \mathit{share\ join}(\llbracket \Gamma_n \rrbracket^W_{\mathcal{J}_n}, \ldots, \mathit{share\ join}(\llbracket \Gamma_2 \rrbracket^W_{\mathcal{J}_2}, \llbracket \Gamma_1 \rrbracket^W_{\mathcal{J}_1}) \ldots)$;

   (e) $\llbracket \langle\rangle_\Gamma \rrbracket^W_{\mathcal{J}_r} \simeq 1_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}}$;

   (f) $\llbracket (cM_1 \ldots M_n)_\Gamma \rrbracket^W_{\mathcal{J}_r} \simeq op_c(\llbracket M_{1\Gamma_1} \rrbracket^W_{\mathcal{J}_{r_1}}, \ldots, \llbracket M_{n\Gamma_n} \rrbracket^W_{\mathcal{J}_{r_n}})$ *in* $\mathcal{J}_r(W)(\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r})$,
   where $\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r} = \mathit{share\ join}(\llbracket \Gamma_n \rrbracket^W_{\mathcal{J}_n}, \ldots, \mathit{share\ join}(\llbracket \Gamma_2 \rrbracket^W_{\mathcal{J}_2}, \llbracket \Gamma_1 \rrbracket^W_{\mathcal{J}_1}) \ldots)$;

   (g) $\llbracket \Lambda x{:}A . B_\Gamma \rrbracket^W_{\mathcal{J}_r} \simeq \Lambda_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}, \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_s}}(\llbracket B_{\Gamma, x:A} \rrbracket^W_{\mathcal{J}_{r+s}})$, *where the extended context is defined in the* $r+s$*-indexed model ;*

   (h) $\llbracket \Lambda x!A . B_\Gamma \rrbracket^W_{\mathcal{J}_r} \simeq \Lambda_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}, \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_r}}(\llbracket B_{\Gamma, x!A} \rrbracket^W_{\mathcal{J}_r})$;

   (i) $\llbracket A \& B_\Gamma \rrbracket^W_{\mathcal{J}_r} \simeq \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_r} \times \llbracket B_\Gamma \rrbracket^W_{\mathcal{J}_r}$;

   (j) $\llbracket \langle\rangle_\Gamma \rrbracket^W_{\mathcal{J}_r} \simeq 1_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}}$;

   (k) $\llbracket c_{!\Gamma} \rrbracket^W_{\mathcal{J}_0} \simeq \Lambda^m(op_c)$;

   (l) $\llbracket x_{\Gamma, x:A} \rrbracket^W_{\mathcal{J}_r} \simeq q_{\llbracket \Gamma, x:A \rrbracket^W_{\mathcal{J}_r}}$;

   (m) $\llbracket \lambda x{:}A . M_\Gamma \rrbracket^W_{\mathcal{J}_r} \simeq \Lambda_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}, \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_r}}(\llbracket M_{\Gamma, x:A} \rrbracket^W_{\mathcal{J}_r})$;

   (n) $\llbracket \lambda x!A . M_\Gamma \rrbracket^W_{\mathcal{J}_r} \simeq \Lambda_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}, \llbracket !A_\Gamma \rrbracket^W_{\mathcal{J}_r}}(\llbracket M_{\Gamma, x!A} \rrbracket^W_{\mathcal{J}_r})$;

   (o) $\llbracket MN_\Xi \rrbracket^W_{\mathcal{J}_t} \simeq (\langle 1_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}}, \llbracket N_\Delta \rrbracket^W_{\mathcal{J}_s} \rangle^* (\Lambda^{-1}_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}, \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_s}}(\llbracket M_\Gamma \rrbracket^W_{\mathcal{J}_r})))$,
   where $\llbracket \Xi' \rrbracket^W_{\mathcal{J}_{r+s}} = \mathit{join}\langle \llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}, \llbracket \Delta \rrbracket^W_{\mathcal{J}_s} \rangle$ *and* $\llbracket \Xi \rrbracket^W_{\mathcal{J}_t} = \mathit{share}(\llbracket \Xi' \rrbracket^W_{\mathcal{J}_{r+s}})$;

(p) $[\![\langle M,N\rangle_\Gamma]\!]^W_{\mathcal{I}_r} \simeq \langle [\![M_\Gamma]\!]^W_{\mathcal{I}_r}, [\![N_\Gamma]\!]^W_{\mathcal{I}_r}\rangle$;

(q) $[\![\pi_i(M)_\Gamma]\!]^W_{\mathcal{I}_r} \simeq \pi_i([\![M_\Gamma]\!]^W_{\mathcal{I}_r})$, *where* $i \in \{0,1\}$.

Otherwise the interpretation is undefined.

3. *There exists a bifunctor* <u>*join*</u> *on C. The purpose of join, on objects, is to extend the first object with the second, discarding any duplicate cartesian objects. The definition of join on objects is as follows:*

$$\begin{array}{rcl}
join\langle [\![\langle\rangle]\!]^W_{\mathcal{I}_r}, [\![\langle\rangle]\!]^W_{\mathcal{I}_r}\rangle & = & [\![\langle\rangle]\!]^W_{\mathcal{I}_r} \\
join\langle [\![\Gamma]\!]^W_{\mathcal{I}_r}, [\![\Delta, x{:}A]\!]^W_{\mathcal{I}_{s+t}}\rangle & = & join\langle [\![\Gamma]\!]^W_{\mathcal{I}_r}, [\![\Delta]\!]^W_{\mathcal{I}_s}\rangle \otimes [\![x{:}A]\!]^W_{\mathcal{I}_t} \\
join\langle [\![\Gamma, x{:}A]\!]^W_{\mathcal{I}_{r+t}}, [\![\Delta]\!]^W_{\mathcal{I}_s}\rangle & = & join\langle [\![\Gamma]\!]^W_{\mathcal{I}_r}, [\![\Delta]\!]^W_{\mathcal{I}_s}\rangle \otimes [\![x{:}A]\!]^W_{\mathcal{I}_t} \\
join\langle [\![\Gamma, x!A]\!]^W_{\mathcal{I}_r}, [\![\Delta, x!A]\!]^W_{\mathcal{I}_s}\rangle & = & join\langle [\![\Gamma]\!]^W_{\mathcal{I}_r}, [\![\Delta]\!]^W_{\mathcal{I}_s}\rangle \times [\![x!A]\!]^W_{\mathcal{I}_{r+s}}
\end{array}$$

The definition of join on morphisms is similar. It is easy to see that $[\![\Xi]\!]^W_{\mathcal{I}_{r+s}} = join\langle [\![\Gamma]\!]^W_{\mathcal{I}_r}, [\![\Delta]\!]^W_{\mathcal{I}_s}\rangle$.

There exists a functor <u>*share*</u> *on C. The purpose of share is to regulate sharing of multiple occurrences of an object. The definition of share on objects is as follows:*

$$\begin{array}{rcl}
share([\![\langle\rangle]\!]^W_{\mathcal{I}_r}) & = & [\![\langle\rangle]\!]^W_{\mathcal{I}_r} \\
share([\![\Xi]\!]^W_{\mathcal{I}_r}) & = & join\langle share([\![\Phi]\!]^W_{\mathcal{I}_{s+s'}}), [\![\Psi]\!]^W_{\mathcal{I}_{t+u+u'}}\rangle \\
 & & \quad if\ [\![\Xi]\!]^W_{\mathcal{I}_t} = join\langle [\![\Gamma, x{:}A, \Gamma']\!]^W_{\mathcal{I}_{s+t+u}}, [\![\Delta, x{:}A, \Delta']\!]^W_{\mathcal{I}_{s'+t+u'}}\rangle \\
 & & \quad \not\exists y{:}B(x) \in \Gamma' \\
 & & \quad [\![\Phi]\!]^W_{\mathcal{I}_{s+s'}} = join\langle [\![\Gamma]\!]^W_{\mathcal{I}_s}, [\![\Delta]\!]^W_{\mathcal{I}_{s'}}\rangle \\
 & & \quad [\![\Psi]\!]^W_{\mathcal{I}_{t+u+u'}} = join\langle [\![x{:}A]\!]^W_{\mathcal{I}_t}, join\langle [\![\Gamma']\!]^W_{\mathcal{I}_u}, [\![\Delta']\!]^W_{\mathcal{I}_{u'}}\rangle\rangle \\
 & = & [\![\Xi]\!]^W_{\mathcal{I}_r}\ otherwise
\end{array}$$

The definition of share on morphisms is similar. The purpose of share is to ensure that the joined objects and morphisms are well-formed.

Both join and share "cut across interpretations" in that the result object is in a different R-indexed model from the argument object(s). This is necessary for defining the interpretation of function application ;

4. <u>*Satisfaction*</u> *in the model is a relation over worlds and sequents such that the following hold:*

(a) *$\mathcal{J}_r, W \models_\Sigma (c{:}A)\ [!\Gamma]$ if and only if $c \in dom(\Sigma)$;*

(b) *$\mathcal{J}_r, W \models_\Sigma (x{:}A)\ [\Gamma, x{\in}A]$ if and only if $[\![\Gamma, x{\in}A]\!]^W_{\mathcal{J}_r}$ is defined ;*

(c) *$\mathcal{J}_r, W \models_\Sigma (M : \Lambda x{:}A\,.B)\ [\Gamma]$ if and only if for all $W \leq W'$ and for all $r' \in R$, if $\mathcal{J}_s, W' \models_\Sigma (N{:}A)\ [\Delta]$, then $\mathcal{J}_t, W' \models_\Sigma (MN{:}B[N/x])\ [\Xi]$, where $[\![\Xi]\!]^{W'}_{\mathcal{J}_{r+s}} = join\langle [\![\Gamma]\!]^{W'}_{\mathcal{J}_r}, [\![\Delta]\!]^{W'}_{\mathcal{J}_s} \rangle$ and $[\![\Xi]\!]^{W'}_{\mathcal{J}_t} = share([\![\Xi']\!]^{W'}_{\mathcal{J}_{r+s}})$;*

(d) *$\mathcal{J}_r, W \models_\Sigma (M{:}A\&B)\ [\Gamma]$ if and only if $\mathcal{J}_r, W \models_\Sigma (\pi_i(M){:}A)\ [\Gamma]$, for $i \in \{0,1\}$;*

(e) *$\mathcal{J}_r, W \models_\Sigma (M : \Lambda x!A\,.B)\ [\Gamma]$ if and only if for all $W \leq W'$, if $\mathcal{J}_r, W \models_\Sigma (N{:}A)\ [!\Delta]$, then $\mathcal{J}_r, W \models_\Sigma (MN{:}B[N/x])\ [\Xi]$, with $[\![\Xi]\!]^W_{\mathcal{J}_r} = join\langle [\![\Gamma]\!]^W_{\mathcal{J}_r}, [\![!\Delta]\!]^W_{\mathcal{J}_r} \rangle$.*

We require two conditions:

1. *(Syntactic monotonicity) If $[\![X]\!]^W_{\mathcal{J}_r}$ is defined, then $[\![X']\!]^W_{\mathcal{J}_{r'}}$ is defined, for all subterms X' of X and summands r' of r. This condition is needed for various inductive arguments. It is not automatic as the interpretation is defined over raw objects ;*

2. *(Accessibility) The functor $\mathcal{J}_r(W)$ has domain $C = \amalg_{W \in \mathcal{W}} C^{op}_{\mathcal{J}_r(W)}$. So that $[\![\Gamma]\!]^W_{\mathcal{J}_r} \in C_W$ and $[\![\Gamma]\!]^{W'}_{\mathcal{J}_r} \in C_{W'}$. If there is an arrow $W \leq W' \in \mathcal{W}$, then*

   (a) *there exists a functor $\xi{:}C_W \to C_{W'}$ such that $\xi([\![X]\!]^W_{\mathcal{J}_r}) = [\![X]\!]^{W'}_{\mathcal{J}_r}$, where X, recall, ranges over contexts, types and terms ;*

   (b) *$\mathcal{J}_r(W')([\![\Gamma]\!]^W_{\mathcal{J}_r}) = \mathcal{J}_r(W')([\![\Gamma]\!]^{W'}_{\mathcal{J}_r})$ and $\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r}) = \mathcal{J}_r(W)([\![\Gamma]\!]^{W'}_{\mathcal{J}_r})$, for each context Γ; otherwise $\mathcal{J}_r(W')([\![\Gamma]\!]^{W'}_{\mathcal{J}_r})$ is undefined.*

This concludes the definition of the Kripke resource Σ-λΛ-model.

A few remarks concerning Definition 53 are in order.

The type theory has a structural freedom at the level of terms which, logically, allows the existence of multiple occurrences of the same proof. However, it can be that, in operating on the representation of two judgements, the same occurrence of an object in the base of the resulting representation is used to form the valid terms and types in both representations. This sharing requirement is regulated by the existence of a functor *share* on C defined as follows.

The second accessibility condition is the simplest one regarding the model-theoretic notion of *relativization*: that of interpreting constructs in one world and reasoning about them from the point of view of another. In the definition of model, and so in the sequel, the accessibility relation we take equates contexts, *etc.* over the worlds. A syntactic term can be seen, in a weak sense, as a "rigid designator", that is, one whose interpretation is the same over different worlds, for a semantic object. For example, suppose **N** proves $\Gamma \vdash_\Sigma M{:}A$. If $[\![M_\Gamma]\!]^W_{\mathcal{J}_r}$ is defined (given soundness this will be the case), then, for all $W \leq W' \in \mathcal{W}$, $[\![M_\Gamma]\!]^{W'}_{\mathcal{J}_r}$ is defined and equal to $[\![M_\Gamma]\!]^W_{\mathcal{J}_r}$. In a sense, the syntactic term M designates all objects $[\![M_\Gamma]\!]^{W'}_{\mathcal{J}_r}$.

We also remark that there are several notions of partiality in the model. Technically, the interpretation function is a partial one because it is defined for raw objects of the syntax. But partiality plays two other roles too. Firstly, there is dependent typing partiality to "bootstrap" the definition. And, secondly, there is Kripke semantic partiality of information, in which the further up the world structure one goes, the more objects have defined interpretations. We refer to Streicher [Str88], Pym [Pym97], and Mitchell and Moggi [MM91] for some comments regarding these matters.

We check the functoriality of *join* and *share*.

Lemma 54 *join and share are functors.*

Proof

1. The action of *join* on morphisms is defined is as follows:

$$\begin{array}{rcl}
join\langle [\![\langle\rangle]\!]^W_{\mathcal{J}_r}, [\![\langle\rangle]\!]^W_{\mathcal{J}_r} \rangle & = & [\![\langle\rangle]\!]^W_{\mathcal{J}_r} \\
join\langle [\![f]\!]^W_{\mathcal{J}_r}, [\![g \otimes x]\!]^W_{\mathcal{J}_{s+t}} \rangle & = & join\langle [\![f]\!]^W_{\mathcal{J}_r}, [\![g]\!]^W_{\mathcal{J}_s} \rangle \otimes [\![x]\!]^W_{\mathcal{J}_t} \\
join\langle [\![f \otimes x]\!]^W_{\mathcal{J}_{r+t}}, [\![g]\!]^W_{\mathcal{J}_s} \rangle & = & join\langle [\![f]\!]^W_{\mathcal{J}_r}, [\![g]\!]^W_{\mathcal{J}_s} \rangle \otimes [\![x]\!]^W_{\mathcal{J}_t} \\
join\langle [\![f \times x]\!]^W_{\mathcal{J}_r}, [\![g \times x]\!]^W_{\mathcal{J}_s} \rangle & = & join\langle [\![f]\!]^W_{\mathcal{J}_r}, [\![g]\!]^W_{\mathcal{J}_s} \rangle \times [\![x]\!]^W_{\mathcal{J}_{r+s}} \\
join\langle [\![f;f']\!]^W_{\mathcal{J}_r}, [\![g;g']\!]^W_{\mathcal{J}_s} \rangle & = & join\langle [\![f]\!]^W_{\mathcal{J}_r}, [\![g]\!]^W_{\mathcal{J}_s} \rangle ; join\langle [\![f']\!]^W_{\mathcal{J}_r}, [\![g']\!]^W_{\mathcal{J}_s} \rangle
\end{array}$$

   We need to show that *join* preserves identities and composition. We do the first by induction on the structure of objects. For the base cases, either or both of D or E are units. In which case the monoidal structures on C are sufficient to infer the result. There are three inductive cases to consider. One of these is when $D = D' \otimes A$. In this case, we compute:

$$\begin{array}{rcll}
join\langle 1_D, 1_E \rangle & = & (join\langle 1_{D'}, 1_E \rangle) \otimes A & \text{defn of } join \\
 & = & (1_{join\langle D', E \rangle}) \otimes A & \text{IH} \\
 & = & 1_{join\langle D' \otimes A, E \rangle} & \text{defn of } join
\end{array}$$

The second inductive case, when $E = E' \otimes A$, is done similarly. The third inductive case is when $D = D' \times A$ and $E = E' \times A$. In this case, we compute:

$$\begin{array}{rcll} join\langle 1_D, 1_E\rangle & = & (join\langle 1_{D'}, 1_{E'}\rangle) \times A & \text{defn of } join \\ & = & (1_{join\langle D',E'\rangle}) \times A & \text{IH} \\ & = & 1_{join\langle D' \times A, E' \times A\rangle} & \text{defn of } join \end{array}$$

We show that *join* preserves composition in its second place; the proof for the first is similar. Assume $E \xrightarrow{g} E' \xrightarrow{g'} E''$. Then we compute:

$$\begin{array}{rcll} join\langle 1, g;g'\rangle & = & join\langle 1, g;g' \times 1\rangle & \\ & = & g;g' \times 1 & \text{defn of } join \text{ and} \\ & & & \text{structure of } (\times, 1) \\ & = & g;g' & \\ & = & (g \times 1);(g' \times 1) & \\ & = & join\langle 1, g \times 1\rangle; join\langle 1, g' \times 1\rangle & \\ & = & join\langle 1, g\rangle; join\langle 1, g'\rangle & \end{array}$$

2. We need to show that *share* preserves identities and composition. We do the first by induction on the structure of objects. The case for $1_{\langle\rangle}$ is immediate by the definition of *share*. For the non-$\langle\rangle$, $\nexists y{:}Bx \in \Gamma'$ case we compute as follows:

$$\begin{array}{rcll} share(1_\Xi) & = & join\langle share(1_{join\langle\Gamma,\Delta\rangle}), 1_{join\langle A, join\langle\Gamma',\Delta'\rangle\rangle}\rangle & \\ & = & join\langle 1_{share\ join\langle\Gamma,\Delta\rangle}, 1_{join\langle A, join\langle\Gamma',\Delta'\rangle\rangle}\rangle & \text{IH} \\ & = & 1_{share(\Xi)} & \end{array}$$

For the proof of composition for this case, assume that $\Gamma \xrightarrow{f'} \Gamma'$, $\Gamma' \xrightarrow{f''} \Gamma''$, $\Delta \xrightarrow{g'} \Delta'$ and $\Delta' \xrightarrow{g''} \Delta''$; that $\Xi \xrightarrow{f} \Xi'$, $\Xi' \xrightarrow{g} \Xi''$; and that $f = share\ join\langle f', f''\rangle$, $g = share\ join\langle g', g''\rangle$. Then we compute:

$$\begin{array}{rcll} share(f;g) & = & join\langle f';g', f'';g''\rangle & \\ & = & join\langle f', f''\rangle; join\langle g', g''\rangle & \text{func of } join \\ & = & share(f); share(g) & \end{array}$$

with the appropriate side-conditions. □

We consider various model-theoretic properties of the satisfaction relation.

Lemma 55 (Monotonicity of $\models_\Sigma$) *Let Σ be a signature and $\langle\{\mathcal{J}_r \mid r \in R\}, [\![-]\!], join, share, \models_\Sigma\rangle$ be a model. If $\mathcal{J}_r, W \models_\Sigma (M{:}A)\ [\Gamma]$ and $W \leq W'$, then $\mathcal{J}_r, W' \models_\Sigma (M{:}A)\ [\Gamma]$.*

Proof By induction on the syntax of $M{:}A$. If $W \leq W'$, then, by accessibility, $[\![X]\!]^{W'}_{\mathcal{J}_r}$ is defined as $\xi[\![X]\!]^{W}_{\mathcal{J}_r}$, where X ranges over Γ, A and M. For each case of $M{:}A$, the conclusion is given by the definition of $\models_\Sigma$. □

Lemma 56 ($\models_\Sigma$-forcing via global sections) *Let $\langle\{\mathcal{J}_r \mid r \in R\}, [\![-]\!], join, share, \models_\Sigma\rangle$ be a Kripke resource model. $\mathcal{J}_r, W \models_\Sigma (M{:}A)\ [\Gamma]$ if and only if $[\![\Gamma]\!]^W_{\mathcal{J}_r}$ is defined, $[\![A_\Gamma]\!]^W_{\mathcal{J}_r}$ is defined, $[\![M_\Gamma]\!]^W_{\mathcal{J}_r}$ is defined and $1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![M_\Gamma]\!]^W_{\mathcal{J}_r}} [\![A_\Gamma]\!]^W_{\mathcal{J}_r}$ is an arrow in $\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})$.*

Proof By induction on the structure of $M{:}A$.

($c{:}A$) For the $\Rightarrow$ direction, we require the model to have enough points, and so get such an arrow. The $\Leftarrow$ direction is immediate from the definition of $\models_\Sigma$;

($x{\in}A$) For the $\Rightarrow$ direction, the second projection map $1 \xrightarrow{q} A$ in the fibre over the context $\Gamma, x{\in}A$ gives us the required arrow. The $\Leftarrow$ direction is immediate from the definition of $\models_\Sigma$;

($\lambda x{:}A\,.M : \Lambda x{:}A\,.B$) For the $\Rightarrow$ direction, by induction hypothesis we have that

$$1_{\mathcal{J}_{r+s}(W)([\![\Gamma,x{:}A]\!]^W_{\mathcal{J}_{r+s}})} \xrightarrow{[\![M_{\Gamma,x:A}]\!]^W_{\mathcal{J}_{r+s}}} [\![B_{\Gamma,x:A}]\!]^W_{\mathcal{J}_{r+s}}$$

is an arrow in $\mathcal{J}_{r+s}(W)([\![\Gamma,x{:}A]\!]^W_{\mathcal{J}_{r+s}})$. We then use the natural isomorphism Λ to get the arrow

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![\lambda x:A\,.M_\Gamma]\!]^W_{\mathcal{J}_r}} [\![\Lambda x{:}A\,.B_\Gamma]\!]^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})$. For the $\Leftarrow$ direction, suppose there exists an arrow

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![\lambda x:A\,.M_\Gamma]\!]^W_{\mathcal{J}_r}} [\![\Lambda x{:}A\,.B_\Gamma]\!]^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})$. It follows immediately that the existence of an arrow

$$1_{\mathcal{J}_s(W)([\![\Delta]\!]^W_{\mathcal{J}_s})} \xrightarrow{[\![N_\Delta]\!]^W_{\mathcal{J}_s}} [\![A_\Delta]\!]^W_{\mathcal{J}_s}$$

implies the existence of an arrow

$$1_{\mathcal{J}_t(W)(\llbracket\Xi\rrbracket^W_{\mathcal{J}_t})} \xrightarrow{\llbracket MN_\Xi\rrbracket^W_{\mathcal{J}_t}} \llbracket B[N/x]_\Xi\rrbracket^W_{\mathcal{J}_t}$$

where $\llbracket\Xi'\rrbracket^W_{\mathcal{J}_{r+s}} = join\langle\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r}, \llbracket\Delta\rrbracket^W_{\mathcal{J}_s}\rangle$ and $\llbracket\Xi\rrbracket^W_{\mathcal{J}_t} = share\llbracket\Xi'\rrbracket^W_{\mathcal{J}_{s+r}}$. The definition of $\models_\Sigma$ then gives us $\mathcal{J}_r, W \models_\Sigma (\lambda x{:}A.M : \Lambda x{:}A.B)\ [\Gamma]$;

$(\lambda x!A.M : \Lambda x!A.B)$ For the $\Rightarrow$ direction, by induction hypothesis we have that

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma,x!A\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket M_{\Gamma,x!A}\rrbracket^W_{\mathcal{J}_r}} \llbracket B_{\Gamma,x!A}\rrbracket^W_{\mathcal{J}_r}$$

is an arrow in $\mathcal{J}_r(W)(\llbracket\Gamma,x!A\rrbracket^W_{\mathcal{J}_r})$. We then use the natural isomorphism Λ to get the arrow

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket\lambda x!A.M_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket\Lambda x!A.B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$. For the $\Leftarrow$ direction, suppose there exists an arrow

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket\lambda x!A.M_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket\Lambda x!A.B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$. It follows immediately that the existence of an arrow

$$1_{\mathcal{J}_r(W)(\llbracket!\Delta\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket N_\Delta\rrbracket^W_{\mathcal{J}_r}} \llbracket A_\Delta\rrbracket^W_{\mathcal{J}_r}$$

implies the existence of an arrow

$$1_{\mathcal{J}_r(W)(\llbracket\Xi\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket MN_\Xi\rrbracket^W_{\mathcal{J}_r}} \llbracket B[N/x]_\Xi\rrbracket^W_{\mathcal{J}_r}$$

where $\llbracket\Xi\rrbracket^W_{\mathcal{J}_r} = join\langle\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r}, \llbracket!\Delta\rrbracket^W_{\mathcal{J}_r}\rangle$. The definition of $\models_\Sigma$ then gives us $\mathcal{J}_r, W \models_\Sigma (\lambda x!A.M : \Lambda x!A.B)\ [\Gamma]$;

$(M{:}A\&B)$ For the $\Rightarrow$ direction, by induction hypothesis twice we have the arrows

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket\pi_0(M)_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket A_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

and

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket\pi_1(M)_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$. Recall that we can construct products in each $\mathcal{J}_r(W)(D)$. So we have the arrow

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \overset{\llbracket\langle\pi_0(M),\pi_1(M)\rangle_\Gamma\rrbracket^W_{\mathcal{J}_r}}{\longrightarrow} \llbracket A\&B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$. For the $\Leftarrow$ direction, by induction hypothesis twice we have that $\mathcal{J}_r,W \models_\Sigma (\pi_0(M){:}A)\ [\Gamma]$ and $\mathcal{J}_r,W \models_\Sigma (\pi_1(M){:}B)\ [\Gamma]$. The definition of $\models_\Sigma$ then gives us $\mathcal{J}_r,W \models_\Sigma (M{:}A\&B)\ [\Gamma]$. □

The substitution lemma for $\models_\Sigma$ has two cases, one for substituting a linear variable and one for substituting an intuitionistic one.

Lemma 57 (Substitutivity of $\models_\Sigma$) *Let Σ be a signature and $\langle\{\mathcal{J}_r \mid r\in R\},\llbracket-\rrbracket, join, share,\models_\Sigma\rangle$ be a model.*

1. *If $\mathcal{J}_r,W \models_\Sigma (U{:}V)\ [\Delta,x{:}A,\Delta']$, $\mathcal{J}_s,W \models_\Sigma (N{:}A)\ [\Gamma]$ and $\llbracket\Delta,\Delta'[N/x]\rrbracket^W_{\mathcal{J}_{r'}}$ is defined, then $\mathcal{J}_t,W \models_\Sigma (U{:}V[N/x])\ [\Xi]$, where $\llbracket\Xi'\rrbracket^W_{\mathcal{J}_{s+r'}} = join(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_s},\llbracket\Delta,\Delta'[N/x]\rrbracket^W_{\mathcal{J}_{r'}})$ and $\llbracket\Xi\rrbracket^W_{\mathcal{J}_t} = share(\llbracket\Xi'\rrbracket^W_{\mathcal{J}_{s+r'}})$.*
2. *If $\mathcal{J}_r,W \models_\Sigma (U{:}V)\ [\Delta,x!A,\Delta']$, $\mathcal{J}_{r'},W \models_\Sigma (N{:}A)\ [!\Gamma]$ and $\llbracket\Delta,\Delta'[N/x]\rrbracket^W_{\mathcal{J}_{r'}}$ is defined, then $\mathcal{J}_{r'},W \models_\Sigma (U{:}V[N/x])\ [\Xi]$, where $\llbracket\Xi\rrbracket^W_{\mathcal{J}_{r'}} = join(\llbracket!\Gamma\rrbracket^W_{\mathcal{J}_{r'}},\llbracket\Delta,\Delta'[N/x]\rrbracket^W_{\mathcal{J}_{r'}})$.*

Proof By induction on the structure of the syntax and the functoriality of models.

1. The linear case is quite interesting as it shows an essential use of several of the model's components. In the following, we will omit the parameters on the interpretation for simplicity, though it can be seen, by induction, what these ought to be. Then, the basic argument is that, by the structure of the model, we can construct the following square in $\mathcal{C}_W$:

$$\begin{tikzcd}
(share\ join\langle\llbracket\Delta\rrbracket,\llbracket\Gamma\rrbracket\rangle) \arrow[r, "e"] \arrow[d, "f"'] & (share\ join\langle\llbracket\Delta\rrbracket,\llbracket\Gamma\rrbracket\rangle)\otimes\llbracket A\rrbracket \arrow[d, "g"] \\
(share\ join\langle\llbracket\Delta\rrbracket,\llbracket\Gamma\rrbracket\rangle)\bullet(\langle 1_{share\ join\langle\llbracket\Delta\rrbracket,\llbracket\Gamma\rrbracket\rangle},\llbracket N\rrbracket\rangle)^*\llbracket\Delta'\rrbracket \arrow[r, "h"'] & (share\ join\langle\llbracket\Delta\rrbracket,\llbracket\Gamma\rrbracket\rangle)\otimes\llbracket A\rrbracket\bullet\llbracket\Delta'\rrbracket
\end{tikzcd}$$

where

$$\begin{array}{rcl} e & = & \langle 1_{share\ join\langle [\![\Delta]\!],[\![\Gamma]\!]\rangle}, [\![N]\!]\rangle \\ f & = & \langle 1_{share\ join\langle [\![\Delta]\!],[\![\Gamma]\!]\rangle}, (\langle 1_{share\ join\langle [\![\Delta]\!],[\![\Gamma]\!]\rangle}, [\![N]\!]\rangle)^* d'\rangle \\ g & = & \langle\langle 1_{share\ join\langle [\![\Delta]\!],[\![\Gamma]\!]\rangle}, [\![N]\!]\rangle, d'\rangle \\ h & = & \langle\langle 1_{share\ join\langle [\![\Delta]\!],[\![\Gamma]\!]\rangle}, [\![N]\!]\rangle, 1_{[\![\Delta']\!]}\rangle \end{array}$$

and $1 \xrightarrow{d'} [\![\Delta']\!]$ is, by induction, an arrow in the fibre over $(share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle) \otimes [\![A]\!]$.

Then, by the functorial structure of the model, we have an arrow

$$1 \xrightarrow{(\langle\langle 1_{share\ join\langle [\![\Delta]\!],[\![\Gamma]\!]\rangle}, [\![N]\!]\rangle, 1_{[\![\Delta']\!]}\rangle)^* [\![U]\!]} (\langle\langle 1_{share\ join\langle [\![\Delta]\!],[\![\Gamma]\!]\rangle}, [\![N]\!]\rangle, 1_{[\![\Delta']\!]}\rangle)^* [\![V]\!]$$

in the fibre over the object $(share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle) \bullet (\langle 1_{share(\langle [\![\Delta]\!] \otimes [\![\Gamma]\!]\rangle)}, [\![N]\!]\rangle)^* [\![\Delta']\!]$.

2. The argument for the intuitionistic case is similar to the linear one, except that we use the pullback condition to extend the context with $[\![A]\!]$. □

4.4 Soundness and completeness

Lemma 58 (Context and Type Interpretations) *Let Σ be a signature and*

$$\langle \{\mathcal{J}_r \mid r \in R\}, [\![-]\!], join, share, \models_\Sigma \rangle$$

be a Kripke resource Σ-$\lambda\Lambda$ model.

1. *If* **N** *proves* $\vdash_\Sigma \Gamma$ context, *then, for those W where $[\![\Gamma]\!]^W_{\mathcal{J}_r}$ is defined, $[\![\Gamma]\!]^W_{\mathcal{J}_r} \in obj(C)$;*
2. *If* **N** *proves* $\Gamma \vdash_\Sigma A$:Type, *then, for those W where $[\![A_\Gamma]\!]^W_{\mathcal{J}_r}$ is defined, $[\![A_\Gamma]\!]^W_{\mathcal{J}_r} \in obj(\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r}))$.*

Proof Follows from Definition 53. The proofs are done by induction on the structure of proofs of system **N** and, because of inter-dependencies, must be done simultaneously with the proof of Theorem 59. □

Theorem 59 (Soundness) *Let Σ be a signature and $\langle \{\mathcal{J}_r \mid r \in R\}, [\![-]\!], join, share, \models_\Sigma \rangle$ be a Kripke resource model, and let W be any world in this model. If* **N** *proves $\Gamma \vdash_\Sigma M{:}A$ and $[\![\Gamma]\!]^W_{\mathcal{J}_r}$ is defined, $[\![A_\Gamma]\!]^W_{\mathcal{J}_r}$ is defined and $[\![M_\Gamma]\!]^W_{\mathcal{J}_r}$ is defined, then $\mathcal{J}_r, W \models_\Sigma (M{:}A)\ [\Gamma]$.*

Proof By induction on the structure of proofs of $\Gamma \vdash_\Sigma M{:}A$. The proof of soundness is done simultaneously with the proof of Lemma 58.

(Mc) Suppose $\mathbf{N}$ proves $!\Gamma \vdash_\Sigma c : \Lambda x_1{\in}A_1 .\dots \Lambda x_m{\in}A_m .A$. By Definition 53, $\{\mathcal{J}_r \mid r \in R\}$ has enough points to interpret $c : \Lambda x_1{\in}A_1 .\dots \Lambda x_m{\in}A_m .A$ and $\llbracket c_{!\Gamma} \rrbracket^W_{\mathcal{J}_r} = \Lambda^m(op_c)$ (*i.e.*, m applications of the natural isomorphism on op_c) where

$$1_{\mathcal{J}_r(W)(\llbracket !\Gamma, x_1 \in A_1, \dots, x_m \in A_m \rrbracket^W_{\mathcal{J}_r})} \xrightarrow{op_c} \llbracket A_{!\Gamma, x_1 \in A_1, \dots, x_m \in A_m} \rrbracket^W_{\mathcal{J}_r}$$

It can be observed that $\llbracket c_{!\Gamma} \rrbracket^W_{\mathcal{J}_r}$ type-checks. By induction, we have that $\llbracket !\Gamma \rrbracket^W_{\mathcal{J}_r}$ is defined. And so $\mathcal{J}_r, W \models_\Sigma (c : \Lambda x_1{\in}A_1 .\dots \Lambda x_m{\in}A_m .A)\ [!\Gamma]$ follows.

($MVar$) Suppose $\mathbf{N}$ proves $\Gamma, x{:}A \vdash_\Sigma x{:}A$ because $\Gamma \vdash_\Sigma A{:}\mathrm{Type}$. By induction, we have that $\llbracket \Gamma, x{:}A \rrbracket^W_{\mathcal{J}_r}$ is defined. According to Definition 53, $\llbracket x_{\Gamma, x:A} \rrbracket^W_{\mathcal{J}_r} = q_{\llbracket \Gamma, x:A \rrbracket^W_{\mathcal{J}_r}}$ and the latter has the correct type. So we have shown

$$1_{\mathcal{J}_r(W)(\llbracket \Gamma, x:A \rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket x_{\Gamma, x:A} \rrbracket^W_{\mathcal{J}_r}} \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_r}$$

and $\mathcal{J}_r, W \models_\Sigma (x{:}A)\ [\Gamma, x{:}A]$ follows.

($MVar!$) This case is done similarly to the ($MVar$) one above. Suppose $\mathbf{N}$ proves $\Gamma, x!A \vdash_\Sigma x{:}A$ because $\Gamma \vdash_\Sigma A{:}\mathrm{Type}$. By induction, we have that $\llbracket \Gamma, x!A \rrbracket^W_{\mathcal{J}_r}$ is defined and that $\llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_r}$ is defined. According to Definition 53, $\llbracket x_{\Gamma, x!A} \rrbracket^W_{\mathcal{J}_r} = q_{\llbracket \Gamma, x!A \rrbracket^W_{\mathcal{J}_r}}$ and the latter has the correct type. So have shown

$$1_{\mathcal{J}_r(W)(\llbracket \Gamma, x!A \rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket x_{\Gamma, x!A} \rrbracket^W_{\mathcal{J}_r}} \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_r}$$

and $\mathcal{J}_r, W \models_\Sigma (x{:}A)\ [\Gamma, x!A]$ follows.

($M\lambda\Lambda I$) Suppose $\mathbf{N}$ proves $\Gamma \vdash_\Sigma \lambda x{:}A .M : \Lambda x{:}A .B$ because $\Gamma, x{:}A \vdash_\Sigma M{:}B$. By induction, we have, for W such that $\llbracket B_{\Gamma, x:A} \rrbracket^W_{\mathcal{J}_r}$ is defined and that $\mathcal{J}_{r+s}, W \models_\Sigma (M{:}B)\ [\Gamma, x{:}A]$. *i.e.*, that

$$1_{\mathcal{J}_{r+s}(W)(\llbracket \Gamma, x:A \rrbracket^W_{\mathcal{J}_{r+s}})} \xrightarrow{\llbracket M_{\Gamma, x:A} \rrbracket^W_{\mathcal{J}_{r+s}}} \llbracket B_{\Gamma, x:A} \rrbracket^W_{\mathcal{J}_{r+s}}$$

We now use the natural isomorphism $\Lambda_{\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r}, \llbracket A_\Gamma \rrbracket^W_{\mathcal{J}_s}}$ to get

$$1_{\mathcal{J}_r(W)(\llbracket \Gamma \rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket \lambda x:A .M_\Gamma \rrbracket^W_{\mathcal{J}_r}} \llbracket \Lambda x{:}A .B_\Gamma \rrbracket^W_{\mathcal{J}_r}$$

So we obtain $\mathcal{J}_r, W \models_\Sigma (\lambda x{:}A .M : \Lambda x{:}A .B)\ [\Gamma]$.

$(M\Lambda\mathcal{E})$ Suppose $\mathbf{N}$ proves $\Xi \vdash_\Sigma MN{:}B[N/x]$ because $\Gamma \vdash_\Sigma M : \Lambda x{:}A.B$ and $\Delta \vdash_\Sigma N{:}A$ with $[\Xi';\Gamma;\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma,\Delta)$. By induction hypothesis twice we have that $\mathcal{J}_r, W \models_\Sigma (M : \Lambda x{:}A.B)\ [\Gamma]$, that is,

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket M_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket\Lambda x{:}A.B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

and $\mathcal{J}_s, W \models_\Sigma (N{:}A)\ [\Delta]$, that is,

$$1_{\mathcal{J}_s(W)(\llbracket\Delta\rrbracket^W_{\mathcal{J}_s})} \xrightarrow{\llbracket N_\Delta\rrbracket^W_{\mathcal{J}_s}} \llbracket A_\Delta\rrbracket^W_{\mathcal{J}_s}$$

Assume $W \leq W' \in \mathcal{W}$. By monotonicity and the definition of satisfaction, we have that $\mathcal{J}_t, W' \models_\Sigma (MN{:}B[N/x])\ [\Xi]$, where $\llbracket\Xi'\rrbracket^{W'}_{\mathcal{J}_{r+s}} = join\langle\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r}, \llbracket\Delta\rrbracket^{W'}_{\mathcal{J}_s}\rangle$ and $\llbracket\Xi\rrbracket^{W'}_{\mathcal{J}_t} = share(\llbracket\Xi'\rrbracket^{W'}_{\mathcal{J}_{s+r}})$, that is

$$1_{\mathcal{J}_t(W')(\llbracket\Xi\rrbracket^{W'}_{\mathcal{J}_t}} \xrightarrow{\llbracket MN_\Xi\rrbracket^{W'}_{\mathcal{J}_t}} \llbracket B[N/x]_\Xi\rrbracket^{W'}_{\mathcal{J}_t}$$

We check that this is the interpretation given by the model. According to Definition 53, $\llbracket MN_\Xi\rrbracket^W_{\mathcal{J}_t}$ is defined and is equal to, using monotonicity,

$$(\langle 1_{\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r}}, \llbracket N_\Delta\rrbracket^{W'}_{\mathcal{J}_s}\rangle^* (\Lambda_{\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r}, \llbracket A_\Gamma\rrbracket^{W'}_{\mathcal{J}_s}}(\llbracket M_\Gamma\rrbracket^{W'}_{\mathcal{J}_r})))$$

where Ξ is defined as above. We need to check the types. First, we already have

$$\llbracket M_\Gamma\rrbracket^{W'}_{\mathcal{J}_r} : 1_{\mathcal{J}_r(W')(\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r})} \longrightarrow \llbracket\Lambda x{:}A.B_\Gamma\rrbracket^{W'}_{\mathcal{J}_r}$$

Applying the natural isomorphism $\Lambda_{\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r}, \llbracket A_\Gamma\rrbracket^{W'}_{\mathcal{J}_s}}$ gives us

$$\Lambda_{\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_{r+s}}, \llbracket A_\Gamma\rrbracket^{W'}_{\mathcal{J}_{r+s}}}(\llbracket M_{\Gamma,x:A}\rrbracket^{W'}_{\mathcal{J}_{r+s}}) : 1_{\mathcal{J}_{r+s}(W')(\llbracket\Gamma,x:A\rrbracket^{W'}_{\mathcal{J}_{r+s}})} \longrightarrow \llbracket B_{\Gamma,x:A}\rrbracket^{W'}_{\mathcal{J}_{r+s}}$$

The functor $\langle 1_{\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r}}, \llbracket N_\Delta\rrbracket^{W'}_{\mathcal{J}_s}\rangle^*$ performs the required substitution. Finally the action of *join* and *share* gives us $\llbracket\Xi\rrbracket^{W'}_{\mathcal{J}_t}$.

$(M\lambda\Lambda!I)$ This case is done similarly to the $(M\lambda\Lambda I)$ one above. Suppose $\mathbf{N}$ proves $\Gamma \vdash_\Sigma \lambda x!A.M : \Lambda x!A.B$ because $\Gamma, x!A \vdash_\Sigma M{:}B$. By induction, we have, for W such that $\llbracket B_{\Gamma,x!A}\rrbracket^W_{\mathcal{J}_r}$ is defined, that $\mathcal{J}_r, W \models_\Sigma (M{:}B)\ [\Gamma, x!A]$. *i.e.*, that

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma,x!A\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket M_{\Gamma,x!A}\rrbracket^W_{\mathcal{J}_r}} \llbracket B_{\Gamma,x!A}\rrbracket^W_{\mathcal{J}_r}$$

We now use the natural isomorphism $\Lambda_{\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r}, \llbracket !A_\Gamma\rrbracket^W_{\mathcal{J}_r}}$ to get

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket\lambda x!A.M_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket\Lambda x!A.B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

So we obtain $\mathcal{J}_r, W \models_\Sigma (\lambda x!A.M : \Lambda x!A.B)\ [\Gamma]$.

($M\Lambda!\mathcal{E}$) This case is done similarly to the ($M\Lambda\mathcal{E}$) one above. Suppose **N** proves $\Xi \vdash_\Sigma MN{:}B[N/x]$ because $\Gamma \vdash_\Sigma M : \Lambda x!A\,.B$ and $!\Delta \vdash_\Sigma N{:}A$ with $[\Xi';\Gamma;!\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma, !\Delta)$. By induction hypothesis twice we have that $\mathcal{J}_r, W \models_\Sigma (M : \Lambda x!A\,.B)\ [\Gamma]$, that is,

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![M_\Gamma]\!]^W_{\mathcal{J}_r}} [\![\Lambda x!A\,.B_\Gamma]\!]^W_{\mathcal{J}_r}$$

and $\mathcal{J}_r, W \models_\Sigma (N{:}A)\ [!\Delta]$, that is,

$$1_{\mathcal{J}_r(W)([\![!\Delta]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![N_{!\Delta}]\!]^W_{\mathcal{J}_r}} [\![A_{!\Delta}]\!]^W_{\mathcal{J}_r}$$

Assume $W \leq W' \in \mathcal{W}$. By monotonicity and the definition of satisfaction, we have that $\mathcal{J}_r, W' \models_\Sigma (MN{:}B[N/x])\ [\Xi]$, where $[\![\Xi]\!]^{W'}_{\mathcal{J}_r} = share(join(\langle [\![\Gamma]\!]^{W'}_{\mathcal{J}_r}, [\![!\Delta]\!]^{W'}_{\mathcal{J}_r} \rangle))$, that is

$$1_{\mathcal{J}_r(W')([\![MN]\!]^{W'}_{\mathcal{J}_r}} \xrightarrow{[\![MN_\Xi]\!]^{W'}_{\mathcal{J}_r}} [\![B[N/x]_\Xi]\!]^{W'}_{\mathcal{J}_r}$$

We check that this is the interpretation given by the model. According to Definition 53, $[\![MN_\Xi]\!]^W_{\mathcal{J}_r}$ is defined and is equal to, using monotonicity,

$$(\langle 1_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r}}, [\![N_{!\Delta}]\!]^{W'}_{\mathcal{J}_r} \rangle^* (\Lambda_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r}, [\![A_\Gamma]\!]^{W'}_{\mathcal{J}_s}} ([\![M_\Gamma]\!]^{W'}_{\mathcal{J}_r})))$$

where $[\![\Xi]\!]^{W'}_{\mathcal{J}_r} = share(join\langle [\![\Gamma]\!]^{W'}_{\mathcal{J}_r}, [\![!\Delta]\!]^{W'}_{\mathcal{J}_r} \rangle)$. We need to check the types. First, we already have

$$[\![M_\Gamma]\!]^{W'}_{\mathcal{J}_r} : 1_{\mathcal{J}_r(W')([\![\Gamma]\!]^{W'}_{\mathcal{J}_r})} \longrightarrow [\![\Lambda x{:}A\,.B_\Gamma]\!]^{W'}_{\mathcal{J}_r}$$

Applying the natural isomorphism $\Lambda_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r}, [\![!A_\Gamma]\!]^{W'}_{\mathcal{J}_r}}$ gives us

$$\Lambda_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r}, [\![!A_\Gamma]\!]^{W'}_{\mathcal{J}_r}} ([\![M_\Gamma]\!]^{W'}_{\mathcal{J}_r}) : 1_{\mathcal{J}_r(W')([\![\Gamma, x!A]\!]^{W'}_{\mathcal{J}_r})} \longrightarrow [\![B_{\Gamma, x!A}]\!]^{W'}_{\mathcal{J}_r}$$

The functor $\langle 1_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r}}, [\![N_{!\Delta}]\!]^{W'}_{\mathcal{J}_r} \rangle^*$ performs the required substitution. Finally the action of *join* and *share* gives us $[\![\Xi]\!]^{W'}_{\mathcal{J}_r}$.

($M\&I$) Suppose **N** proves $\Gamma \vdash_\Sigma \langle M, N\rangle{:}A\&B$ because **N** proves $\Gamma \vdash_\Sigma M{:}A$ and **N** proves $\Gamma \vdash_\Sigma N{:}B$. By induction hypothesis twice, we have that $\mathcal{J}_r, W \models_\Sigma (M{:}A)\ [\Gamma]$, that is

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![M_\Gamma]\!]^W_{\mathcal{J}_r}} [\![A_\Gamma]\!]^W_{\mathcal{J}_r}$$

and that $\mathcal{J}_r, W \models_\Sigma (N{:}B)\ [\Gamma]$, that is

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket N_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

Now each category $\mathcal{J}_r(W)(D)$ in the model has products. We use this property in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$ to construct

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket\langle M,N\rangle_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket(A\&B)_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

and $\mathcal{J}_r, W \models_\Sigma (\langle M,N\rangle{:}(A\&B))\ [\Gamma]$ follows.

$(M\&\mathcal{E}_i)$ Suppose **N** proves $\Gamma \vdash_\Sigma \pi_i(M){:}A_i$, for $i \in \{0,1\}$ because **N** proves $\Gamma \vdash_\Sigma M{:}A_0\&A_1$. By induction hypothesis, we have that $\mathcal{J}_r, W \models_\Sigma (M{:}A_0\&A_1)\ [\Gamma]$, that is,

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket M_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket(A_0\&A_1)_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

Then the definition of satisfaction allows us to construct, for $i \in \{0,1\}$,

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket\pi_i(M)_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket A_{i\Gamma}\rrbracket^W_{\mathcal{J}_r}$$

and $\mathcal{J}_r, W \models_\Sigma (\pi_i(M){:}A_i)\ [\Gamma]$ follows.

$(M \equiv)$ It is convenient, as we are working in the $M{:}A$-fragment of the type theory, to observe that $\beta\eta$-equalities are generated by the rule

$$\frac{\Gamma, x{\in}A \vdash_\Sigma M{:}B \quad \Delta \vdash_\Sigma N{:}A}{\Xi \vdash_\Sigma (\lambda x{\in}A\,.M)N =_\beta M[N/x]\ :\ B[N/x]}$$

where $[\Xi';\Gamma;\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma,\Delta)$, and by the rule

$$\frac{\Gamma \vdash_\Sigma M\ :\ \Lambda x{\in}A\,.B}{\Gamma \vdash_\Sigma (\lambda y{\in}A\,.M)y =_\eta M\ :\ \Lambda x{\in}A\,.B}$$

where $y \notin FV(\Gamma, x{\in}A)$. Then, an application of the natural isomorphism and Lemma 57, allows us to show that if $M =_{\beta\eta} N$, then $\llbracket M\rrbracket^W_{\mathcal{J}_r} \simeq \llbracket N\rrbracket^W_{\mathcal{J}_r}$. □

We now turn to consider completeness. We begin with the appropriate definition of validity for $\models_\Sigma$.

Definition 60 (**$\models_\Sigma$-validity for $\lambda\Lambda$**) $\Gamma \models_\Sigma M{:}A$, i.e., *$M{:}A$ is valid with respect to Γ, if and only if, for all models $\langle\{\mathcal{J}_r \mid r \in R\}, [\![-]\!], join, share, \models_\Sigma\rangle$ and all worlds W such that $[\![\Gamma]\!]^W_{\mathcal{J}_r}$, $[\![A_\Gamma]\!]^W_{\mathcal{J}_r}$ and $[\![M_\Gamma]\!]^W_{\mathcal{J}_r}$ are defined, $\mathcal{J}_r, W \models_\Sigma (M{:}A)\ [\Gamma]$.* □

Several components of the term model will be defined using the category of contexts and realizations, $C(\Sigma)$, that we defined previously.

Proposition 61 *$C(\Sigma)$ is a doubly monoidal category.*

Proof The two context extension operators are taken to be an extension with A and an extension with $!A$. The units for each context extension operator require the following rules to be taken in the syntax of the type theory:

$$\frac{}{\vdash_\Sigma 1 \text{ context}} \qquad \frac{}{\vdash_\Sigma I \text{ context}}$$

together with the context equivalences which let I and 1 be, respectively, units of extension with A and extension with $!A$. Then we define $\langle\rangle = I;1$, allowing us to continue with the use of $\langle\rangle$ as a general unit. □

Definition 62 *The category $\mathcal{P}(\Sigma)$, a full sub-category of $C(\Sigma)$, is defined as follows:*

- *Objects:*
  - *$\langle\rangle$ is an object of $\mathcal{P}(\Sigma)$;*
  - *If Γ is an object of $\mathcal{P}(\Sigma)$ and there exists an arrow $\Gamma \overset{\langle 1,M\rangle}{\to} \Gamma \times A$ in $C(\Sigma)$, then $\Gamma \times A$ is an object of $\mathcal{P}(\Sigma)$.*
- *Morphisms: The arrows just considered.*

Lemma 63 *The tuple consisting of the set of objects in $C(\Sigma)$, the context joining operation $[-;-;-]$ and the unit context $\langle\rangle$ defines a commutative monoid.*

Proof For ease of argument, we will adopt the following notation: $\Gamma\Delta$ will denote the join of the contexts Γ and Δ.

We first show that $\langle\rangle$ behaves as a 2-sided identity. This is immediate due to the coherence equivalences between contexts.

Next, we show that the joining relation is associative: if Γ, Δ and Θ are valid contexts, then $\Gamma(\Delta\Theta) = (\Gamma\Delta)\Theta$, where $[\Gamma(\Delta\Theta);\Gamma;\Delta\Theta]$, $[\Delta\Theta;\Delta;\Theta]$, $[(\Gamma\Delta)\Theta;\Gamma\Delta;\Theta]$ and $[\Gamma\Delta;\Gamma;\Delta]$.

The proof of associativity is by induction on the length of the context $\Gamma(\Delta\Theta)$. The base case is when $\Gamma(\Delta\Theta) = \langle\rangle$. By the definition of the joining relation, this implies that $\Gamma = \langle\rangle$ and that $\Delta\Theta = \langle\rangle$. By the same argument, we know that $\Delta = \langle\rangle$ and $\Theta = \langle\rangle$. We use the definition to construct $(\Gamma\Delta)\Theta$ which is also equal to $\langle\rangle$.

There are three inductive cases to consider, one for each of the (JOIN-L), (JOIN-R) and (JOIN-!) rules. For the first of these, we have $\Gamma(\Delta\Theta) = \Gamma(\Delta(\Theta',x{:}A))$ by (JOIN-L). By assumption, $\Gamma(\Delta(\Theta',x{:}A))$ splits into Γ and $\Delta(\Theta',x{:}A)$, and $\Delta(\Theta',x{:}A)$ splits into Δ and $\Theta',x{:}A$. By induction hypothesis, Γ and Δ join to form $\Gamma\Delta$, and $\Gamma\Delta$ and Θ' join to form $(\Gamma\Delta)\Theta'$. By (JOIN-L), $(\Gamma\Delta)\Theta'$ and $x{:}A$ join to form $(\Gamma\Delta)\Theta',x{:}A = (\Gamma\Delta)\Theta$. The other two cases are argued similarly and we omit the details.

Lastly, we show the commutativity of the joining relation: if $[\Xi;\Gamma;\Delta]$, then $[\Xi;\Delta;\Gamma]$. The proof is by induction on the length of the context Ξ. For the base case, when $\Xi = \langle\rangle$, the proof is immediate. There are three inductive cases to consider, one for each of the (JOIN-L), (JOIN-R) and (JOIN-!) rules. For the first of these, suppose $[\Xi',x{:}A;\Gamma,x{:}A;\Delta]$. By induction hypothesis, we have that if $[\Xi';\Gamma;\Delta]$, then $[\Xi;\Delta;\Gamma]$. Then an application of (JOIN-R) gives us $[\Xi',x{:}A;\Delta;\Gamma,x{:}A]$. The other two cases are argued similarly and we omit the details. □

As joining is associative, we can informally say "Γ, Δ and Θ join to form $\Gamma\Delta\Theta$". That is, we can talk about n-way joining and there need be no confusion.

We note that in logics, such as intuitionistic logic or **BI**, which include conjunctions and disjunctions, one must develop the notion of prime theory. Prime theories have exactly the structure required by the semantic clauses for the connectives and are used to prove completeness. The construction of prime theories is not necessary in the minimal cases of both the $\lambda\Pi$- and $\lambda\Lambda$-calculi, where the function spaces are the only connectives. (The $\lambda\Lambda$-calculus does have the additive conjunction, but the term model inherits enough structure from the syntax to push the definitions through.)

Lemma 64 (Model Existence) *There is a Kripke Σ-$\lambda\Lambda$ model*

$$\langle\{\mathcal{T}(\Sigma)_\Delta\}, [\![-]\!]^{-}_{\mathcal{T}(\Sigma)_\Delta}, join, share, \models_\Sigma\rangle$$

with a world W_0 such that if $\Gamma \not\vdash_\Sigma M{:}A$, then $\mathcal{T}(\Sigma)_\Delta, W_0 \not\models_\Sigma (M{:}A)\ [\Gamma]$

Proof We construct such a model out of the syntax of the λΛ-calculus.

The Kripke Σ-λΛ structure $\mathcal{T}(\Sigma)_\Delta$ is defined as follows. The category of worlds is taken to be $\mathcal{P}(\Sigma)$. The base category is the co-product of $C(\Sigma)_\Delta$, where each $\Delta \in ob(\mathcal{P}(\Sigma))$. The indexing monoid is given by the context joining relation $[-;-;-]$, as defined by Lemma 63. The functor $\mathcal{T}(\Sigma)_\Delta$, indexed by an element $\Delta \in obj(C(\Sigma))$, is defined as follows:

$$\mathcal{T}(\Sigma)_\Delta(\Theta)(\Gamma) \quad = \quad \begin{cases} \mathit{Objects} & \text{Types } A \text{ such that } \mathbf{N} \text{ proves } \Psi \vdash_\Sigma A{:}\mathrm{Type} \\ \mathit{Arrows} & \mathcal{E}(\Sigma)(\Psi) \text{ arrows} \end{cases}$$
$$\text{where } [\Phi';\Theta;\Delta],\ \Phi = \Phi' \backslash \kappa(\Theta,\Delta),$$
$$\text{and } [\Psi';\Gamma;\Phi],\ \Psi = \Psi' \backslash \kappa(\Gamma,\Phi).$$

It is easy to see that $\mathcal{T}(\Sigma)_\Theta$ is a functor. Suppose $\Delta \xrightarrow{f} \Delta'$, $\Delta' \xrightarrow{f'} \Delta''$, $\Gamma \xrightarrow{g} \Gamma'$ and $\Gamma' \xrightarrow{g'} \Gamma''$. For identities, if $\mathcal{T}(\Sigma)_\Theta(\Delta)(\Gamma) = \mathcal{A}$, then the corresponding functor in **Cat** to $\mathcal{T}(\Sigma)_\Theta(id_\Delta)(id_\Gamma)$ is given by the identity one on $\mathcal{A}$. For composition, by induction hypothesis twice we have that $\mathcal{T}(\Sigma)_\Theta(\Delta)(\Gamma) \xrightarrow{\mathcal{T}(\Sigma)_\Theta(f)(g)} \mathcal{T}(\Sigma)_\Theta(\Delta')(\Gamma')$ and that $\mathcal{T}(\Sigma)_\Theta(\Delta')(\Gamma') \xrightarrow{\mathcal{T}(\Sigma)_\Theta(f')(g')} \mathcal{T}(\Sigma)_\Theta(\Delta')(\Gamma')$. And we just compose these two functors in **Cat** to give us the required answer.

We next check that $\mathcal{T}(\Sigma)_\Theta$ is a Kripke structure.

1. The terminal object in each $\mathcal{T}(\Sigma)_\Theta(\Delta)(\Gamma)$ is taken to be the unit additive context 1. We choose this as the proof theory has the judgement **N** proves $\Gamma \vdash_\Sigma 1$ so that 1 always exists in each fibre. 1 contains no free variables and so is always preserved on the nose by any f^*;

2. The two extensions, $D \to D \otimes A$ and $D \to D \times A$, are given by the context extension rules of the type theory.

   The first projection map for an intuitionistically extended context

   $$\Gamma = x_1{\in}A_1, \ldots, x_n{\in}A_n, x_{n+1}!A_{n+1}$$

   is defined by $p(\Gamma) = x_1{\in}A_1, \ldots, x_n{\in}A_n$. This is well-defined as weakening is admissible in the syntax. The map $q_{\langle\Gamma,A\rangle}$ is given by the term x where $\Gamma, x{\in}A \vdash_\Sigma x{:}A$.

   For the first projection map, we need to check that the appropriate square is a pullback. Let $\Gamma = x_1{\in}A_1, \ldots, x_n{\in}A_n$, $\Delta = y_1{\in}B_1, \ldots, y_m{\in}B_m$ and $\Delta, y_{m+1}!B_{m+1}$ be contexts, and let $t = \langle t_1, \ldots, t_m\rangle{:}\Gamma \to \Delta$ be a morphism. We need to show that

$$\begin{tikzcd}
\Gamma, t^*(y_{m+1}!B_{m+1}) \arrow[r, "\langle t, B_{m+1}\rangle"] \arrow[d, "p_{\Gamma,t^*(B_{m+1})}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \Delta, y_{m+1}!B_{m+1} \arrow[d, "p_{\Delta,B_{m+1}}"] \\
\Gamma \arrow[r, "t"'] & \Delta
\end{tikzcd}$$

So let Θ be a context and let

$$a = \langle a_1, \dots, a_n\rangle : \Theta \to \Gamma$$

and

$$b = \langle b_1, \dots, b_{m+1}\rangle : \Theta \to \Delta, y_{m+1}!B_{m+1}$$

be morphisms such that $a;t = b; p_{\Delta,B_{m+1}}$. *i.e.*, for all $1 \leq i \leq m$ b_i and $t_i[a_j/x_j]_{j=1}^n$ are equal terms of type $B_i[a_j/x_j]_{j=1}^n$. Then we define the mediating arrow

$$\alpha = \langle a_1, \dots, a_n, b_{n+1}\rangle : \Theta \to \Gamma, t^*(y_{m+1}!B_{m+1}),$$

as b_{m+1} type-checks.

We need to show that $\alpha; p_{\Gamma,B_{m+1}} = a$ and $\alpha; \langle t, B_{m+1}\rangle = b$.

$$\begin{aligned}
\alpha; p_{\Gamma,B_{m+1}} &= \langle x_1[a_1/x_1, \dots, a_n/x_n, b_{m+1}/x_{m+1}], \dots, \\
&\qquad x_n[a_1/x_1, \dots, a_n/x_n, b_{m+1}/x_{m+1}]\rangle \\
&= \langle a_1, \dots, a_n\rangle \\
&= a
\end{aligned}$$

$$\begin{aligned}
\alpha; \langle t, B_{m+1}\rangle &= \langle t_1[a_1/x_1, \dots, a_n/x_n, b_{m+1}/x_{m+1}], \dots, \\
&\qquad t_m[a_1/x_1, \dots, a_n/x_n, b_{m+1}/x_{m+1}], \\
&\qquad x_{m+1}[a_1/x_1, \dots, a_n/x_n, b_{m+1}/x_{m+1}]\rangle \\
&= \langle t_1[a_1/x_1, \dots, a_n/x_n], \dots, t_m[a_1/x_1, \dots, a_n/x_n], b_{m+1}\rangle \\
&= b
\end{aligned}$$

We also need to show that α is unique. So suppose

$$\beta = \langle \beta_1, \dots, \beta_{n+1}\rangle : \Theta \to \Gamma, t^*(y_{m+1}!B_{m+1})$$

is another mediating arrow. *i.e.*, $\beta; p_{\Gamma,B_{m+1}} = a$ and $\beta;\langle t, B_{m+1}\rangle = b$. Then we get the equations

$$\begin{aligned}
\beta; p_{\Gamma,B_{m+1}} &= \langle x_1[\beta_j/x_j]_{j=1}^{n+1}, \ldots, x_n[\beta_j/x_j]_{j=1}^{n+1}\rangle \\
&= \langle \beta_1, \ldots, \beta_n\rangle
\end{aligned}$$

$$\begin{aligned}
\beta;\langle t, B_{m+1}\rangle &= \langle t_1[\beta_j/x_j]_{j=1}^{n+1}, \ldots, t_m[\beta_j/x_j]_{j=1}^{n+1}, x_{n+1}[\beta_j/x_j]_{j=1}^{n+1}\rangle \\
&= \langle t_1[\beta_j/x_j]_{j=1}^{n+1}, \ldots, t_m[\beta_j/x_j]_{j=1}^{n+1}, \beta_{n+1}\rangle
\end{aligned}$$

Now $\beta = \langle \beta_1, \ldots, \beta_{n+1}\rangle$ and by the above two equations we have

$$\langle \beta_1, \ldots, \beta_n\rangle = \langle a_1, \ldots, a_n\rangle$$

and

$$\langle t_1[\beta_j/x_j]_{j=1}^{n+1}, \ldots, t_m[\beta_j/x_j]_{j=1}^{n+1}, \beta_{n+1}\rangle = \langle b_1, \ldots, b_{m+1}\rangle\ .$$

So we have $\langle \beta_1, \ldots, \beta_{n+1}\rangle = \langle a_1, \ldots, a_n, b_{m+1}\rangle = \alpha$.

Lastly, we need to check the strictness conditions. Let Γ, Δ and Θ be contexts, let $\Gamma \xrightarrow{t} \Delta$ and $\Delta \xrightarrow{s} \Theta$ be morphisms, and let $\mathbf{N}$ prove $\Theta \vdash_\Sigma A{:}\mathrm{Type}$. Then we have:

$$\begin{aligned}
(1_\Gamma)^* A &= A[1_\Gamma] \\
&= A
\end{aligned}$$

$$\begin{aligned}
1_\Gamma \bullet 1_A &= \langle \vec{x}\rangle \bullet \langle A\rangle \\
&= \langle \vec{x}, A\rangle \\
&= 1_{\Gamma \bullet A}
\end{aligned}$$

$$\begin{aligned}
t^*(s^* A) &= t^*(A[s]) \\
&= A[s[t]] \\
&= A[t;s] \\
&= (t;s)^*(A)
\end{aligned}$$

$$\begin{aligned}
(\langle t, s^*(A)\rangle; \langle s, A\rangle)\langle B, C\rangle &= (\langle t, A[s]\rangle; \langle s, A\rangle)\langle B, C\rangle \\
&= \langle s, A\rangle\langle t(B), C\rangle \\
&= \langle s(t(B)), C\rangle \\
&= \langle s;t, A\rangle\langle B, C\rangle
\end{aligned}$$

where, in the last equation, we assume that **N** proves $\Gamma \vdash_\Sigma B{:}\mathsf{Type}$ and **N** proves $\Theta \vdash_\Sigma C{:}\mathsf{Type}$.

3. The natural isomophism is given by the abstraction and application rules of the type theory:

$$\frac{\Gamma, x{\in}A \vdash_\Sigma M{:}B}{\overline{\Gamma \vdash_\Sigma \lambda x{\in}A\,.M : \Lambda x{\in}A\,.B}}$$

where $x{\in}A$, recall, ranges over both linear $x{:}A$ and intuitionistic $x!A$ declarations. We need to check that these do meet the Beck-Chevalley condition:

$$\begin{aligned}
&f^*(\Lambda_{\Gamma,A}(\Gamma, x{\in}A \vdash_\Sigma M{:}B))\\
&\quad= f^*(\Gamma \vdash_\Sigma \lambda x{\in}A\,.M{:}\Lambda x{\in}A\,.B)\\
&\quad= (\Delta \vdash_\Sigma (\lambda x{\in}A\,.M{:}\Lambda x{\in}A\,.B)[f])\\
&\quad= \Delta \vdash_\Sigma ((\lambda x{\in}A\,.M)[f]{:}(\Lambda x{\in}A\,.B)[f])\\
&\quad= \Delta \vdash_\Sigma \lambda x{\in}A[f]\,.M[f]{:}\Lambda x{\in}A[f]\,.B[f]\\
&\quad= \Delta \vdash_\Sigma \lambda x{\in}A[f]\,.M[f,x{\in}A]{:}\Lambda x{\in}A[f]\,.B[f,x{\in}A]\\
&\quad= \Delta \vdash_\Sigma \lambda x{\in}f^*(A)\,.(f,x{\in}A)^*(M){:}\Lambda x{\in}f^*(A)\,.(f,x{\in}A)^*(B)\\
&\quad= \Lambda_{\Delta,f^*(A)}(\Delta, x{\in}f^*(A) \vdash_\Sigma (f,x{\in}A)^*(M){:}(f,x{\in}A)^*(B))\\
&\quad= \Lambda_{\Delta,f^*(A)}(\Delta, x{\in}f^*(A) \vdash_\Sigma (f,x{\in}A)^*(M{:}B))\\
&\quad= \Lambda_{\Delta,f^*(A)}((f,x{\in}A)^*(\Gamma,x{\in}A) \vdash_\Sigma (f,x{\in}A)^*(M{:}B))\\
&\quad= \Lambda_{\Delta,f^*(A)}((f,x{\in}A)^*(\Gamma,x{\in}A \vdash_\Sigma M{:}B))
\end{aligned}$$

4. The products in each $\mathcal{T}(\Sigma)_\Theta(\Delta)(\Gamma)$ are given by the $(M\&I)$ and $(M\&\mathcal{E}_i)$ rules.

The model is defined as follows. $\mathcal{T}(\Sigma)_\Delta$ is the Kripke $\lambda\Lambda$-structure defined above. The Σ-operations of the model are given by the constants declared in the signature Σ. The interpretation $[\![-]\!]^{-}_{\mathcal{T}(\Sigma)\_}$ is the obvious one in which a term (type) is interpreted by the class of terms definitionally equivalent to the term (type) in the appropriate component of $\mathcal{T}(\Sigma)$. The functors *join* and *share* are defined by the joining relation $[-;-;-]$ and κ, respectively.

The satisfaction relation $\models_\Sigma$ in $\mathcal{T}(\Sigma)$ is given by provability in the type theory. That is, $\mathcal{T}(\Sigma)_\Theta, \Delta \models_\Sigma (M{:}A)\ [\Gamma]$ is defined to be $\Xi \vdash_\Sigma M{:}A$, where Ξ is the sharing-sensitive join of Θ, Δ and Γ. We must check that this relation satisfies the inductive clauses of the satisfaction relation:

1. $!\Xi \vdash_\Sigma c{:}A$ if and only if $c{:}A \in \Sigma$ is immediate as the Σ-operations are the $c{:}A$s ;

2. $\Xi, x{:}A \vdash_\Sigma x{:}A$ if and only if $\vdash_\Sigma \Xi, x{:}A$ context by induction on the structure of proofs of both hypotheses ;

3. $\Xi \vdash_\Sigma M{:}\Lambda x{:}A.B$ if and only if $\Phi \vdash_\Sigma N{:}A$ implies $\Psi \vdash_\Sigma MN{:}B[N/x]$, where $[\Psi';\Xi;\Phi]$ and $\Psi = \Psi' \backslash \kappa(\Xi,\Phi)$, holds, in one direction, by $(M\Lambda\mathcal{E})$ and, in the other, by an application of Cut. The intuitionistic case is similar ;

4. $\Xi \vdash_\Sigma M{:}A\&B$ if and only if $\Xi \vdash_\Sigma \pi_0(M){:}A$ and $\Xi \vdash_\Sigma \pi_1(M){:}B$ is immediate by the & rules.

The conditions on the models are met as follows:

1. Monotonicity is met by the fact that all terms are well-defined. *i.e.*, constructed in accordance with the proof rules. so a valid term will only ever be constructed from valid sub-terms; and

2. Accessibility is provided by the posetal nature of $\mathcal{P}(\Sigma)$.

From Theorem 49, we have that $\mathcal{T}(\Sigma)_\Theta, \Delta \models_\Sigma (M{:}A)\ [\Gamma]$ if and only if $\Xi \vdash_\Sigma M{:}A$, where Ξ is the sharing-sensitive join of Θ, Δ and Γ.

We can now finish the proof of model existence. We assume the premiss, that $\Gamma \not\vdash_\Sigma M{:}A$. Then, at the initial node ($W_0 = \langle\rangle$), the model constructed from the syntax has the required property; that $\mathcal{T}(\Sigma)_\Theta, W_0 \not\models_\Sigma (M{:}A)\ [\Phi]$, where $[\Gamma;\Theta;\Phi]$. □

Theorem 65 (Completeness) $\Gamma \vdash_\Sigma M{:}A$ *if and only if* $\Gamma \models_\Sigma M{:}A$.

Proof Theorem 59 (Soundness) shows the forward direction. For the other, we assume $\Gamma \not\vdash_\Sigma M{:}A$ and apply Lemma 64. □

4.5 A class of set-theoretic models

We describe a class of concrete Kripke resource models, in which the $\lambda\Lambda$-structure

$$\{\mathcal{J}_r{:}[\mathcal{W},[C^{op},\mathbf{Cat}]] \mid r \in R\}$$

is given by $\mathbf{BIFam}{:}[C,[Ctx^{op},\mathbf{Set}]]$, where C is a small monoidal category and Ctx is a small, set-theoretic category of "contexts". The model is a construction on the category of families of sets and exploits Day's tensor product construction [Day70] to define the linear dependent function space.

We start by describing the indexed category of families of sets, $\mathbf{Fam}{:}[Ctx^{op},\mathbf{Cat}]$. The base, Ctx, is a small, set-theoretic category defined inductively as follows. The objects of Ctx, called "contexts", are (*i.e.*, their denotations are) sets and the arrows of Ctx, called "realizations", are set-theoretic functions. For each $D \in obj(Ctx)$, $\mathbf{Fam}(D) = \{y \in B(x) \mid x \in D\}$. The fibre can be described as a discrete category whose objects are the ys and whose arrows are the maps $1_y{:}y \to y$ corresponding to the identity functions $id{:}\{y\} \to \{y\}$ on y considered as a singleton set. If $E \xrightarrow{f} D$ is an arrow in Ctx, then $\mathbf{Fam}(f) = f^*{:}\mathbf{Fam}(D) \to \mathbf{Fam}(E)$ re-indexes the set $\{y \in B(x) \mid x \in D\}$ over D to the set $\{f(z) \in B(f(z)) \mid z \in E\}$ over E. We are viewing **Set** as **Cat**; each object of **Set** is seen as an object, a discrete category, in **Cat**. Because of this, the category of families of sets can just be considered as a presheaf $\mathbf{Fam}{:}[Ctx^{op}, \mathbf{Set}]$, rather than as an indexed category; we will adopt this view in the sequel.

We can explicate the structure of Ctx by describing **Fam** as a contextual category [Car86]. The following definition is from Streicher [Str88]. The contextual category **Fam**, together with its length and denotation DEN:**Fam** $\to$ **Set**, is described as follows:

1. 1 is the unique context of length 0 and

$$\mathrm{DEN}(1) = \{\emptyset\}$$

2. If D is a context of length n and $A{:}\mathrm{DEN}(D) \to \mathbf{Set}$ is a family of sets indexed by elements of $DEN(D)$, then $D \times A$ is a context of length $n+1$ and

$$\mathrm{DEN}(D \times A) = \{\langle x, y\rangle \mid x \in \mathrm{DEN}(D),\ y \in A(x)\}$$

If D and E are objects of the contextual category **Fam**, then the morphisms between them are simply the functions between $\mathrm{DEN}(D)$ and $\mathrm{DEN}(E)$.

The codomain of the denotation, **Set**, allows the definition of an extensional context extension $\times$. But **Set** does not have enough structure to define an intensional context extension $\otimes$. (The obvious candidate, such as some tuple-based construction, is really just special kinds of cartesian product and inherits the $\times$'s structural properties. It may be that **Dom** has enough structure to define such a product.) In order to be able to define both $\times$ and $\otimes$, we denote **Fam** not in **Set** but in a presheaf $\mathbf{Set}^{C^{op}}$, where C is a monoidal category. We emphasize that, in general, C can be any monoidal category and, therefore, we are actually going to describe a class of set-theoretic models. For simplicity, we take C^{op} to be a partially-ordered commutative monoid $\mathcal{M} = (M, \cdot, e, \sqsubseteq)$. The cartesian structure on the presheaf gives us the $\times$ context extension and a restriction of Day's tensor product [Day70] gives us the $\otimes$ context extension.

We note that the restriction of Day's tensor product that we consider is merely this: consider the set-theoretic characterization of Day's tensor product as tuples $\langle x,y,f\rangle$ and, of all those tuples, consider only those where the y is an element of the family of sets in x. This is quite concrete, in the spirit of the Cartmell-Streicher models, and is not to be considered a general construction for a fibred Day product.

Within the contextual setting, we then have the following definition. The contextual category **BIFam**, together with its length and denotation DEN:**BIFam** $\rightarrow$ **Set**$^{\mathcal{M}}$, is described as follows:

1. 1 is a context of length 0 and
$$\mathrm{DEN}(1)(Z) = \{\emptyset\}$$
2. I is a context of length 0 and
$$\mathrm{DEN}(I)(-) = \mathcal{M}[-,I]$$
3. If D is a context of length n and A:DEN$(D)(X) \rightarrow$ **Set**$^{\mathcal{M}}$ is a family of $\mathcal{M}$-sets indexed by elements of $DEN(D)(X)$, then
   (a) $D \times A$ is a context of length $n+1$ and
   $$\mathrm{DEN}(D \times A)(X) = \{\langle x,y\rangle \mid x \in \mathrm{DEN}(D)(X),\ y \in (A(x))(X)\}$$
   and
   (b) $D \otimes A$ is a context of length $n+1$ and
   $$\mathrm{DEN}(D \otimes A)(Z) = \{\langle x,y,f\rangle \in \int^{X,Y} \mathrm{DEN}(D)(X) \times (A(x))(Y) \times \mathcal{M}[Z, X \otimes Y]\}$$
   The $\otimes$ extension is defined using co-ends. Here we have used the characterization of Day's tensor product as tuples, with the restriction, to account for dependency, of the triples $\langle x,y,f\rangle$ to those in which $y \in A(x)(Y)$.

If D and E are objects of **BIFam**, then the morphisms between them are the functions between DEN$(D)(X)$ and DEN$(E)(Y)$. **BIFam** is **Fam** parametrized by $\mathcal{M}$; objects that were interpreted in **Set** are now interpreted in **Set**$^{\mathcal{M}}$.

Now consider **BIFam** in an indexed setting. By our earlier argument relating indexed and contextual presentations of families of sets, **BIFam** can be seen as a functor category **BIFam**:$[Ctx^{op}, \mathbf{Set}^{\mathcal{M}}]$. This is not quite the presheaf setting we require. However, if we calculate

$$\begin{aligned}[Ctx^{op}, [\mathbf{Set}^{\mathcal{M}}]] &\cong [Ctx^{op} \times \mathcal{M}, \mathbf{Set}] \\ &\cong [\mathcal{M} \times Ctx^{op}, \mathbf{Set}] \\ &\cong [\mathcal{M}, [Ctx^{op}, \mathbf{Set}]]\end{aligned}$$

then this restores the indexed setting and also reiterates the idea that $\mathcal{M}$ parametrizes **Fam**.

Lastly, we say what the R and $\mathcal{W}$ components of the concrete model are. Define $(R, +, 0) = (M, \cdot, e)$ and define $(\mathcal{W}, \leq) = (M/\sim, \sqsubseteq)$, where the quotient of M by the relation $w \sim w \cdot w$ is necessary because of the separation of worlds from resources (*cf.* **BI**'s semantics [OP99, Urq72]). This allows us to define $\mathcal{J}_r(w) = \mathbf{BIFam}(r \cdot w)$. The quotiented $\mathcal{M}$ maintains the required properties of monotonicity and bifunctoriality of the internal logic forcing relation. It is then easy to check that $\mathbf{BIFam}(r \cdot w)$ does simulate $\mathcal{J}_r(w)$.

We now check that $\mathbf{BIFam}$:$[\mathcal{M}, [Ctx^{op}, \mathbf{Set}]]$ is a Kripke resource $\lambda\Lambda$-structure and that it can be used to define a Kripke resource Σ-$\lambda\Lambda$-model.

Lemma 66 $\mathbf{BIFam}$:$[\mathcal{M}, [Ctx^{op}, \mathbf{Set}]]$ *is a Kripke resource* $\lambda\Lambda$*-structure.*

Proof Recall that we view **Set** as **Cat**. Each of the following points refers to those of Definition 51:

1. The terminal object in each fibre is taken to be the one-point set $\{\emptyset\}$. As this is not indexed by any variables, it will always be preserved on the nose by any f^* ;

2. If $D \in obj(Ctx)$ and A is a set indexed by $x \in \mathbf{BIFam}(Z)(D)$, then the two context extensions are given by $D \times A$ and $D \otimes A$. The second projection $q_{D,A}$ is simply the element-hood of the set A. The first projection $p_{D,A}$ is defined for the object $D \times A$ by $p_{D,A}\langle x, y\rangle = x$. The first projection cannot be defined for the object $D \otimes A$ as the elements of the denotation of $D \otimes A$ do not consist of pairs. The pullback and other coherences need to be checked but these are very similar to the calculations for the term model, so we omit them ;

3. The natural isomorphism Λ is defined for each type of context extension as follows. For the $\otimes$ extension, for any set B indexed by the denotation of $D \otimes A$ (write this as $B_{D\otimes A}$), we define $curry(B_{D\otimes A}) = \Lambda_A B_D$ where, for any $x \in \mathbf{BIFam}(X)(D)$, $\Lambda_A B$ is defined as

$$\{f{:}\mathbf{BIFam}(Y)(A(x)) \to \cup_y \{\mathbf{BIFam}(X \otimes Y)(B(x,y)) \mid y \in \mathbf{BIFam}(Y)(A(x))\}$$
$$\mid \forall a \in \mathbf{BIFam}(Y)(A(x))\ f(a) \in \mathbf{BIFam}(X \otimes Y)(B(x,a))\}$$

For the $\times$ extension, for any set B indexed by the denotation of $D \times A$ (write this as $B_{D\times A}$), we define $curry(B_{D\times A}) = \Pi_A B_D$ where, for any $x \in \mathbf{BIFam}(X)(D)$, $\Pi_A B$ is defined as

$$\{g{:}\mathbf{BIFam}(X)(A(x)) \to \cup_y \{\mathbf{BIFam}(X)(B(x,y)) \mid y \in \mathbf{BIFam}(X)(A(x))\}$$
$$\mid \forall a \in \mathbf{BIFam}(X)(A(x))\ f(a) \in \mathbf{BIFam}(X)(B(x,a))\}$$

The inverse functor is just application and defined as follows. For the $\otimes$ case, for any $x \in \mathbf{BIFam}(X)(D)$ and $y \in \mathbf{BIFam}(Y)(A(x))$ and $f \in \Lambda_A(B)$,

$$uncurry\ f \quad = \quad f(y)$$

For the $\times$ case, for any $x \in \mathbf{BIFam}(X)(D)$ and $y \in \mathbf{BIFam}(X)(A(x))$ and $g \in \Pi_A(B)$,

$$uncurry\ g \quad = \quad g(y)$$

We need to check that *curry* and *uncurry* are isomorphisms. We further need to check the Beck-Chevalley condition. But these are fairly simple calculations about abstraction, application and substitution in **Set**, with just the additional, resource parametrization, and we omit them.

We comment that the description of the function space corresponds to an indexed version of Day's function space. And that, though we only require a natural isomorphism for our lemma, we have described the adjoints to $\otimes$ and $\times$ in $\mathbf{Set}^{\mathcal{M}}$;

4. The product of two objects $M \times N$ in each fibre is taken to be the cartesian product of the corresponding singleton sets $\{M\} \times \{N\}$. □

Lemma 67 **BIFam** *can be extended to a Kripke resource Σ-$\lambda\Lambda$-model.*

Proof **BIFam** is defined as follows: $\mathbf{BIFam}(s)(D)$ is the family of D-indexed sets at s. That **BIFam** is a functor arises due to a combination of the bifunctoriality of $\sqsubseteq$ and the functoriality of the presheaf $[Ctx^{op}, \mathbf{Set}]$.

The definition of a model requires the structure to have enough points to interpret the constants of the signature. We can work with an arbitrary signature and interpret constants and variables as the functors $Const{:}\mathcal{M} \to \mathbf{Set}$ and $D{:}\mathcal{M} \to \mathbf{Set}$ respectively.

The interpretation function $[\![-]\!]^r_{\mathbf{BIFam}}$ is parametrized over worlds-resources r. The interpretation of contexts is defined following the same idea as the construction of the category *Ctx*:

1. $[\![\Gamma, x{:}A]\!]^{r+r'}_{\mathbf{BIFam}} \simeq [\![\Gamma]\!]^{r}_{\mathbf{BIFam}} \otimes [\![A\Gamma]\!]^{r'}_{\mathbf{BIFam}}$;

2. $[\![\Gamma, x!A]\!]^{r}_{\mathbf{BIFam}} \simeq [\![\Gamma]\!]^{r}_{\mathbf{BIFam}} \times [\![A\Gamma]\!]^{r}_{\mathbf{BIFam}}$.

The interpretation of functions is defined using the *curry* functor and the interpretation of & is defined using the product in each set.

The interpretation function is defined simultaneously with the instances of the functors *join* and *share* on *Ctx*. The definition of these are similar to those for the joining relation $[-;-;-]$ and the sharing function κ of the term model, and we omit the details.

Satisfaction is a relation over M and $[Ctx^{op}, \mathbf{Set}]$ with the clauses reflecting the properties of the example model:

1. $m \models_\Sigma f{:}\Lambda x{:}A\,.B\ [D]$ if and only if $n \models a{:}A\ [E]$ implies $m \cdot n \models_\Sigma f(a){:}B[a/x]\ [D \otimes E]$;

2. $m \models_\Sigma f{:}\Pi x{:}A\,.B\ [D]$ if and only if $m \models_\Sigma a{:}A\ [E]$ implies $m \models_\Sigma f(a){:}B[a/x]\ [D \times E]$;

3. $m \models_\Sigma M{:}A\&B\ [D]$ if and only if $m \models_\Sigma \pi_i(M){:}A\ [D]$ and $m \models_\Sigma \pi_i(M){:}A\ [D]$, where $i \in \{0,1\}$.

The conditions on the model are met due to the bifunctoriality of $\sqsubseteq$ and the definition of the interpretation function. □

This concludes the definition of the example model.

It was simpler to work with an unspecified, arbitrary signature. However, if we consider a signature which encoded the judgements of an imperative programming language (as in Chapter 2), so that we had such Σ-operations in each fibre, then we conjecture that **BIFam** provides a basis for a good model for an imperative programming language. In particular, we think that **BIFam** would model the behaviour of the store quite finely. The basis for this conjecture lies in work which uses the internal logic to reason about local-global issues in state-ful programming. For instance, O'Hearn has a nice example of treating the two context extension operators as different kinds of storage allocators [O'H99a].

4.6 Summary

This chapter began with the primitive notion of "resource", as presented in **BI**'s analysis. We generalized this for a type-theoretic presentation and defined the notion of Kripke resource models. These are a monoid-indexed set of Kripke functors, in which the indexing monoid is seen as providing an account of resource consumption. We showed that these models are sound and complete for the $\lambda\Lambda$-calculus. We discussed one interesting concrete Kripke resource model, which was constructed using families of sets.

Chapter 5

The Gentzenized λΛ-calculus

5.1 Introduction

The λΛ-calculus is a first order linear dependent type theory, parameterized by κ, a function which implements sharing. The type theory is defined as a system of linearized natural deduction. In this chapter, we present a Gentzen-style sequent calculus for the λΛ-calculus and prove the cut-elimination theorem for it.

The main difference between the natural deduction system, hereinafter referred to as system **N**, and the Gentzen-style sequent calculus, hereinafter referred to as system **G**, is that the elimination rules are replaced with left rules, which introduce the connective on the left hand side of the sequent. The systems **N** and **G** are equivalent, provided that the system **G** includes the structural rule of cut. As a simple example, consider the natural deduction presentation of propositional linear logic, in which the rules for ⊸ are

$$\frac{\Gamma,\phi\vdash\psi}{\Gamma\vdash\phi\multimap\psi}\,(\multimap I) \qquad \frac{\Gamma\vdash\phi\multimap\psi \quad \Delta\vdash\phi}{\Gamma,\Delta\vdash\psi}\,(\multimap E)$$

In a Gentzen-style sequent calculus, the $(\multimap E)$ rule is replaced by

$$\frac{\Gamma\vdash\phi \quad \psi,\Delta\vdash\xi}{\Gamma,\Delta,\phi\multimap\psi\vdash\xi}\,(\multimap L)$$

and the cut rule

$$\frac{\Gamma\vdash\phi \quad \Delta,\phi\vdash\psi}{\Gamma,\Delta\vdash\psi}$$

is included in the sequent calculus. The Gentzenization of a predicate type theory basically follows the same pattern, though there are a clutch of side-conditions relating to capture-avoiding substitution and definitional equality.

In this chapter, we will follow the technique of Pym [Pym95], where the intuitionistic λΠ-calculus is Gentzenized. With this reference, we remark on the form of the left rules presented in the sequel. In the λΠ-calculus, the left rule for Π is formulated as follows:

$$\frac{\Gamma \vdash_\Sigma N{:}B \quad \Gamma, x{:}D \vdash_\Sigma M{:}A}{\Gamma, y{:}\Pi z{:}B.C \vdash_\Sigma M[yN/x]{:}A}\ (\Pi L)$$

with the side-conditions that y is new and $D =_\beta C[N/z]$. In the intuitionistic λΠ-calculus, the structural rules of weakening and contraction are admissible. A consequence of this is that the above left rule can also be expressed as follows, with a contraction on the principal formula:

$$\frac{@{:}\Pi z{:}B.C \in \Sigma \cup \Gamma \quad \Gamma \vdash_\Sigma N{:}B \quad \Gamma, x{:}D \vdash_\Sigma M{:}A}{\Gamma \vdash_\Sigma M[@N/x]{:}A}\ (\Pi L^C)$$

with the side-conditions that $y \notin FV(A)$ and $D =_\beta C[N/z]$. This rule has the advantage that it has the sub-formula property and hence yields a resolution (or backchaining) rule, thus providing a suitable basis for logic programming [PW91] . Obviously, in weaker Gentzen systems, such as the one considered in the sequel, the two formulations — (ΠL) and (ΠL^C) — are not equivalent. Hence, the rules we consider will be analogues of the first formulation, though we are mindful of its disadvantages.

The premiss that $@{:}\Pi z{:}B.C \in \Sigma \cup \Gamma$ is due to the fact that cut-elimination relates normalized proofs to cut-free proofs in the sequent calculus. A consequence of this is that cut-elimination obtains for β-normal forms only. We expand on this remark in § 5.4.

5.2 The Gentzenized λΛ-calculus

In this section, we present the system **G**, a presentation of the λΛ-calculus as sequent calculus, in which the elimination rules for Λ, Π and & of the system **N** are replaced by left rules. For the function spaces, we deal only with the dependent cases (Λ, Π) and omit the treatment of the (simpler) non-dependent ones ($\multimap$, $\rightarrow$).

The case for the additive conjunction is a fairly standard one. The elimination rule

$$\frac{\Gamma \vdash_\Sigma M{:}A_0\&A_1}{\Gamma \vdash_\Sigma \pi_i(M){:}A_i}$$

is replaced by the following left rule

$$\frac{\Gamma, x{:}A_i \vdash_\Sigma M{:}B}{\Gamma, y{:}A_0\&A_1 \vdash_\Sigma (M{:}B)[\pi_i(y)/x]}$$

We will omit the treatment of & in the sequel but refer to Girard, Lafont and Taylor [GLT89], and Troelstra and Schwichtenberg [TS96] for the standard treatment.

The combinatorics of the two types of declarations (linear and intuitionistic) and the two types of dependent function spaces (linear and intuitionistic) leads us to consider the following four left rules for the dependent function spaces:

$$\frac{\Gamma \vdash_\Sigma N{:}B \quad \Delta, x{:}D \vdash_\Sigma M{:}A \quad (*)}{\Xi, y{:}\Lambda z{:}B.C \vdash_\Sigma M[yN/x]{:}A}\ (:\Lambda L) \qquad \frac{!\Gamma \vdash_\Sigma N{:}B \quad \Delta, x{:}D \vdash_\Sigma M{:}A \quad (*)}{\Xi, y{:}\Lambda z!B.C \vdash_\Sigma M[yN/x]{:}A}\ (:\Pi L)$$

$$\frac{\Gamma \vdash_\Sigma N{:}B \quad \Delta, x!D \vdash_\Sigma M{:}A \quad (*)}{\Xi, y!\Lambda z{:}B.C \vdash_\Sigma M[yN/x]{:}A}\ (!\Lambda L) \qquad \frac{!\Gamma \vdash_\Sigma N{:}B \quad \Delta, x!D \vdash_\Sigma M{:}A \quad (*)}{\Xi, y!\Lambda z!B.C \vdash_\Sigma M[yN/x]{:}A}\ (!\Pi L)$$

Table 5.1: The left rules in **G**

where $(*)$ denotes the side-condition that $x \notin FV(A)$, y new, $D \equiv C[N/z]$, $[\Xi'; \Gamma; (!)\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma, (!)\Delta)$.

We make a couple of comments regarding the row and column axes of the above square. Firstly, we can see the first row (*i.e.*, the rules $(:\Lambda L)$ and $(:\Pi L)$) as prior to the second row, in the sense that the first row can be derelicted to get the second row. Evidence for this point of view will be seen in the nature of the proofs of soundness and completeness in the sequel. It can also be seen by observing that if x is an intuitionistic variable, then so is y. This is because of the relation between C and D: D is C with all the occurrences of z in the latter replaced by N. We remark that the first row being prior to the second row also refers to the part of dereliction in the propositions-as-types correspondence and in the nature of the categorical models of the internal logic of the λΛ-calculus.

The second point to make is that in the second column (*i.e.*, the rules $(:\Pi L)$ and $(!\Pi L)$), the intuitionistic minor premiss $!\Gamma \vdash_\Sigma N{:}B$ allows us to form an intuitionistic function space. The reason for this is that the premiss derelicts $N{:}B$, allowing z, the abstracted variable of the function space, to be an intuitionistic variable.

The following two cut rules, one for linear and one for intuitionistic variables, are included in the system **G**:

$$\frac{\Gamma \vdash_\Sigma N{:}A \quad \Delta, x{:}A, \Delta' \vdash_\Sigma U{:}V \ \ (*)}{\Xi \vdash_\Sigma (U{:}V)[N/x]}\,(Cut) \qquad \frac{!\Gamma \vdash_\Sigma N{:}A \quad \Delta, x!A, \Delta' \vdash_\Sigma U{:}V \ \ (*)}{\Xi \vdash_\Sigma (U{:}V)[N/x]}\,(Cut!)$$

where $(*)$ denotes the side-condition that $[\Xi';\Gamma;\Delta,\Delta'[N/x]]$ and $\Xi = \Xi' \backslash \kappa(\Gamma, (\Delta, \Delta'[N/x]))$

Theorem 68 (Structural Admissibilities in G) *The following structural rules are admissible:*

1. Exchange*: If $\Gamma, x{\in}A, y{\in}B, \Delta \vdash_\Sigma U{:}V$, then $\Gamma, y{\in}B, x{\in}A, \Delta \vdash_\Sigma U{:}V$, provided $x \notin FV(B)$, $y \notin FV(A)$ and $\Gamma \vdash_\Sigma B{:}\mathrm{Type}$;*
2. Weakening*: If $\Gamma \vdash_\Sigma U{:}V$ and $!\Delta \vdash_\Sigma A{:}\mathrm{Type}$, then $\Xi, x!A \vdash_\Sigma U{:}V$, where $[\Xi;\Gamma;!\Delta]$;*
3. Dereliction*: If $\Gamma, x{:}A \vdash_\Sigma U{:}V$, then $\Gamma', x!A \vdash_\Sigma U{:}V$, where Γ' is Γ in which the free variables of A have been derelicted too;*
4. Contraction*: If $\Gamma, x!A, y!A \vdash_\Sigma U{:}V$, then $\Gamma, x!A \vdash_\Sigma (U{:}V)[x/y]$.*

Proof By induction on the structure of the proof of the premisses. The proof is essentially that for system **N** and we omit the details. □

5.3 Soundness and completeness

In this section, we prove the soundness and completeness of **G** with respect to **N**. The results crucially depend on the presence of the two cut rules in **G**.

Theorem 69 (Soundness of G) *If **G** proves $\Gamma \vdash_\Sigma U{:}V$, then **N** proves $\Gamma \vdash_\Sigma U{:}V$.*

Proof By induction on the structure of the proofs in **G**. As the two cut rules are admissible in **N**, the only additional, and difficult, cases are those of the four left rules.

$(:\Lambda L)$ Suppose the last rule of **G** applied is $(:\Lambda L)$

$$\frac{\begin{matrix}\pi_0\\ \vdots\\ \Gamma \vdash_\Sigma N{:}B\end{matrix} \quad \begin{matrix}\pi_1\\ \vdots\\ \Delta, x{:}D \vdash_\Sigma M{:}A\end{matrix} \quad (\dagger)}{\Xi, y{:}\Lambda z{:}B.C \vdash_\Sigma M[yN/x]{:}A}\,(:\Lambda L),$$

where (†) is the side-condition that $x \notin FV(A)$, y new, $D \equiv C[N/z]$, $[\Xi';\Gamma;\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma,\Delta)$.

By induction hypothesis, we have that **N** proves $\Gamma \vdash_\Sigma N{:}B$ and that **N** proves $\Delta, x{:}D \vdash_\Sigma M{:}A$. By induction on the structure of the proof the second of these, we obtain **N** proves $\Psi \vdash_\Sigma \Lambda z{:}B.C{:}\mathrm{Type}$, for some $\Psi \subseteq \Delta$. We apply (MVar) to this to obtain **N** proves $\Psi, y{:}\Lambda z{:}B.C \vdash_\Sigma y{:}\Lambda z{:}B.C$. We then apply $(M\Lambda\mathcal{E})$ to **N** proves $\Psi, y{:}\Lambda z{:}B.C \vdash_\Sigma y{:}\Lambda z{:}B.C$ and **N** proves $\Gamma \vdash_\Sigma N{:}B$ to obtain **N** proves $\Phi, y{:}\Lambda z{:}B.C \vdash_\Sigma yN{:}C[N/z]$, where $[\Phi';\Psi;\Gamma]$ and $\Phi = \Phi' \backslash \kappa(\Psi,\Gamma)$. From this and **N** proves $\Delta, x{:}D \vdash_\Sigma M{:}A$ we obtain $\Xi, y{:}\Lambda z{:}B.C \vdash_\Sigma M[yN/x]{:}A$, where $[\Xi';\Phi;\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Phi,\Delta)$, by the (Cut) rule.

The argument is summarized by the following deduction in **N**:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\text{IH on }\pi_1\\ \vdots\\ \Psi \vdash_\Sigma \Lambda z{:}B.C\ \mathrm{Type}\end{array}}{\Psi, y{:}\Lambda z{:}B.C \vdash_\Sigma y{:}\Lambda z{:}B.C}\,(\mathit{MVar}) \quad \begin{array}{c}\text{IH on }\pi_0\\ \vdots\\ \Gamma \vdash_\Sigma N{:}B\end{array} \quad (\dagger)}{\Phi, y{:}\Lambda z{:}B.C \vdash_\Sigma yN{:}C[N/z]}\,(M\Lambda\mathcal{E}) \qquad \begin{array}{c}\text{IH on }\pi_1\\ \vdots\\ \Delta, x{:}D \vdash_\Sigma M{:}A\end{array} \quad (\ddagger)}{\Xi, y{:}\Lambda z{:}B.C \vdash_\Sigma M[yN/x]{:}A}\,(\mathit{Cut})$$

where the side-conditions (†) and (‡) denote the ones given in the argument.

$(:\Pi L)$ Suppose the last rule of **G** applied is $(:\Pi L)$:

$$\dfrac{\begin{array}{c}\pi_0\\ \vdots\\ !\Gamma \vdash_\Sigma N{:}B\end{array} \quad \begin{array}{c}\pi_1\\ \vdots\\ \Delta, x{:}D \vdash_\Sigma M{:}A\end{array} \quad (\dagger)}{\Xi, y{:}\Lambda z!B.C \vdash_\Sigma M[yN/x]{:}A}\,(:\Pi L),$$

where (†) is the side-condition that $x \notin FV(A)$, y new, $D \equiv C[N/z]$, $[\Xi';!\Gamma;\Delta]$ and $\Xi = \Xi' \backslash \kappa(!\Gamma,\Delta)$.

By induction hypothesis, we have that **N** proves $!\Gamma \vdash_\Sigma N{:}B$ and that **N** proves $\Delta, x{:}D \vdash_\Sigma M{:}A$. By induction on the structure of the proof the second of these, we obtain **N** proves $\Psi \vdash_\Sigma \Lambda z!B.C{:}\mathrm{Type}$, for some $\Psi \subseteq \Delta$. We apply $(\mathit{MVar}!)$ to this to obtain **N** proves $\Psi, y{:}\Lambda z!B.C \vdash_\Sigma y{:}\Lambda z!B.C$. We then apply $(M\Lambda!\mathcal{E})$ to **N** proves $\Psi, y{:}\Lambda z!B.C \vdash_\Sigma y{:}\Lambda z!B.C$ and **N** proves $!\Gamma \vdash_\Sigma N{:}B$ to obtain **N** proves $\Phi, y{:}\Lambda z!B.C \vdash_\Sigma yN{:}C[N/z]$, where $[\Phi';\Psi;!\Gamma]$ and $\Phi = \Phi' \backslash \kappa(\Psi,!\Gamma)$. We note that it might be necessary to weaken Ψ for the join to work properly. From this and **N** proves $\Delta,x{:}D\vdash_\Sigma M{:}A$ we obtain $\Xi,y{:}\Lambda z!B.C\vdash_\Sigma M[yN/x]{:}A$, where $[\Xi';\Phi;\Delta]$ and $\Xi=\Xi'\backslash\kappa(\Phi,\Delta)$, by the (Cut) rule.

The argument is summarized by the following deduction in **N**:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\text{IH on }\pi_1\\ \vdots\\ \Psi\vdash_\Sigma \Lambda z!B.C\ \text{Type}\end{array}}{\Psi,y{:}\Lambda z!B.C\vdash_\Sigma y{:}\Lambda z!B.C}\,(MVar)\qquad \begin{array}{c}\text{IH on }\pi_0\\ \vdots\\ !\Gamma\vdash_\Sigma N{:}B\end{array}\quad(\dagger)}{\Phi,y{:}\Lambda z!B.C\vdash_\Sigma yN{:}C[N/z]}\,(M\Lambda!\mathcal{E})\qquad \begin{array}{c}\text{IH on }\pi_1\\ \vdots\\ \Delta,x{:}D\vdash_\Sigma M{:}A\end{array}\quad(\ddagger)}{\Xi,y{:}\Lambda z!B.C\vdash_\Sigma M[yN/x]{:}A}\,(Cut)$$

where the side-conditions $(\dagger)$ and $(\ddagger)$ denote the ones given in the argument.

$(!\Lambda L)$ Suppose the last rule of **G** applied is $(!\Lambda L)$

$$\dfrac{\begin{array}{c}\pi_0\\ \vdots\\ \Gamma\vdash_\Sigma N{:}B\end{array}\quad \begin{array}{c}\pi_1\\ \vdots\\ \Delta,x!D\vdash_\Sigma M{:}A\end{array}\quad(\dagger)}{\Xi,y!\Lambda z{:}B.C\vdash_\Sigma M[yN/x]{:}A}\,({:}\,\Pi L),$$

where $(\dagger)$ is the side-condition that $x\notin FV(A)$, y new, $D\equiv C[N/z]$, $[\Xi';\Gamma;\Delta]$ and $\Xi=\Xi'\backslash\kappa(\Gamma,\Delta)$.

By induction hypothesis, we have that **N** proves $\Gamma\vdash_\Sigma N{:}B$ and that **N** proves $\Delta,x!D\vdash_\Sigma M{:}A$. By induction on the structure of the proof the second of these, we obtain **N** proves $\Phi\vdash_\Sigma \Lambda z{:}B.C{:}\text{Type}$, for some $\Phi\subseteq\Delta$. We apply $(MVar)$ to this to obtain **N** proves $\Phi,y{:}\Lambda z{:}B.C\vdash_\Sigma y{:}\Lambda z{:}B.C$. We then apply $(M\Lambda\mathcal{E})$ to **N** proves $\Phi,y{:}\Lambda z{:}B.C\vdash_\Sigma y{:}\Lambda z{:}B.C$ and **N** proves $\Gamma\vdash_\Sigma N{:}B$ to obtain **N** proves $\Psi,y{:}\Lambda z{:}B.C\vdash_\Sigma yN{:}C[N/z]$, where $[\Psi';\Phi;\Gamma]$ and $\Psi=\Psi'\backslash\kappa(\Phi,\Gamma)$. We note that it might be necessary to weaken Φ and Γ for the join to work properly. From this and **N** proves $\Delta,x!D\vdash_\Sigma M{:}A$ we obtain $\Xi,y{:}\Lambda z{:}B.C\vdash_\Sigma M[yN/x]{:}A$, where $[\Xi';\Psi;\Delta]$ and $\Xi=\Xi'\backslash\kappa(\Psi,\Delta)$, by the $(Cut!)$ rule, using the admissibility of dereliction where necessary for the applicability of the cut.

The argument is summarized by the following deduction in **N**:

$$\dfrac{\dfrac{\dfrac{\dfrac{\begin{array}{c}\text{IH on }\pi_1\\ \vdots\\ \Phi\vdash_\Sigma \Lambda z{:}B.C{:}\text{Type}\end{array}}{\Phi,y{:}\Lambda z{:}B.C\vdash_\Sigma y{:}\Lambda z{:}B.C}\,(MVar)\qquad \begin{array}{c}\text{IH on }\pi_0\\ \vdots\\ \Gamma\vdash_\Sigma N{:}B\end{array}\quad(\dagger)}{\Psi,y{:}\Lambda z{:}B.C\vdash_\Sigma yN{:}C[N/z]}\,(M\Lambda\mathcal{E})}{!\Psi,y!\Lambda z{:}B.C\vdash_\Sigma yN{:}C[N/z]}\,(D)\qquad \begin{array}{c}\text{IH on }\pi_1\\ \vdots\\ \Delta,x!D\vdash_\Sigma M{:}A\end{array}\quad\ddagger}{\Xi,y!\Lambda z{:}B.C\vdash_\Sigma M[yN/x]{:}A}\,(Cut!)$$

where † and ‡ are the side-conditions mentioned in the argument.

$(!\Pi L)$ This case amounts to the one for the (ΠE) case in the $\lambda\Pi$-calculus. Suppose the last rule of **G** applied is $(!\Pi L)$:

$$\dfrac{\begin{array}{c}\pi_0\\ \vdots\\ !\Gamma \vdash_\Sigma N{:}B\end{array} \quad \begin{array}{c}\pi_1\\ \vdots\\ \Delta, x{!}D \vdash_\Sigma M{:}A\end{array} \quad (\dagger)}{\Xi, y{:}\Lambda z{:}B.C \vdash_\Sigma M[yN/x]{:}A}\,(!\Pi L),$$

where $(\dagger)$ is the side-condition that $x \not\in FV(A)$, y new, $D \equiv C[N/z]$, $[\Xi'; !\Gamma; \Delta]$ and $\Xi = \Xi' \backslash \kappa(!\Gamma, \Delta)$.

By induction hypothesis, we have that **N** proves $!\Gamma \vdash_\Sigma N{:}B$ and that **N** proves $\Delta, x{!}D \vdash_\Sigma M{:}A$. By induction on the structure of the proof the second of these, we obtain **N** proves $\Psi \vdash_\Sigma \Lambda z{!}B.C{:}\mathsf{Type}$, for some $\Psi \subseteq \Delta$. We apply $(MVar!)$ to this to obtain **N** proves $\Psi, y{:}\Lambda z{!}B.C \vdash_\Sigma y{:}\Lambda z{!}B.C$. We then apply $(M\Lambda{!}\mathcal{E})$ to **N** proves $\Psi, y{:}\Lambda z{!}B.C \vdash_\Sigma y{:}\Lambda z{!}B.C$ and **N** proves $!\Gamma \vdash_\Sigma N{:}B$ to obtain **N** proves $\Phi, y{:}\Lambda z{!}B.C \vdash_\Sigma yN{:}C[N/z]$, where $[\Phi'; \Psi; !\Gamma]$ and $\Phi = \Phi' \backslash \kappa(\Psi, !\Gamma)$. We note that it might be necessary to weaken Ψ and Γ for the join to work properly. From this and **N** proves $\Delta, x{!}D \vdash_\Sigma M{:}A$ we obtain $\Xi, y{:}\Lambda z{!}B.C \vdash_\Sigma M[yN/x]{:}A$, where $[\Xi'; \Phi; !\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Phi, \Delta)$, by the $(Cut!)$ rule, using the admissibility of dereliction where necessary for the applicability of the cut.

The argument is summarized by the following deduction in **N**:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\text{IH on } \pi_1\\ \vdots\\ \Psi \vdash_\Sigma \Lambda z{!}B.C\ \mathsf{Type}\end{array}}{\Psi, y{!}\Lambda z{!}B.C \vdash_\Sigma y{:}\Lambda z{!}B.C}\,(MVar!) \quad \begin{array}{c}\text{IH on } \pi_0\\ \vdots\\ !\Gamma \vdash_\Sigma N{:}B\end{array} \quad (\dagger)}{\Phi, y{!}\Lambda z{!}B.C \vdash_\Sigma yN{:}C[N/z]}\,(M\Lambda{!}\mathcal{E}) \quad \begin{array}{c}\text{IH on } \pi_1\\ \vdots\\ \Delta, x{!}D \vdash_\Sigma M{:}A\end{array} \quad (\ddagger)}{\Xi, y{!}\Lambda z{!}B.C \vdash_\Sigma M[yN/x]{:}A}\,(Cut!)$$

where the side-conditions $(\dagger)$ and $(\ddagger)$ denote the ones given in the argument. □

Theorem 70 (Completeness of G) *If* **N** *proves* $\Gamma \vdash_\Sigma U{:}V$, *then* **G** *proves* $\Gamma \vdash_\Sigma U{:}V$.

Proof By induction on the structure of proofs in system **N**. The majority of the cases are simple inductive arguments. The only difficult ones are those for the two dependent function space elimination rules, $(M\Lambda\mathcal{E})$ and $(M\Lambda{!}\mathcal{E})$, which we consider.

$(M\Lambda\mathcal{E})$ Suppose the last rule of **N** applied is the $(M\Lambda\mathcal{E})$ rule:

$$\dfrac{\begin{array}{c}\pi_0\\ \vdots\\ \Gamma\vdash_\Sigma M{:}\Lambda x{:}A\,.B\end{array}\quad\begin{array}{c}\pi_1\\ \vdots\\ \Delta\vdash_\Sigma N{:}A\end{array}\quad(\dagger)}{\Xi\vdash_\Sigma MN{:}B[N/x]}\,(M\Lambda\mathcal{E})$$

where $(\dagger)$ is the side-condition that $[\Xi';\Gamma;\Delta]$ and $\Xi=\Xi'\backslash\kappa(\Gamma,\Delta)$. By the induction hypothesis, we have that **G** proves $\Gamma\vdash_\Sigma M{:}\Lambda x{:}A\,.B$ and that **G** proves $\Delta\vdash_\Sigma N{:}A$. By induction on the structure of the proofs of these two assertions in **G** we obtain that **G** proves $\Phi,x{:}A\vdash_\Sigma B(x){:}\mathrm{Type}$, for some $\Phi\subset\Gamma$, and **G** proves $\Psi\vdash_\Sigma A{:}\mathrm{Type}$, for some $\Psi\subset\Delta$.

We then apply the $(MVar)$ rule twice: from **G** proves $\Psi\vdash_\Sigma A{:}\mathrm{Type}$, we obtain **G** proves $\Psi,x{:}A\vdash_\Sigma x{:}A$; and from **G** proves $\Phi,x{:}A\vdash_\Sigma B(x){:}\mathrm{Type}$ we obtain **G** proves $\Phi,x{:}A,z{:}B(x)\vdash_\Sigma z{:}B(x)$. From these two assertions, by an application of $(:\Lambda L)$, we obtain **G** proves $\Phi\Psi,x{:}A,y{:}\Lambda x{:}A\,.B\vdash_\Sigma yx{:}B(x)$, where $\Phi\Psi$ is the κ-sensitive join of Φ and Ψ.

From **G** proves $\Phi\Psi,x{:}A,y{:}\Lambda x{:}A\,.B\vdash_\Sigma yx{:}B(x)$ and **G** proves $\Delta\vdash_\Sigma N{:}A$, we obtain $\Delta\Phi,y{:}\Lambda x{:}A\,.B\vdash_\Sigma yN{:}B[N/x]$ by an application of the (Cut) rule, where $\Delta\Phi$ is the κ-sensitive join of Δ and $\Phi\Psi$. And from this last assertion and **G** proves $\Gamma\vdash_\Sigma M{:}\Lambda x{:}A\,.B$, we obtain $\Xi\vdash_\Sigma MN{:}B[N/x]$ by another application of the (Cut) rule, where Ξ is the κ-sensitive join of Γ and $\Delta\Phi$.

The above argument is summarized by the following proof figure:

$$\dfrac{\begin{array}{c}\text{IH on }\pi_0\\ \vdots\\ \Gamma\vdash_\Sigma M{:}\Lambda x{:}A\,.B\end{array}\quad\dfrac{\begin{array}{c}\text{IH on }\pi_1\\ \vdots\\ \Delta\vdash_\Sigma N{:}A\end{array}\quad\dfrac{\dfrac{\begin{array}{c}\text{IH on }\pi_1\\ \vdots\\ \Psi\vdash_\Sigma A{:}\mathrm{Type}\end{array}}{\Psi,x{:}A\vdash_\Sigma x{:}A}\,(MVar)\quad\dfrac{\begin{array}{c}\text{IH on }\pi_0\\ \vdots\\ \Phi,x{:}A\vdash_\Sigma B(x){:}\mathrm{Type}\end{array}}{\Phi,x{:}A,z{:}B(x)\vdash_\Sigma z{:}B(x)}\,(MVar)\quad(\dagger)}{\Phi\Psi,x{:}A,y{:}\Lambda x{:}A\,.B\vdash_\Sigma yx{:}B(x)}\,(:\Lambda L)\quad(\dagger\dagger)}{\Delta\Phi,y{:}\Lambda x{:}A\,.B\vdash_\Sigma yN{:}B[N/x]}\,(Cut)\quad(\dagger\dagger\dagger)}{\Xi\vdash_\Sigma MN{:}B[N/x]}\,(Cut)$$

where $(\dagger)$, $(\dagger\dagger)$ and $(\dagger\dagger\dagger)$ are the side-conditions mentioned in the argument.

$(M\Lambda!\mathcal{E})$ Suppose the last rule of **N** applied is the $(M\Lambda!\mathcal{E})$ rule:

$$\dfrac{\begin{array}{c}\pi_0\\ \vdots\\ \Gamma\vdash_\Sigma M{:}\Lambda x!A\,.B\end{array}\quad\begin{array}{c}\pi_1\\ \vdots\\ !\Delta\vdash_\Sigma N{:}A\end{array}\quad(\dagger)}{\Xi\vdash_\Sigma MN{:}B[N/x]}\,(M\Lambda!\mathcal{E})$$

where $(\dagger)$ is the side-condition that $[\Xi';\Gamma;\Delta]$ and $\Xi = \Xi' \backslash \kappa(\Gamma,\Delta)$. By the induction hypothesis, we have that **G** proves $\Gamma \vdash_\Sigma M{:}\Lambda x!A\,.B$ and that **G** proves $!\Delta \vdash_\Sigma N{:}A$. By induction on the structure of the proofs of these two assertions in **G**we obtain that **G** proves $\Phi, x!A \vdash_\Sigma B(x){:}\mathsf{Type}$, for some $\Phi \subset \Gamma$, and **G** proves $\Psi \vdash_\Sigma A{:}\mathsf{Type}$, for some $\Psi \subset \Delta$.

We then apply the variable introduction rules. From **G** proves $\Psi \vdash_\Sigma A{:}\mathsf{Type}$ we obtain **G** proves $\Psi, x!A \vdash_\Sigma x{:}A$ by the $(MVar!)$ rule. And from **G** proves $\Phi, x!A \vdash_\Sigma B(x){:}\mathsf{Type}$ we obtain **G** proves $\Phi, x!A, z{:}B(x) \vdash_\Sigma z{:}B(x)$ by the $(MVar)$ rule. From these two assertions, by an application of $(!\Lambda L)$, we obtain **G** proves $\Phi\Psi, x!A, y{:}\Lambda x!A\,.B \vdash_\Sigma yx{:}B(x)$, where $\Phi\Psi$ is the κ-sensitive join of Φ and Ψ.

From **G** proves $\Phi\Psi, x!A, y{:}\Lambda x!A\,.B \vdash_\Sigma yx{:}B(x)$ and **G** proves $!\Delta \vdash_\Sigma N{:}A$, we obtain $\Phi\Psi, y{:}\Lambda x!A\,.B \vdash_\Sigma yN{:}B[N/x]$ by an application of the (Cut) rule. And from this last assertion and **G** proves $\Gamma \vdash_\Sigma M{:}\Lambda x!A\,.B$, we obtain $\Xi \vdash_\Sigma MN{:}B[N/x]$ by another application of the (Cut) rule, where Ξ is the κ-sensitive join of Γ and $\Phi\Psi$.

The above argument is summarized by the following proof figure:

$$
\dfrac{
\begin{array}{c}\text{IH on } \pi_0\\ \vdots\\ \Gamma \vdash_\Sigma M{:}\Lambda x!A\,.B\end{array}
\qquad
\dfrac{
\begin{array}{c}\text{IH on } \pi_1\\ \vdots\\ !\Delta \vdash_\Sigma N{:}A\end{array}
\qquad
\dfrac{
\dfrac{\begin{array}{c}\text{IH on } \pi_1\\ \vdots\\ \Psi \vdash_\Sigma A{:}\mathsf{Type}\end{array}}{\Psi, x!A \vdash_\Sigma x{:}A}\,(MVar!)
\qquad
\dfrac{\begin{array}{c}\text{IH on } \pi_0\\ \vdots\\ \Phi, x!A \vdash_\Sigma B(x){:}\mathsf{Type}\end{array}}{\Phi, x!A, z{:}B(x) \vdash_\Sigma z{:}B(x)}\,(MVar)\;(\dagger)
}{\Phi\Psi, x!A, y{:}\Lambda x!A\,.B \vdash_\Sigma yx{:}B(x)}\,(:\Pi L)\;(\dagger\dagger)
}{\Phi\Psi, y{:}\Lambda x!A\,.B \vdash_\Sigma yN{:}B[N/x]}\,(Cut)\;(\dagger\dagger\dagger)
}{\Xi \vdash_\Sigma MN{:}B[N/x]}\,(Cut)
$$

where $(\dagger)$, $(\dagger\dagger)$ and $(\dagger\dagger\dagger)$ are the side-conditions mentioned in the argument. □

5.4 Cut-elimination

In this section, we prove the cut-elimination theorem for the $\lambda\Lambda$-calculus. There are two ways to go about proving such a theorem: in the style of Gentzen, where it is shown that **G** proves a sequent if and only if **G**$\backslash$**cut** (*i.e.*, **G** without the cut rule) proves it too; and in a style analogous to Prawitz's, where it is shown that β-normal forms are provable in **N** if and only if they are provable in **G**$\backslash$**cut** too [Gen34, Pra65]. Following Pym [Pym95], we shall undertake the proof in the second style.

The result obtains only for β-normal forms. This corresponds to the fact that for systems without cut, we do not get completeness for proofs; rather for every proof in such system which is in normal form we get a proof in the system without cut. Pym notes that the relationship between the sequent calculus version of the λΠ-calculus and the natural deduction version of the λΠ-calculus is directly analogous with that between Gentzen's LJ and NJ. (One is reminded of Girard's "moral": normal=cut-free [GLT89].)

We prove that the system **G\cut** is sound and complete with respect to **N**. The cut-elimination theorem follows as a corollary of the completeness part.

Theorem 71 (Soundness of G\cut) *If* **G\cut** *proves* $\Gamma \vdash_\Sigma U{:}V$, *then* **N** *proves* $\Gamma \vdash_\Sigma U{:}V$.

Proof This is an immediate consequence of the soundness of **G** with respect to **N**. □

The proof of completeness of **G\cut** with respect to **N** depends crucially on a simple technical device, that of replacing the $(M\Lambda\mathcal{E})$ and $(M\Lambda!\mathcal{E})$ rules in the system **N** with the following slightly weaker versions:

$$\frac{\Gamma \vdash_\Sigma M{:}\Lambda x{:}A\,.B \quad \Delta \vdash_\Sigma N{:}A \quad \Psi \vdash_\Sigma B[N/x]{:}\mathsf{Type}}{\Xi \vdash_\Sigma MN{:}B[N/x]}\,(M\Lambda\mathcal{E})^\heartsuit$$

$$\frac{\Gamma \vdash_\Sigma M{:}\Lambda x!A\,.B \quad !\Delta \vdash_\Sigma N{:}A \quad \Psi \vdash_\Sigma B[N/x]{:}\mathsf{Type}}{\Xi \vdash_\Sigma MN{:}B[N/x]}\,(M\Lambda!\mathcal{E})^\heartsuit$$

where $\Psi \subseteq \Xi$ and Ξ is the κ-sensitive join of Γ and $(!)\Delta$. The rules are weaker because of the additional premiss. However, it is relatively straightforward to prove that the system $\mathbf{N}^\heartsuit$ which is obtained by replacing the $(M\Lambda\mathcal{E})$, $(M\Lambda!\mathcal{E})$ rules of system **N** by the above rules, is equivalent to **N**. The proof idea, in fact, is already to be found in the meta-theory of **N**, specifically Lemma 16 (Subderivation II), where we prove that if **N** proves $\Gamma \vdash_\Sigma M{:}A$, then **N** proves $\Gamma' \vdash_\Sigma A{:}K$, for some $\Gamma' \subset \Gamma$.

Lemma 72 (Equivalence of N and $\mathbf{N}^\heartsuit$) $\mathbf{N}^\heartsuit$ *proves* $\Gamma \vdash_\Sigma U{:}V$ *if and only if* **N** *proves* $\Gamma \vdash_\Sigma U{:}V$.

Proof If $\mathbf{N}^\heartsuit$ proves $\Gamma \vdash_\Sigma U{:}V$, then **N** proves $\Gamma \vdash_\Sigma U{:}V$ is immediate.

For the converse, we proceed by induction on the structure of the proofs in **N**. We only need to consider the $(M\Lambda\mathcal{E})$ and $(M\Lambda!\mathcal{E})$ rules. So suppose the last rule applied is $(M\Lambda\mathcal{E})$

$$\frac{\Gamma \vdash_\Sigma M{:}\Lambda x{:}A\,.B \quad \Delta \vdash_\Sigma N{:}A}{\Xi \vdash_\Sigma MN{:}B[N/x]}$$

By induction we have that $\mathbf{N}^{\heartsuit}$ proves $\Gamma \vdash_{\Sigma} M{:}\Lambda x{:}A\,.B$, and by induction on the structure of the proof of that assertion, we have that $\mathbf{N}^{\heartsuit}$proves $\Gamma' \vdash_{\Sigma} \Lambda x{:}A\,.B{:}\mathrm{Type}$, for some $\Gamma' \subseteq \Gamma$. By inversion, we obtain $\Gamma', x{:}A \vdash_{\Sigma} B{:}\mathrm{Type}$. By induction again, we have that $\mathbf{N}^{\heartsuit}$ proves $\Delta \vdash_{\Sigma} N{:}A$. We can assume that the cut rule is admissible in $\mathbf{N}^{\heartsuit}$; the proof of this is essentially the same as that for $\mathbf{N}$. Then, from these last two assertions we obtain, by the cut rule, $\mathbf{N}^{\heartsuit}$ proves $\Psi \vdash_{\Sigma} B[N/x]{:}\mathrm{Type}$, where Ψ is the κ-sensitive join of Γ' and Δ. It is easy to see that $\Psi \subset \Xi$. Finally, we obtain $\Xi \vdash_{\Sigma} MN{:}B[N/x]$ by the $(M\Lambda\mathcal{E})^{\heartsuit}$ rule.

The proof for the $(M\Lambda!\mathcal{E})$ rule is similar and we omit the details. $\square$

The proof of completeness is facilitated by the following lemma.

Lemma 73 (Application in β-normal form) *If M is a β-nf and if the last rule applied in the system* $\mathbf{N}$ *of the proof of $\Gamma \vdash_{\Sigma} M{:}A$ is either $(M\Lambda\mathcal{E})$, $(M\Lambda!\mathcal{E})$ or their non-dependent versions, then M is of the form $@N_1 \ldots N_m$ for some $@ \in dom(\Sigma) \cup dom(\Gamma)$ of appropriate type.*

Proof By induction on the structure of the proof of $\Gamma \vdash_{\Sigma} M{:}A$. We do the $(M\Lambda\mathcal{E})$ case.

Suppose $\Gamma \vdash_{\Sigma} MN{:}B[N/x]$ because $\Phi \vdash_{\Sigma} M{:}\Lambda x{:}A\,.B$ and $\Psi \vdash_{\Sigma} N{:}A$, with the usual application side-conditions. If MN is in β-nf, then M cannot be of the form $\lambda x{:}A\,.P$. Therefore, M is of the form $@N_1 \ldots N_m$ for some appropriate type. $\square$

In the sequel, we will take @ to denote either a constant or a variable.

Before we present the proof of completeness, we explain more informally how a proof in the system $\mathbf{N}$ with $(M\Lambda\mathcal{E})^{\heartsuit}$ (or $(M\Lambda!\mathcal{E})^{\heartsuit}$) as its last step is transformed into a proof in the system $\mathbf{G}$. *i.e.*, how we end up with a proof in $\mathbf{G}$ in which the final judgement is $\Xi \vdash_{\Sigma} MN{:}B[N/x]$. By induction hypothesis on the extra premiss in the $(M\Lambda\mathcal{E})^{\heartsuit}$ rule we have that $\mathbf{G}$ proves $\Psi \vdash_{\Sigma} B[N/x]{:}\mathrm{Type}$. So we can declare a variable of type $B[N/x]$. We then apply a series of left rules, with, in order, $\Delta \vdash_{\Sigma} N{:}A$, $\Gamma_m \vdash_{\Sigma} N_m{:}A_m[N_1/x_1, \ldots, N_{m-1}/x_{m-1}]$, $\ldots$, $\Gamma_1 \vdash_{\Sigma} N_1{:}A_1$ as the minor premisses. The applications of the left rules have the effect, firstly, of building up a declaration of @ in the antecedent and, secondly, of building up a term of type $B[N/x]$ in the succedent. Specifically, the i^{th} application of the left rule builds a variable y_i of type $\Lambda x_{i+1} \in A_{i+1}[N_1/x_1, \ldots, N_i/x_i] \ldots . (\Lambda x{:}A^*\,.B^*)[N_1/x_1, \ldots, N_m/x_m]$ in the antecedent and a term of form $y_i N_i N_{i-1} \ldots N$ in the succedent. The sharing-sensitive joining of the contexts eventually leaves Ξ as the final context.

The above description is the basic, underlying method in Pym [Pym95]. We say underlying because there the above-described prescription is preceded by a construction of the context $\vdash_\Sigma \Gamma, y_1{:}V_1, \ldots, y_{m-1}{:}V_{m-1}$, where each type V_i is the type of y_i as described above. This is necessary because of the form of the left rule adopted there, in which a contraction on the principal formula takes place. Here, we have to be more careful regarding context construction due to the action of context splitting.

Theorem 74 (Completeness of G\cut) *If* **N** *proves* $\Gamma \vdash_\Sigma U{:}V$, *then* **G\cut** *proves* $\Gamma \vdash_\Sigma U{:}V$.

Proof By induction on the structure of the proof of the hypothesis. The main idea is to exploit the structure of the β-nf U and it's type, V. The only differences between the systems **N** and **G\cut** are that the rules $(M\Lambda\mathcal{E})$, $(M\Lambda!\mathcal{E})$ and their non-dependent versions of **N** are replaced by the corresponding left rules. The majority of cases are trivial inductive arguments, which we omit, and some more difficult ones, which we include.

$(M\Lambda\mathcal{E})$ Suppose **N** proves $\Xi \vdash_\Sigma MN{:}B[N/x]$ because **N** proves $\Gamma \vdash_\Sigma M{:}\Lambda x{:}A\,.B$ (call this sub-proof π_0) and **N** proves $\Delta \vdash_\Sigma N{:}A$ (call this sub-proof π_1), with the usual application side-conditions. By hypothesis, MN is in β-nf, so that, by Lemma 73, M must be of the form $@N_1 \ldots N_m$, where $@$ is in $\Sigma \cup \Gamma$.

The type of $@$ is given by $\Lambda x_1 \in B_1 \ldots . \Lambda x_n \in B_n . \Lambda x{:}A^* . B^*$, where abstractions might be non-dependent ones (Pym actually has a notation for expressing both dependent and non-dependent function spaces, but we think it is easier for the reader to keep this fact at the back of his mind) and $\Lambda x{:}A^* . B^*[N/x] =_\beta \Lambda x{:}A\,.B$.

From this part of the proof, we let σ denote the substitution $[[N_1/x_1], \ldots [N_m/x_m]]$, and let σ_i denote the first i substitutions of σ. That is, $\sigma_i = [N_1/x_1, \ldots N_i/x_i]$.

By induction on the structure of proof of π_0, we have that

$$\mathbf{N} \;\text{ proves } \Gamma_i \vdash_\Sigma N_i{:}A_i\sigma_{i-1}$$

by shorter proofs, where Γ_i is that part of Γ needed to prove $N_i{:}A_i\sigma_{i-1}$. Therefore, by induction hypothesis, we have that

$$\mathbf{G\backslash cut} \;\text{ proves } \Gamma_i \vdash_\Sigma N_i{:}A_i\sigma_{i-1}$$

and that each N_i is in β-nf.

By induction hypothesis, making use of the additional premiss in the $(M\Lambda\mathcal{E})^{\heartsuit}$ rule, we have that

$$\mathbf{G}\backslash\mathbf{cut} \text{ proves } \Psi \vdash_{\Sigma} B[N/x]{:}\mathsf{Type}$$

From this, we get that **G\cut** proves $\Psi, y{:}B[N/x] \vdash_{\Sigma} y{:}B[N/x]$.

By induction hypothesis a third time, on π_1 this time, we have that

$$\mathbf{G}\backslash\mathbf{cut} \text{ proves } \Delta \vdash_{\Sigma} N{:}A$$

From these last two judgements, and by an application of the $(:\Lambda L)$ rule, we obtain **G\cut** proves $\Psi_1, y_1{:}\Lambda x{:}A.B \vdash_{\Sigma} y_1 N{:}B[N/x]$, where Ψ_1 is the κ-sensitive join of Ψ and Δ.

By the induction hypothesis, we have that **G\cut** proves $\Gamma_m \vdash_{\Sigma} N_m{:}A_m\sigma_{m-1}$. Then, by one of the left rules (the one we employ depends on the structure of the type of @), we obtain

$$\mathbf{G}\backslash\mathbf{cut} \text{ proves } \Psi_m, y_{m-1} \in \Lambda x_m \in A_m^* \sigma_{m-1}.\Lambda x{:}A^*.B^*\sigma \vdash_{\Sigma} y_{m-1} N_m N{:}B[N/x]$$

We proceed similarly for another $(m-2)$ (appropriate) left rule applications, until we obtain

$$\mathbf{G}\backslash\mathbf{cut} \text{ proves } \Psi_2, y_1 \in \Lambda x_2 \in A_2^* \sigma_1 \ldots .\Lambda x{:}A^*.B^*\sigma \vdash_{\Sigma} y_1 N_2 \ldots N_m N{:}B[N/x]$$

We then take one more such step and apply the appropriate left rule to **G\cut** proves $\Gamma_1 \vdash_{\Sigma} N_1{:}A_1$, which we have obtained by an induction hypothesis, to obtain

$$\mathbf{G}\backslash\mathbf{cut} \text{ proves } \Psi_1, @ \in \Lambda x_1 \in A_1^* \ldots .\Lambda x{:}A^*.B^*\sigma \vdash_{\Sigma} @N_1 \ldots N_m N{:}B[N/x]$$

It might now be necessary to derelict @ if @ is in Σ. We can then see that the context is Ξ, and thus have obtained the required judgement.

There is a non-dependent version of this case, whose proof is done similarly. We omit the details.

$(M\Lambda!\mathcal{E})$ This case is done similarly to the $(M\Lambda\mathcal{E})$ one above. And like the $(M\Lambda\mathcal{E})$ case, there is a non-dependent version of $(M\Lambda!\mathcal{E})$, whose details we omit. □

Cut-elimination follows.

Corollary 75 (Cut-elimination) *Suppose that* **G** *proves* $\Gamma \vdash_\Sigma M{:}A$. *If* M *is in* β*-nf, then* **G\cut** *proves* $\Gamma \vdash_\Sigma M{:}A$.

Proof By Theorem 69, if **G** proves $\Gamma \vdash_\Sigma M{:}A$ then $\mathbf{N}^\heartsuit$ proves $\Gamma \vdash_\Sigma M{:}A$. By Lemma 72 and Theorem 74, if M is β-nf then **G\cut** proves $\Gamma \vdash_\Sigma M{:}A$. □

5.5 Summary

The earlier presentation of the $\lambda\Lambda$-calculus, in Chapter 2, is in linearized natural deduction form. In this chapter, we presented a Gentzenization of the original type theory, replacing the elimination rules by left rules. We have proved soundness and completeness results, and a cut-elimination theorem for the resulting sequent calculus system. The categorical semantics of the Gentzenized system remains a subject of further investigation.

This chapter begins the investigation of a notion of computation in the type theory. This is one of proof-search, an application of cut-elimination. We would then like to relate proof-search in the $\lambda\Lambda$-calculus, the type-theoretic meta-language of the logical framework RLF, to proof-search in those object-logics which can be uniformly encoded in RLF.

Chapter 6

Conclusions

In this thesis, we have presented a study of a logical framework, RLF, for defining natural deduction presentations of linear and other relevant logics. RLF consists in a language together, in a manner similar to that of LF, with a representation mechanism. The language of RLF is the $\lambda\Lambda$-calculus; the representation mechanism is judgements-as-types, developed for relevant logics.

The $\lambda\Lambda$-calculus is a first-order dependent type theory with two kinds of dependent function spaces, a linear one and an intuitionistic one. A large portion of this thesis was devoted to studying the semantics of the type theory. We gave a natural deduction presentation of the $\lambda\Lambda$-calculus, establishing the required proof-theoretic meta-theory. We also gave a sequent calculus presentation of the type theory and showed cut-elimination for it. The internal logic of the type theory is a structural fragment of **BI**, the logic of bunched implications. Inspired by the resource semantics of **BI**, we studied the categorical model theory of the type theory. We presented an interesting class of Kripke resource models by a construction on $\mathbf{Set}^{C^{op}}$, where C is a small monoidal category.

There are several avenues of work that we might wish to explore. We outline these next.

The $\lambda\Lambda$-calculus and the RLF logical framework

The type theory that we have studied has an interesting genesis. Though it arose from an analysis, primarily, of intuitionistic linear logic, it is structurally much closer to **BI**. One of the departure points of the $\lambda\Lambda$-calculus from ILL is linear dependency. The approach presented here, using a certain notion of context joining and by restricting Π elimination, is one way that linear dependency can be set up. It is not clear whether some other, more sophisticated context combination, taking account of well-typedness criteria, might work too.

We have also noted the concept of a "bunches cube". Within the cube, there are notions of predicate and polymorphic dependency. But there is also a lattice of cubes, each with its particular structural properties. There are interesting representational issues here, and the field can be seen as a programme in van Benthem's notion of logical pluralism, investigating connections between meta-properties of proof calculi and structural properties of systems of deduction [vB93].

This thesis began by stating that logical frameworks provide a foundation for the development of a generic proof assistant, and then went on to develop a logical framework for linear and other relevant logics. One practical piece of further work is the development of logical frameworks. Such a project would involve issues of representation, man-machine interface, and the technology of proof search. One particularly interesting issue here is the relationship concerning a cognitive gesture, an object logic proof step, and a meta-logic proof step. The implementation of RLF should be contrasted to LEGO, which is really an implementation of the language of LF and does not take the issues of representation seriously [LP92]. It should also be contrasted to Jape, which has a highly developed user interface but a very weak logical and representation foundation [BS97]. Similar comments apply to other type-theoretic proof assistants (Coq [BBC$^+$97], ALF [AGNvS96]) and syntactic calculators (Burstall's ProveEasy [Bur98]). The development of RLF explicates the view of the logical framework as an architecture, in which encoding an object logic is similar to compiling a language and a uniform representation theorem amounts to providing a proof of correct implementation.

The propositions-as-types correspondence

In Chapter 3, our presentation of **BI** was mainly for the purposes of establishing the Curry-Howard-de Bruijn correspondence. We forced ourselves to work with a fragment of **BI**, a fragment which is very close to being Linear Logic, in fact. What precisely is the internal logic of the $\lambda\Lambda$-calculus — what is its relationship with regard to Linear Logic and to full **BI** — is left to further work.

Quite apart from this, **BI** has some quite interesting computational interpretations, including a connection with Reynold's Syntactic Control of Interference and Idealized Algol [Rey78, Rey81]. We should like to investigate **BI** as a programming language.

Kripke resource semantics

In Chapter 4, we gave a class of models based on presheaves. It would be interesting to investigate whether there are other concrete models too. In their theses, Ambler studied Q-sets (sets over quantales) and Streicher studied ω-sets/realizability and PER structures [Amb92, Str88]. It is an interesting question to ask what account of resources these structures might give (and, in fact, this question might help in clarifying our notion of resource).

The account that the semantics gives of the syntax is a fairly static one. It would be interesting to investigate the extent to which structures can explain the dynamic nature of proof-search. For instance, the realizers might contain information on how they were constructed. (One is reminded of the "proofs as processes" slogan.) We can start by giving a semantic account of the Gentzenized system.

This leads us to consider the semantics of logical frameworks. This thesis begins this by studying the semantics of the language of the RLF logical framework. The next step is to give models of judged object-logics. Syntactically, the latter can be presented via Martin-Löf's arities [NPS90] or Aczel's Frege structures [Acz80], developed for relevant logics. We will then be in a position to study the relation between models of RLF-encoded judged object-logics and the Kripke resource models of the $\lambda\Lambda$-calculus, and to give a semantic account of the judgement-as-types method of encoding.

Finally, there are several aspects of the Kripke resource models which lead to interesting programming concepts. For instance, we can ask about the Kliesli category that arises from considering $(C, \otimes, I)$ and $(C, \times, 1)$ as monads. If we can interpret monads as continuations, then what, computationally, are these two kinds of continuations?

The Gentzenized $\lambda\Lambda$-calculus

In Chapter 5, we started to investigate the notion of proof-search, the basis of logic programming, in the $\lambda\Lambda$-calculus. The further refinement of the Gentzenized $\lambda\Lambda$-calculus can proceed along the lines of Pym and Wallen [PW91]. This involves the development of resolution rules, uniform proofs, permutation theorems, unification algorithms, and effective search strategies.

Pragmatically, the role of the quantifiers seems quite interesting; if understood as module interfaces, then we get two kinds of `consult`. We might ask what are the similarities between the two kinds of `consult` and the two kinds of storage allocation suggested in O'Hearn's regions analysis of **BI** [O'H99a].

Towards a theory of meaning

It is important to consider the philosophical ideas underlying the more technical presentation of Kripke resource semantics. Let us start by considering Kripke's theory of meaning for intuitionistic logic, and then go on to sketch an idea for a resource-conscious creative subject.

Kripke's semantics is based on an ideal mathematician, or a creative subject, constructing knowledge through time. At an initial time W_0, the subject starts off with a certain amount of knowledge (which may be nil). He extends his state of knowledge and constructs new objects over time. That is, the subject engages in "positivistic research". The states of knowledge form themselves into a partial order. Moreover, as the subject has certain choices for his future activities, the order is a branching one.

How does the creative subject interpret the logical connectives? Conjunction, disjunction and existential quantification are straight-forward: $A \wedge B$ is known whenever A and B are known; $A \vee B$ is known whenever A or B are known; $\exists x.A$ is known whenever $A[a/x]$ is known. Implication presents the problem that $A \supset B$ might be known without A and/or B being known. The solution is to interpret implication as follows: $A \supset B$ is known at stage W_i if, at some future stage W_{i+j}, if A is known, then B is known too. Universal quantification is dealt with similarly. That is, $\forall x.A$ is known at state W_i if, at some future stage W_{i+j}, for all objects a that exist, $A[a/x]$ is known.

Good accounts of the creative subject are given by Dummett [Dum77] and van Dalen [vD94].

One criticism of this semantics is that it captures only a certain aspect of the subjects behaviour, the temporal one. This is perfectly adequate for much of traditional mathematical knowledge, where lemmata once proven remain so and can be used or not as the subject feels able. But it doesn't account for other concepts, such as state, which mutate or evolve over time, and not necessarily in a monotonic fashion.

The idea is to refine Kripke semantics by providing another dimension, a resource one, which further indexes the creative subject. One example of a mechanical resource is storage space or memory, which holds the current state of a computer. X amount of storage space only keeps X

amount of state. If the creative subject is to progress, then he has to adjoin, or add, further chunks of storage space.

There is a sense in which knowledge is a resource. But, in general, these seem to be distinct categories. Knowledge increases monotonically over time; the creative subject can synthesize more knowledge from the knowledge he currently has. This does not seem to be the case for resources. The creative subject cannot create more resources, such as space, from what he has already. (He can re-use them afterwards — and this is an interesting link to n-variable logics and similar areas in the logical complexity of space.) Resources put a constraint on the extent of what the creative subject can construct. We can look back at Urquhart's semantics for relevant logic in a similar explanatory light [Urq72]. His semantics is based on *pieces of information*; these are the resources which index computation. The *pieces of information* are distinguished from *facts*.

The canonical example of a resource-conscious creative subject is that of the bottom-up execution of a linear logic program. As the program executes through time, it adds to its knowledge (of answer substitutions). The linear propositions are consumed during the execution. The cost of the execution is given by the memory freed up by these propositions till the current point in time.

Bibliography

[AB75] AR Anderson and JD Belnap. *Entailment: The Logic of Relevance and Necessity*. Princeton University Press, 1975.

[Abr93] S Abramsky. Computational interpretations of linear logic. *Theoretical Computer Science*, 111:3–57, 1993.

[Acz80] P Aczel. Frege Structures and the Notions of Proposition, Truth and Set. In J Barwise, HJ Keisler, and K Kunen, editors, *The Kleene Symposium*. North-Holland, 1980.

[AGM92] S Abramsky, D Gabbay, and TSE Maibaum, editors. Oxford Science Publications, 1992.

[AGNvS96] T Altenki, V Gaspes, B Nordstrm, and B von Sydow. A user's guide to ALF, 1996. Available from Department of Computing Science, University of Göteborg/Chalmers.

[AHMP92] A Avron, F Honsell, IA Mason, and R Pollack. Using typed lambda calculus to implement formal systems on a machine. *Journal of Automated Reasoning*, 9:309–354, 1992.

[AHMP98] A Avron, F Honsell, M Miculan, and C Paravano. Encoding modal logics in logical frameworks. *Studia Logica*, 60(1), 1998.

[Amb92] S Ambler. *First Order Linear Logic in Symmetric Monoidal Closed Categories*. PhD thesis, Edinburgh University, 1992. Available as Edinburgh University Computer Science Department Technical Report ECS-LFCS-92-194.

[Avr88] A Avron. The semantics and proof theory of linear logic. *Theoretical Computer Science*, 57:161–184, 1988.

[Bar84] HP Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[Bar92] HP Barendregt. *Lambda Calculi with types*, pages 118–310. Volume 2 of Abramsky et al. [AGM92], 1992.

[Bar97] A Barber. *Linear Type Theories, Semantics and Action Calculi*. PhD thesis, University of Edinburgh, 1997.

[BBC$^+$97] B Barras, S Boutin, C Cornes, J Courant, J Filliatre, E Giménez, H Herbelin, G Huet, CM Noz, C Murthy, C Parent, C Paulin, A Saïbi, and B Werner. The Coq Proof Assistant Reference Manual – Version v6.1. Technical Report 0203, INRIA, 1997.

[BD97] J Baez and J Dolan. Higher-dimensional algebra III: n-categories and the algebra of opetopes, 1997. To appear in *Advances in Mathematics*.

[Ben94] PN Benton. A mixed linear and non-linear logic: Proofs, terms and models (preliminary report). Technical Report 352, Computer Laboratory, University of Cambridge, 1994.

[Bie94] GM Bierman. On intuitionistic linear logic. Technical Report 346, University of Cambridge Computer Laboratory, August 1994.

[BS97] R Bornat and B Sufrin. Roll your own Jape logic, 1997. Available from Department of Computer Science, Queen Mary & Westfield College.

[Bur98] R Burstall. Teaching people to write proofs: a tool. In *CafeOBJ Symposium, Numazu, Japan*, 1998.

[Car86] J Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, 32:209–243, 1986.

[Cer96] I Cervesato. *A Linear Logical Framework*. PhD thesis, Università di Torino, 1996. (In Italian.).

[Coq91] T Coquand. An algorithm for testing conversion in type theory. In Huet and Plotkin [HP91], pages 255–279.

[CP96] I Cervesato and F Pfenning. A linear logical framework. In E Clarke, editor, *11th LICS, New Brunswick, NJ*, pages 264–275. IEEE Computer Society Press, 1996.

[Day70] BJ Day. On closed categories of functors. In S Mac Lane, editor, *Reports of the Midwest Category Seminar*, volume 137 of *Lecture Notes in Mathematics*, pages 1–38. Springer-Verlag, 1970.

[dB91] NG de Bruijn. A plea for weaker frameworks. In Huet and Plotkin [HP91], pages 40–68.

[Dum77] M Dummett. *Elements of Intuitionism.* Clarendon Press, Oxford, 1977.

[Dun86] JM Dunn. Relevance logic and entailment. In Gabbay and F Guenthner, editors, *Handbook of Philosophical Logic*, volume III, chapter 3, pages 117–224. D Reidel, 1986.

[Gan80] RO Gandy. Proofs of strong normalization. In Seldin and Hindley [SH80], pages 457–478.

[GdQ92] DM Gabbay and RJB de Queiroz. Extending the Curry-Howard interpretation to linear, relevant and other resource logics. *Journal of Symbolic Logic*, 57(4), 1992.

[Gen34] G Gentzen. Untersuchungen über das logishche Schliessen. *Mathematische Zeitschrift*, 39:176–210, 405–431, 1934. Translation in [Gen69].

[Gen69] G Gentzen. *The Collected Papers of Gerhard Gentzen.* North-Holland, 1969. English translation, edited and introduced by ME Szabo.

[Geu93] JH Geuvers. *Logics and Type Systems.* PhD thesis, Katholieke Universiteit Nijmegen, 1993.

[Gir87] J-Y Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[GLT89] J-Y Girard, Y Lafont, and P Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, 1989.

[HHP93] R Harper, F Honsell, and G Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

[HM94] JS Hodas and D Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, 110(2):327–365, 1994.

[How80] WA Howard. The formula-as-types notion of construction. In Seldin and Hindley [SH80], chapter 9, pages 479–490.

[HP91] G Huet and G Plotkin, editors. Cambridge University Press, 1991.

[HPW96] JA Harland, DJ Pym, and J Winikoff. Programming in Lygon: An overview. In M Wirsing and M Nivat, editors, *Algebraic Methodology and Software Technology*, volume 1101 of *LNCS*, pages 391–405. Springer-Verlag, 1996.

[HST94] R Harper, D Sannella, and A Tarlecki. Structured theory representations and logic representations. *Annals of Pure and Applied Logic*, 67:113–160, 1994.

[IP98] SS Ishtiaq and DJ Pym. A Relevant Analysis of Natural Deduction. *Journal of Logic and Computation*, 8(6):809–838, 1998.

[Kan00] I Kant. *Immanuel Kants Logik (Edited by GB Jäsche).* Friedrich Nicolovius, Königsberg, 1800. In translation: RS Hartman and W Schwarz, Dover Publications, Inc., 1988.

[Kri65] S Kripke. Semantic analysis of intuitionistic logic I. In JN Crossley and MAE Dummett, editors, *Formal Systems and Recursive Functions*, pages 92–130. North-Holland, 1965.

[LP92] Z Luo and R Pollack. Lego proof development system: User's manual. Technical Report ECS-LFCS-92-211, Department of Computer Science, University of Edinburgh, 1992.

[Mey76] RK Meyer. Relevant arithmetic. *Polish Academy of Sciences, Institute of Philosophy and Bulletin of the Section of logic*, 5:133–137, 1976.

[ML75] P Martin-Löf. An intuitionistic theory of types: Predicate part. In HE Rose and JC Shepherdson, editors, *Logic Colloquium '73*, volume 80 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1975.

[ML96] P Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, 1996. (Also: Technical Report 2, Scuola di Specializiazzione in Logica Matematica, Dipartimento di Matematica, Università di Siena, 1982.).

[MM91] JC Mitchell and E Moggi. Kripke-style models for typed lambda calculus. *Annals of Pure and Applied Logic*, 51:99–124, 1991.

[MTV93] M Masseron, C Tollu, and J Vauzeilles. Generating plans in linear logic. *Theoretical Computer Science*, 113:349–370 and 371–375, 1993.

[NPS90] B Nordström, K Petersson, and JM Smith. *Programming in Martin-Löf type theory: an introduction*. Oxford, 1990.

[O'H99a] PW O'Hearn. A regions interpretation of BI. Manuscript, 1999.

[O'H99b] PW O'Hearn. Resource Interpretations, Bunched Implications and the $\alpha\lambda$-calculus. In J-Y Girard, editor, *Proceedings of Typed Lambda Calculus and Applications, L'Aquila, Italy*, 1999.

[OP99] PW O'Hearn and DJ Pym. The logic of bunched implications. To appear in *Bulletin of Symbolic Logic*, 1999.

[PH94] DJ Pym and JA Harland. A uniform proof-theoretic investigation of linear logic programming. *Journal of Logic and Computation*, 4(2):175–207, 1994.

[Pit92] A Pitts. Categorical logic. In Abramsky et al. [AGM92].

[Pra65] D Prawitz. *Natural Deduction: A Proof-Theoretic Study*. Almqvist & Wiksell, 1965.

[Pra78] D Prawitz. Proofs and the meaning and completeness of the logical constants. In J Hintikka, J Nuniluoto, and E Saarinen, editors, *Essays on Mathematical and Philosophical Logic*, pages 25–40. D Reidel, 1978.

[PS78] R Paré and D Schumacher. Abstract Families and the Adjoint Functor Theorems. In PT Johnstone *et al.*, editor, *Indexed Categories and Their Applications*, volume 661 of *Lecture Notes in Mathematics*. Springer-Verlag, 1978.

[PW91] DJ Pym and L Wallen. Proof-search in the $\lambda\Pi$-calculus. In Huet and Plotkin [HP91], pages 309–340.

[Pym90] DJ Pym. *Proofs, Search and Computation in General Logic*. PhD thesis, University of Edinburgh, 1990. Available as Edinburgh University Computer Science Department Technical Report ECS-LFCS-90-125.

[Pym92] DJ Pym. A relevant analysis of natural deduction. Lecture at Workshop, EU Espirit Basic Research Action 3245, Logical Frameworks: Design, Implementation and Experiment, Båstad, Sweden, May 1992. (Joint work with D Miller and G Plotkin.).

[Pym95] DJ Pym. A note on the proof theory [of] the λΠ-calculus. *Studia Logica*, 54:199–230, 1995.

[Pym96] DJ Pym. A note on representation and semantics in logical frameworks. In D Galmiche, editor, *Proceedings of CADE-13 Workshop on Proof-search in Type-theoretic Languages, Rutgers University, New Brunswick, NJ*, 1996. (Also: Technical Report 725, Department of Computer Science, Queen Mary & Westfield College, University of London.).

[Pym97] DJ Pym. Functorial Kripke models of the λΠ-calculus, 1997. Lecture at Isaac Newton Institute for Mathematical Sciences, Semantics Programme, Workshop on Categories and Logic Programming, Cambridge, 1995. Paper(s) in preparation.

[Pym98] DJ Pym. Logic Programming with Bunched Implications (extended abstract). *Electronic Notes in Theoretical Computer Science*, 17, 1998.

[Rd97] E Ritter and V de Paiva. New models of intuitionistic linear logic. Manuscript, 1997.

[Rea88] S Read. *Relevant Logic: A Philosophical Examination of Inference*. Basil Blackwell, 1988.

[Rey78] JC Reynolds. Syntactic control of interference. In *Conference Record of the Fifth Annual ACM Symposiym on Principles of Programming Languages*, pages 39–46, 1978.

[Rey81] JC Reynolds. The essence of Algol. In JW de Bakker and JC van Vliet, editors, *Algorithmic Languages*. North-Holland, 1981.

[Sal90] A Salvesen. A proof of the Church-Rosser property for the Edinburgh LF with η-conversion. Lecture given at the First Workshop on Logical Frameworks, Sophia-Antipolis, France, May 1990.

[SH80] JP Seldin and JR Hindley, editors. *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism.* Academic Press, 1980.

[SH83] P Schroeder-Heister. Generalized rules for quantifiers and the completeness of the intuitionistic operators $\&, \vee, \supset, \curlywedge, \forall, \exists$. In MM Richter *et al.*, editor, *Computation and Proof Theory, Logic Colloquim Aachen*, volume 1104 of *Lecture Notes in Mathematics*, pages 399–426. Springer-Verlag, 1983.

[SH91] P Schroeder-Heister. Structural frameworks, substructural logics, and the role of elimination inferences. In Huet and Plotkin [HP91], pages 385–403.

[Ste72] S Stenlund. *Combinators, λ-terms and Proof Theory.* D Reidel, 1972.

[Str88] T Streicher. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions.* PhD thesis, Universität Passau, 1988.

[Ten92] N Tennant. *Autologic.* Edinburgh University Press, 1992.

[TF92] H Tonino and K Fujita. On the adequacy of representing higher order intuitionistic logic as a pure type system. *Annals of Pure and Applied Logic*, 57:251–276, 1992.

[Tro92] A Troelstra. *Lectures on Linear Logic.* CSLI, 1992.

[TS96] AS Troelstra and H Schwichtenberg. *Basic Proof Theory.* Number 43 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1996.

[Urq72] A Urquhart. Semantics for relevant logics. *Journal of Symbolic Logic*, 37(1):159–169, March 1972.

[vB93] J van Benthem. The landscape of deduction. In P Schroeder-Heister and K Došen, editors, *Sub-structural Logics.* Oxford Science Publications, 1993.

[vD80] D van Daalen. *The Language Theory of AUTOMATH.* PhD thesis, Technical University of Eindhoven, Eindhoven, Netherlands, 1980.

[vD94] D van Dalen. *Logic and Structure.* Springer-Verlag, 1994.

[Wal90] LA Wallen. *Automated Deduction in Non-Classical Logics.* MIT Press, 1990.

[Wan93] H Wansing. *The Logic of Information Structures.* Number 681 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1993.