Kripke Resource Models of a Dependently-typed, Bunched λ-calculus

SAMIN ISHTIAQ, *ARM Ltd., Cambridge, England, UK.*
E-mail: samin.ishtiaq@arm.com

DAVID J. PYM, *University of Bath, Claverton Down, Bath BA2 7AY, England, UK.*
E-mail: d.j.pym@bath.ac.uk

Abstract

The $\lambda\Lambda$-calculus is a dependent type theory with both linear and intuitionistic dependent function spaces. It can be seen to arise in two ways. Firstly, in logical frameworks, where it is the language of the RLF logical framework and can uniformly represent linear and other relevant logics. Secondly, it is a presentation of the proof-objects of a structural variation, with Dereliction, of a fragment of **BI**, the logic of bunched implications. As such, it is also closely related to linear logic. **BI** is a logic which directly combines linear and intuitionistic implication and, in its predicate version, has both linear and intuitionistic quantifiers. The $\lambda\Lambda$-calculus is the dependent type theory which generalizes both implications and quantifiers. In this paper, we study the categorical semantics of the $\lambda\Lambda$-calculus, gives a theory of 'Kripke resource models', i.e. monoid-indexed sets of functorial Kripke models, in which the monoid gives an account of resource consumption. A class of concrete, set-theoretic models is given by the category of families of sets parametrized over a small monoidal category, in which the intuitionistic dependent function space is described in the established way, but the linear dependent function space is described using Day's tensor product.

Keywords: Dependent type theory, categorical semantics, Kripke models, logical frameworks, sub-structural logics.

1 Introduction

A long-standing problem has been to combine type-dependency and linearity. In [17], the present authors introduced the $\lambda\Lambda$-calculus, a first-order dependent type theory with a full linear dependent function space, as well as the usual intuitionistic dependent function space. The $\lambda\Lambda$-calculus can be seen to arise in two ways: in logical frameworks and in linear and bunched logics.

Logical frameworks. Logical frameworks are formal meta-logics which, *inter alia*, provide languages for describing logics in a manner that is suitable for mechanical implementation. 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

$$Framework = Language + Representation.$$

We remark that these components are not entirely independent of each other.

One representation mechanism is that of judgements-as-types, which originates from Martin-Löf's [21] development of Kant's notion of judgement [18]. 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 rule and proofs are seen as proofs of hypothetico-general judgements. Representation theorems relate consequence in an object-logic $\vdash_L$ to consequence in an encoded logic $\vdash_{\Sigma_L}$

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

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.

The LF logical framework consists of the $\lambda\Pi$-calculus together with the judgements-as-types mechanism for representing logics [1, 10, 29]. One consequence of this method of encoding is that encoded systems inherit the structural properties of the meta-logic. Now, the structural strength of LF is determined by the structural strength of the $\lambda\Pi$-calculus which, as it stands in propositions-as-types correspondence with the $\{\rightarrow,\forall\}$-fragment of intuitionistic logic, admits the structural rules of weakening and contraction. Consequently, LF is able to *uniformly* represent, i.e. the encoding $\Downarrow$ is surjective on proof-objects, only logics which also admit these structurals [34, 11].

We illustrate the use of LF by giving a brief example of how the $\{\wedge,\supset\}$-fragment of Intuitionistic Logic (IL) is uniformly represented in LF via judgements-as-types. The natural deduction presentation of this logic, in which the sole judgement is concerned with the $proof$ of a proposition, is as follows:

$$\frac{\phi\ proof \quad \psi\ proof}{\phi\wedge\psi\ proof}\wedge I \qquad \frac{\phi_0\wedge\phi_1\ proof}{\phi_i\ proof}\ (i\in\{0,1\})\ \wedge E$$

$$\frac{\begin{array}{c}(\phi)\\ \vdots\\ \psi\ proof\end{array}}{\phi\supset\psi\ proof}\supset I \qquad \frac{\phi\supset\phi\ proof \quad \phi\ proof}{\psi\ proof}\supset E.$$

The signature Σ_{IL} begins by declaring the constant $o{:}\mathsf{Type}$ to represent the syntactic category of propositions. We declare the constant $proof{:}o\rightarrow\mathsf{Type}$ to represent the object-logic judgement. A proof of $\vdash_{IL}\phi\ proof$ is represented by a term of type $proof(\phi)$ in the meta-logic. The representation of the syntax is completed by declaring constants for each of the two formula constructors:

$$\wedge{:}o\rightarrow o\rightarrow o \qquad \supset{:}o\rightarrow o\rightarrow o.$$

The object-logic rules are then represented by the following declarations, where $i\in\{0,1\}$:

$$\begin{array}{rcl} \text{AND-I} & : & \mathbf{\Pi}\,\phi,\psi{:}o\,.\,proof(\phi)\rightarrow proof(\psi)\rightarrow proof(\wedge(\phi,\psi)) \\ \text{AND-E}_i & : & \mathbf{\Pi}\,\phi_0,\phi_1{:}o\,.\,proof(\wedge(\phi_0,\phi_1))\rightarrow proof(\phi_i) \\ \text{IMP-I} & : & \mathbf{\Pi}\,\phi,\psi{:}o\,.\,(proof(\phi)\rightarrow proof(\psi))\rightarrow proof(\supset(\phi,\psi)) \\ \text{IMP-E} & : & \mathbf{\Pi}\,\phi,\psi{:}o\,.\,proof(\supset(\phi,\psi))\rightarrow proof(\phi)\ proof(\psi). \end{array}$$

A strong representation theorem — a bijection, in fact — can be given for this encoding since the $\lambda\Pi$-calculus has the same structural strength as IL; both admit the structural rules of weakening and contraction. It is, in fact, for this reason that LF cannot uniformly encode linear and other relevant logics. 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. This structural strength excludes from LF's scope object logics involving the notions of intension and state.

In [17], the present authors present a language in which such weakening and contraction are not forced. The connectives of such a language are motivated by studying the natural deduction form of rules for relevant logics and the resulting language forms the basis of the RLF logical framework [17]. This is done quite generally, by considering Prawitz's general form of schematic introductions from a more relevant point of view. Consider a schematic $\#I$ rule as given by the figure 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 $H_{j,1},\ldots,H_{j,h_j}$ which 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.

The introduction schema is annotated as follows to indicate the 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 $\Pi F{:}o$. We also use $\Lambda F{!}o$ 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 $\multimap$.

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

The premisses $G_1,\ldots,Gp$ are combined either multiplicatively or additively, depending on whether their contexts are disjoint or not. These combinations are distinguished by the use of two conjunctions, the multiplicative $\otimes$ and the additive $\&$, and so the structural strength is forced. We have used $\Box$ as meta-syntax for both $\otimes$ and $\&$.

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

$$\Lambda\, F_g, G_j, H_{j,k} \in o \ldots \Box(\Box_{l \leq h_j}(H_{j,l}) \longrightarrow\!\!\circ\ G_j)\Box \ldots \longrightarrow\!\!\circ\ \#(F_1, \ldots, F_n),$$

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 connective $\otimes$ occurs only negatively, allowing us to curry it away.

We emphasize how the three logical constants have been used in the framework: the $\&$ is used to undertake additive conjunction; the Λ is used to quantify and (in its non-dependent form $-\!\circ$) to represent implication; and the Π is used to represent dereliction from relevant inference. It should be possible then to formulate a precise idea regarding the completeness of the set $\{\&, \to, -\!\circ, \Pi, \Lambda\}$ with respect to all sentential operators that have explicit schematic introduction rules [27, 40]. A similar analysis can be undertaken for the corresponding elimination rule.

Thus, by analysing the form of relevant natural deduction, the $\lambda\Lambda$-calculus can be seen to arise as the language of the RLF logical framework, which consists of the $\lambda\Lambda$-calculus together with the judgements-as-types mechanism for representing logics. Since the $\lambda\Lambda$-calculus admits weakening and contraction only in highly restricted circumstances, determined by the $!- \ -\!\circ\ -$ translation of intuitionistic logic into linear logic, the structural strength of RLF corresponds to that of intuitionistic linear logic and, so, is sufficiently weak to uniformly represent systems such as linear and other relevant logics and stateful type systems such as ML with references [17]. In fact, RLF is closely related, via a propositions-as-types correspondence [13, 16], to a structural variant (with Dereliction) of a fragment of the bunched logic, **BI**.

Linear and bunched logics. The second way in which the $\lambda\Lambda$-calculus arises is in the context linear logic [9] and of **BI**, the logic of bunched implications [24]. In **BI**, a multiplicative (or linear) and an additive (or intuitionistic) implication live side-by-side. The propositional version of **BI** arises from an analysis of the proof-theoretic relationship between conjunction and implication, and can be viewed as a merging of intuitionistic logic and multiplicative, intuitionistic linear logic. Such a system requires that antecedents of logical consequences be structured not as lists but rather as *bunches*, in which there are two kinds of combining operation, ';', which admits weakening and contraction and ',', which does not. Bunches, which originally arose in relevant logic [38], allow the formation of two kinds of function space, the intuitionistic one $\to$, corresponding to ';' and the linear one $-\!\circ$, corresponding to ',',[1] The introduction rules are given by

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A -\!\circ B} -\!\circ I \qquad \frac{\Gamma; A \vdash B}{\Gamma \vdash A \to B} \to I.$$

In eliminating the connectives, the premiss contexts must be combined with regard to the type (whether linear or intuitionistic) of the connective.

$$\frac{\Gamma \vdash A -\!\circ B \quad \Delta \vdash A}{\Gamma, \Delta \vdash B} -\!\circ E \qquad \frac{\Gamma \vdash A \to B \quad \Delta \vdash A}{\Gamma; \Delta \vdash B} \to E.$$

[1]The usual presentation of **BI** uses the symbol $-\!\!*$, or 'magic wand', to denote multiplicative implication. We use $-\!\circ$ here simply for uniformity within this presentation.

A similar distinction obtains at the level of predicates and quantifiers. To see this, consider the following form of first-order sequent:

$$(X)\,\Gamma \;\vdash\; \phi\,,$$

where X denotes the collection of first-order variables occurring in Γ and ϕ. If we allow X to be structured as a bunch, then we can identify two forms of universal quantifier, with the following introduction rules:

$$\frac{(X;!x)\Gamma \vdash \phi}{(X)\Gamma \vdash \forall X.\phi}\quad \forall I \qquad \frac{(X,x)\Gamma \vdash \forall_{\mathrm{new}} x.\phi}{(X)\Gamma \vdash \phi}\quad \forall_{\mathrm{new}} I,$$

where $x \notin \mathrm{FV}(\Gamma)$.[2]

The rule of Dereliction in **BI**, in a simplified and propositional form,[3]

$$\frac{\Gamma, \phi \vdash \psi}{\Gamma; \phi \vdash \psi},$$

reveals the relationship between the fragment of **BI** with which we shall be concerned and linear logic. We return to this point in the sequel.

BI possesses two very natural semantics. The first of these, a categorical semantics of proofs, is given, at the propositional level, by doubly closed categories (DCCs). A DCC is a category equipped with two monoidal closed structures; it is called cartesian if one of the closed structures is cartesian and the other symmetric monoidal. So, categorically, the ';' is modelled by a cartesian product and the ',' is modelled by a symmetrical monoidal product in the same category.

A rich class of models of **BI** can be obtained by using a construction due to Day [8], who shows that any monoidal category $\mathcal{C}$ induces a monoidal closed structure on $\mathbf{Set}^{\mathcal{C}^{op}}$. This, combined with the cartesian structure on $\mathbf{Set}^{\mathcal{C}^{op}}$, yields a host of concrete models for **BI**. The construction follows. The unit I of the monoidal structure is $\mathcal{C}[-, I]$. The tensor product is written using co-ends,

$$(A \otimes B)Z = \int^{X,Y} AX \times BY \times \mathcal{C}[Z, X \otimes Y],$$

and the linear implication is written using an end,

$$(A \multimap B)X = \int_{Y} \mathbf{Set}[AY, B(X \otimes Y)] \cong \mathbf{Set}^{\mathcal{C}^{op}}[A(-), B(X \otimes -)].$$

(An end and its dual can be seen, roughly speaking, as quantifiers. The categorical definition of an end is as a limit obtained from certain bifunctors [20].)

The second semantics of **BI** is a Kripke-style semantics of formulae, which combines Kripke's semantics of intuitionistic logic [19] and Urquhart's semantics of relevant logic [42]. These can be understood to be given, respectively, in $\mathbf{Set}^{\mathcal{P}}$, where $\mathcal{P}$ is a poset, and in $\mathbf{Set}^{\mathcal{M}}$, where $\mathcal{M}$ is a commutative monoid. A semantics of **BI** can be obtained by working

[2] For technical reasons, it is necessary to associate the marker ! with 'intuitionistic' variables.

[3] This rule is explained in general in the sequel.

in $\mathbf{Set}^{\mathcal{C}^{op}}$, where, for simplicity, we can take $\mathcal{C}^{op} = \mathcal{M}$ to be a partially-ordered commutative monoid. Given such a monoid $\mathcal{M} = (M, \cdot, e, \sqsubseteq)$, the second semantics of **BI** can be defined via a forcing relation $m \models F$, for a world m and a formula F. All propositions must satisfy the familiar Kripke monotonicity property from intuitionistic logic. The clauses for the linear and intuitionistic implication are then given 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. $r \models \phi \to \psi$ if and only if, for all $s \in M$, if $s \sqsubseteq r$ and $s \models \phi$, then $s \models \psi$.

If we consider a predicate bunched logic, in which the variables in the antecedent have a bunched structure too, then we can form two kinds of quantifiers, a linear one $\forall_{\mathrm{new}}$ and an intuitionistic one $\forall$. The Kripke-style semantics for predicate **BI** can be given by extending the above ideas, and are discussed in [24, 36]. Although presheaf DCCs are adequate for such a semantics of predicate **BI**, they do not yield a good interpretation of proofs. For this, we must move to an indexed or fibred setting in which the predicate **BI** judgement $(X)\Gamma \vdash \phi$ is interpreted by interpreting the propositional judgement $\Gamma \vdash \phi$ over the interpretation of the variable context X.[4]

The relationship between **BI** and the $\lambda\Lambda$-calculus is that the $\lambda\Lambda$-calculus is the dependent type theory which generalizes, in the presence of a certain notion of dereliction, both implications, $\to$ and $\multimap$, and both quantifiers, $\forall$ and $\forall_{\mathrm{new}}$. It is, thus, in a propositions-as-types correspondence with a variant of the $\{\&, \to, \multimap, \forall, \forall_{\mathrm{new}}\}$-fragment of **BI** with Dereliction. It must be stressed, however, that the development of a dependently-typed λ-calculus in proper correspondence with **BI** remains outstanding (though see the discussion in [37]).

The purpose of this paper is to study the categorical semantics of the $\lambda\Lambda$-calculus. This is given here by *Kripke resource models*, which are monoid-indexed sets of functorial Kripke models, $\{\mathcal{J}_r{:}[\mathcal{W}, [\mathcal{C}^{op}, \mathbf{Cat}]] \mid r \in R\}$. The indexing element $r \in R$ can be seen as the resource able to realize the functorial Kripke structure it indexes. We work with indexed category theory, rather than, for example, Cartmell's contextual categories [5], as the indexed approach allows us to separate certain conceptual issues and, hence, allows us to recognize the extra structure needed for studying the model theory of structurally weaker logics and type theories than the intuitionistic ones. We will see this later, in Section 3.2, when we motivate the structure needed to model the $\lambda\Lambda$-calculus.

Kripke resource models generalize, as we might expect, the functorial Kripke models of the $\lambda\Pi$-calculus [35]. These consist of a functor $\mathcal{J}{:}[\mathcal{W}, [\mathcal{C}^{op}, \mathbf{Cat}]]$, where $\mathcal{W}$ is a Kripke world structure, $\mathcal{C}$ is a category with a $(\times, 1)$ cartesian monoidal structure on it and $[\mathcal{C}^{op}, \mathbf{Cat}]$ is a strict indexed category. The intuitionistic dependent function space Π is modelled as right adjoint to the weakening, or projection, functor $p^*{:}\mathcal{J}(W)(D) \to \mathcal{J}(W)(D \times A)$.

In the $\lambda\Lambda$-calculus, we have two kinds of context extension operators, so we require $\mathcal{C}$ to have two kinds of monoidal structure on it, $(\otimes, I)$ and $(\times, 1)$. The intuitionistic dependent function space Π can be modelled, as usual, using the right adjoint to projection. However, there is no similar projection functor corresponding to Λ. For this, we must require the existence of the natural isomorphism $Hom_{\mathcal{J}_{r+r'}(W)(D\otimes A)}(1, B) \cong Hom_{\mathcal{J}_r(W)(D)}(1, \Lambda\, x{:}A\,.\,B)$, where $D \otimes A$ is defined in the $(r + r')$-indexed model. This is sufficient to define the function space.

[4] The two semantics we describe are, of course, formally instances of the same abstract construction. However, we suggest that such a view is conceptually misleading. The forcing semantics is not required to model the typing assertion associating a proof-object to a logical consequence and so admits more immediately a very wide range of conceptual interpretations: a great deal is possible in the category of sets.

While the $\lambda\Lambda$-calculus has familiar soundness and, via a term model, completeness theorems, it is important to ask if there is a natural class of models. For the $\lambda\Pi$-calculus, for instance, the most intuitive concrete model is that of families of sets, $\mathbf{Fam}$. This can be viewed as an indexed category $\mathbf{Fam}{:}[Ctx^{op}, \mathbf{Cat}]$. The base, Ctx, is a small set-theoretic category whose objects are sets and morphisms are set-theoretic functions. For each $D \in obj(\mathbf{Fam})$, $\mathbf{Fam}(D) = \{y \in B(x) \mid x \in D\}$. The fibre is just a discrete category whose objects are the elements of $B(x)$. If $f \in \mathbf{Fam}(C, D)$, then $\mathbf{Fam}(f)$ just re-indexes the set over D to one over C. As there is little structure required in the fibre, the description of families of sets can also be given sheaf-theoretically, as $\mathbf{Fam}{:}[Ctx^{op}, \mathbf{Set}]$, each $\mathbf{Fam}(D)$ being considered as a discrete category. Using Day's construction, we can show how to obtain a corresponding class of set-theoretic models, parameterized on a small monoidal category, for the $\lambda\Lambda$-calculus. That is, we describe a families of sets model in $\mathbf{BIFam}{:}[\mathcal{C}, [Ctx^{op}, \mathbf{Set}]]$, where $\mathcal{C}$ is some small monoidal category. The definition of Π in $\mathbf{BIFam}$ is given as usual, but the definition of Λ uses a restriction of Day's tensor product [8].

The rest of this paper is organized as follows: in Section 2, we present the type theory in both syntactic and algebraic forms. We also make a brief remark regarding the propositions-as-types correspondence between **BI** with Dereliction and the type theory; in Section 3, we describe the Kripke resource models of the type theory; Section 5 concludes the paper by constructing a class of Kripke resource models from the category of families of sets.

This paper, the content of which was sketched in [15], continues the work first reported in Ishtiaq and Pym [17] (see also [16]). The reader is referred to that paper for a full syntactic study of the type theory, together with its use as a language in the logical framework RLF. The ideas presented there were first considered by one of us in [33].

2 The $\lambda\Lambda$-calculus

In this section, we give a syntactic presentation of the type theory, which we henceforth refer to as System $\mathbf{N}$. We briefly comment on the propositions-as-types correspondence with a fragment of predicate **BI**. We also give a algebraic presentation of the type theory, in order to prepare for the completeness argument later.

2.1 A syntactic presentation

The $\lambda\Lambda$-calculus is a first-order dependent type theory with entities at three levels: *objects*, *types* and *families of types*, and *kinds*. Objects (denoted by M,N) are classified by types. Types and families of types (denoted by A, B) are used to represent syntactic classes and judgement forms. Families of types 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.

The abstract syntax of the entities is specified by the following grammar:

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

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

can also be written as $\Pi\, x{:}A\,.\,B$.[5] The notion of linear free and bound variables (LFV, LBV) and substitution may be defined accordingly. When x is not free in B we write $A \multimap B$ and $A \to B$ for $\Lambda\, x{:}A\,.B$ and $\Lambda\, x!A\,.B$, respectively.

We can define the notion of linear occurrence by extending the general idea of occurrence for the λ-calculus [3], though we note that other definitions may be possible.

DEFINITION 2.1 (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 2.2 (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 Girard's sense [4, 2]. However, they seem quite natural in the bunches setting. O'Hearn and Pym, for instance, have examples of **BI** terms — the $\lambda\Lambda$-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 [24].

EXAMPLE 2.3
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 generalize this, so that 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 $\lambda\Lambda$-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:

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

The $\lambda\Lambda$-calculus is a formal system for deriving the following judgements:

$\vdash \Sigma\ \mathsf{sig}$	Σ is a valid signature
$\vdash_\Sigma \Gamma\ \mathsf{context}$	Γ is a valid context in Σ
$\Gamma \vdash_\Sigma K\ \mathsf{Kind}$	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.

[5] Indeed, we could take Π as a primitive, with Λ and Π being connected not by linear logic's Dereliction, which uses ! to convert an extension, $\Gamma, x{:}A$, of a context to an extension $\Gamma; x!A$, but rather by a version of **BI**'s dereliction rule, which allows the inference of an intuitionistic extension, $\Gamma; x!A$, from a linear extension, $\Gamma, x{:}A$. We return to this point in Section 2.7.

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 $\Gamma \vdash_\Sigma K\ \mathsf{Kind}$ or $\Gamma \vdash_\Sigma U{:}V$.

The definition of the type theory depends crucially on several notions to do with the joining and maintenance of contexts; these are the notions of *context joining*, *variable sharing* and *multiple occurrences*. In joining together two contexts to form a third, the order of declarations and type of variables (linear versus intuitionistic) must be respected. 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 and is crucial in allowing the formation of sufficiently complex types. A technical prerequisite to sharing is the notion of multiple occurrences, which allows us to declare contexts of the form $x{:}A, x{:}A$, i.e. contexts in which repeated but distinct declarations of the same variable are possible. These notions will be explicated at appropriate points in this section.

The rules for deriving judgements in the type theory are given in Tables 1 and 2. These are conveniently separated into a linear and an intuitionistic set, the latter directly related to the intuitionistic $\lambda\Pi$-calculus.

TABLE 1. $\lambda\Lambda$-calculus

Valid Signatures

$$\frac{}{\vdash \langle\rangle\ \mathsf{sig}}\,(\Sigma)$$

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

Valid Contexts

$$\frac{\vdash \Sigma\ \mathsf{sig}}{\vdash_\Sigma \langle\rangle\ \mathsf{context}}\,(\Gamma)$$

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

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

Valid Kinds

$$\frac{\vdash_\Sigma \Gamma\ \mathsf{context}}{\Gamma \vdash_\Sigma\ \mathsf{Type}\ \mathsf{Kind}}\,(KAx) \qquad \frac{\Gamma, x{:}A \vdash_\Sigma K\ \mathsf{Kind}}{\Gamma \vdash_\Sigma \Lambda\, x{:}A\,.K\ \mathsf{Kind}}\,(K\Lambda\mathcal{I}1)$$

$$\frac{\Gamma \vdash_\Sigma A{:}\mathsf{Type} \quad \Delta \vdash_\Sigma K{:}\mathsf{Kind} \quad [\Xi';\Gamma;\Delta] \quad \Xi = \Xi' \backslash (lin(\Gamma) \cap lin(\Delta))}{\Xi \vdash_\Sigma A \multimap K{:}\mathsf{Kind}}\,(K\Lambda\mathcal{I}2)$$

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

One of the main points to note about the context formation rules is that a context can be extended with either linear or intuitionistic declarations. There is no zone or 'stoup' separating the linear from the intuitionistic parts of the context. The context formation rules also

TABLE 2. $\lambda\Lambda$-calculus (continued)

Valid Families of Types

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

$$\frac{\Gamma, x{:}A \vdash_\Sigma B{:}\mathsf{Type}}{\Gamma \vdash_\Sigma \Lambda\, x{:}A\,.B \;:\; \mathsf{Type}}\,(A\Lambda\mathcal{I}1) \qquad \frac{\Gamma \vdash_\Sigma A{:}\mathsf{Type} \quad \Delta \vdash_\Sigma B{:}\mathsf{Type} \quad [\Xi';\Gamma;\Delta] \quad \Xi = \Xi'\backslash(lin(\Gamma) \cap lin(\Delta))}{\Xi \vdash_\Sigma A \multimap B{:}\mathsf{Type}}\,(A\Lambda\mathcal{I}2)$$

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

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

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

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

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

Valid Objects

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

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

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

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

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

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

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 exponential variables. A method to join Γ and Δ into Ξ — denoted by $[\Xi; \Gamma; \Delta]$ — is defined in Section 2.2.

In order to motivate the second characteristic of the type theory, consider the following example of a non-derivation:[6]

EXAMPLE 2.4
Let $A!\mathsf{Type}$, $c!A \multimap \mathsf{Type} \in \Sigma$ and note that the argument type, cx, is a dependent one; the linear x is free in it.

$$\dfrac{\dfrac{\dfrac{\vdots}{x{:}A \vdash_\Sigma cx{:}\mathsf{Type}}}{x{:}A,\, z{:}cx \vdash_\Sigma z{:}cx} \quad \dfrac{\vdots}{x{:}A \vdash_\Sigma cx{:}\mathsf{Type}}}{x{:}A \vdash_\Sigma \lambda\, z{:}cx\,.z \;:\; \Lambda\, z{:}cx\,.cx} \quad \dfrac{}{x{:}A,\, y{:}cx \vdash_\Sigma y{:}cx}}{x{:}A,\, x{:}A,\, y{:}cx \vdash_\Sigma (\lambda\, z{:}cx\,.z)\, y \;:\; cx}\,.$$

The problem here is that an excess of linear xs now appear in the combined context after the application step. Our solution is to recognize the xs in each premiss context as *distinct* occurrences of the *same* variable, the one occurring in the argument type cx. The x is said to be shared between the function and its argument. This sharing is implemented via the κ function, which is defined in Section 2.4. Sharing necessitates a binding strategy for multiple occurrences: this is described in Section 2.3. 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.

There are several interesting object-level rules. The two variable declaration rules, $(MVar)$ and $(MVar!)$, declare linear and intuitionistic variables, respectively. These rules should be not seen as weakening in the context Γ as, by induction, the variables declared in Γ are 'used' in the type A. The other interesting set of rules are those for the two function spaces. Consider the introduction rules first. If the context has been extended linearly, then $(M\lambda\Lambda\mathcal{I})$ introduces the linear dependent function space $\Lambda\, x{:}A\,.B$. Otherwise, if the context has been extended intuitionistically, then $(M\lambda\Lambda!\mathcal{I})$ introduces the intuitionistic dependent function space $\Lambda\, x!A\,.B$. In the elimination rules, the side-conditions realize a sharing-sensitive join of the premiss contexts. For the $(M\Lambda!\mathcal{E})$ rule, 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.

2.2 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

[6]Note that the fact that $\Lambda\, z{:}cx\,.cx$ is just $cx \multimap cx$ is of no importance here, the dependency on x being the point of interest.

account of the intuitionistic behaviour of exponential variables. In search, the intuitionistic variable $x!A$ would be sent both ways when the context is split. The rules are given in Table 3.

TABLE 3. Context joining

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

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

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

Further, the context joining relation must respect the ordering of the contexts and the linearity constraint as defined by the binding strategy. 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.

2.3 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 a proposition, as opposed to $x{:}A, y{:}A$, which can be seen as two distinct proofs of the proposition. Though this idea can be seen, in the presence of the binding strategy that we are about to describe, 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 2.4).

The 'leftmost free occurrence of x in U' is the linear occurrence of x in the leftmost subterm of U. The important cases are those for abstraction and application, which are defined as follows:

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

We define the leftmost occurrence of $x{:}A$ in a context Γ as the first declaration of $x{:}A$ in Γ. Similarly, the rightmost occurrence of $x{:}A$ in Γ is the last such declaration. The binding strategy now formalizes the concept of linearity constraint:

DEFINITION 2.5 (Leftmost binding)
Assume $\Gamma, x{:}A, \Delta \vdash_\Sigma U{:}V$ and that $x{:}A$ is the rightmost occurrence of x in the context. Then x binds:

1. the first leftmost occurrence of x in $ran(\Delta)$, if there is such a declaration;

2. the unbound leftmost linear occurrences of x in $U{:}V$.

There is no linearity constraint for intuitionistic variables: the rightmost 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 $(M\lambda\Lambda\mathcal{I})$ rule, the $\lambda(\Lambda)$ binds the leftmost occurrence of x in $M(B)$. Similarly, in the (admissible) cut rule, the term $N{:}A$ cuts with the leftmost 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 leftmost binding and cutting strategy as discussed above. We remark that there is a general ij strategy, that of binding the ith variable from the left and cutting the jth variable from the left.

EXAMPLE 2.6
If $b, c : A \multimap \mathsf{Type} \in \Sigma$ and $a\,!\,\Lambda x{:}A.bx \multimap cx \multimap \mathsf{Type} \in \Sigma$, then

$$x{:}A, y{:}bx, x{:}A, z{:}cx \vdash_\Sigma axyz{:}\mathsf{Type}.$$

We return to this example and give a proof of the typing in Example 2.9.

Though we can construct such terms, the main motivation for multiple occurrences is to introduce the notion of sharing.

2.4 Variable sharing

Variable sharing is a central notion which allows linear dependency to be set up. In fact, this notion is already implicit in Definition 2.1 of linear occurrence. Sharing occurs when linear variables are needed for the well-formedness of the premiss types but not necessarily for the well-formedness of the conclusion type. This requirement is regulated by a function κ.

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 2.7
The function κ is defined for the binary, multiplicative $(A\Lambda\mathcal{E})$, $(M\Lambda\mathcal{E})$ and (Cut) rules

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

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

For each $x{:}A$ occurring in both Γ and Δ, construct from right to left as follows: (Formally, $\kappa(\Gamma, \Delta)$ is defined recursively on the structure of Γ and Δ, read from right to left. We adopt the following informal notation for ease of expression.)

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

$$\kappa(\Gamma,\Delta) \;=\; \begin{cases} \{x{:}A \in lin(\Gamma)\cap lin(\Delta) \mid \text{either } (i)\ there\ is\ no\ y{:}B(x) \\ \qquad\qquad to\ the\ right\ of\ x{:}A\ in\ \Gamma \\ \qquad\text{or } (ii)\ there\ is\ no\ y{:}B(x) \qquad otherwise. \\ \qquad\qquad to\ the\ right\ of\ x{:}A\ in\ \Delta \\ \qquad\text{or both } (i)\ and\ (ii)\} \end{cases}$$

The second 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.

We should emphasize that κ relies on the formation of a set, rather than a multiset, of variables. It might be that alternative definitions for variable sharing are possible.

In the absence of sharing of variables, when the first clause only applies, we have a linear dependent function space but without the dependency of the abstracting A_is on the previously abstracted variables.

Given the definition, we can now consider the following example, which corrects Example 2.4:

EXAMPLE 2.8
Suppose $A!\mathsf{Type}, c!A \multimap \mathsf{Type} \in \Sigma$. Then we construct the following:

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

The † denotes the context join to get $x{:}A$. The †† side-condition is more interesting. Firstly, 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.

The function κ is not required, i.e. its use is vacuous, when certain restrictions of the $\lambda\Lambda$-calculus type theory are considered. For instance, if we restrict type-formation to be entirely intuitionistic so that type judgements are of the form $!\Gamma \vdash_\Sigma A{:}\mathsf{Type}$, then we recover the $\{\Pi, \multimap, \&\}$-fragment of Cervesato and Pfenning's $\lambda^{\Pi\multimap\&\top}$ type theory [6].

We conclude this section by completing Example 2.6, of a linear dependent type formed using the notions of multiple occurrences and variable sharing.[7]

[7]There is an erroneous claim in [15, Section 2.2], corrected in [16]. The term cxx is indeed a valid one but it does not require mutliple occurences of x. Example 2.9, given here, is correct.

EXAMPLE 2.9
If $b, c : A \multimap \mathsf{Type} \in \Sigma$ and $a\,!\,\Lambda x{:}A\,.\,bx \multimap cx \multimap \mathsf{Type} \in \Sigma$, then we can construct the following proof:

$$\dfrac{\dfrac{\dfrac{\vdash_\Sigma a : \Lambda x{:}A\,.bx \multimap cx \multimap \mathsf{Type} \quad x{:}A \vdash_\Sigma x{:}A}{x{:}A \vdash_\Sigma ax{:}bx \multimap cx \multimap \mathsf{Type}} \quad \dfrac{x{:}A \vdash_\Sigma bx{:}\mathsf{Type}}{x{:}A, y{:}bx \vdash_\Sigma y{:}bx}}{x{:}A, y{:}bx \vdash_\Sigma z{:}cx \multimap \mathsf{Type}} \quad \dfrac{x{:}A \vdash_\Sigma cx{:}\mathsf{Type}}{x{:}A, z{:}cx \vdash_\Sigma z{:}cx}}{x{:}A, y{:}bx, x{:}A, z{:}cx \vdash_\Sigma axyz{:}\mathsf{Type}}\,.$$

The last two applications have a non-trivial κ action which forces one of the $x{:}A$s to be shared. It can be checked that all the constants used in the proof are well-typed.

2.5 *Definitional equality*

The definitional equality relation that we consider here is the $\beta\eta$-conversion of terms at all three levels, subject to the binding strategy. The parallel nested reduction form of $\beta\eta$-reduction is written as $\rightarrow$. The transitive closure of $\rightarrow$ is denoted by $\rightarrow^*$. The definitional equality relation, $\equiv$, between terms at each respective level is defined to be the symmetric and transitive closure of $\rightarrow$. The one-step reduction relation is written as $\rightarrow_1$.

The relation, subject to the binding strategy, is given by the rules in Table 4. We include just the rules for β-reduction; the rules for η-reduction follow the usual pattern [10, 7, 39], e.g.

$$\dfrac{\Gamma \vdash_\Sigma \lambda\, x{\in}A\,.M\,x : B \quad \Gamma \vdash_\Sigma M : C \quad x \notin \mathrm{FV}(M)}{\Gamma \vdash_\Sigma \lambda\, x{\in}A\,.M\,x \equiv M}\,.$$

This concludes the syntactic presentation of the type theory. We refer to this presentation as system $\mathbf{N}$. We will write $\mathbf{N}$ proves $\Gamma \vdash_\Sigma M{:}A$, etc. to mean that the assertion $\Gamma \vdash_\Sigma M{:}A$, etc. is derivable in the system $\mathbf{N}$. A term is said to be *well-typed* or *valid* in a signature and context if it can be shown to either be a kind, have a kind, or have a type in that signature and context. We speak similarly of valid contexts relative to a signature and of valid signatures.

2.6 *Basic properties*

A summary of the major meta-theorems pertaining to system $\mathbf{N}$ and its reduction properties are given by the following theorem:

THEOREM 2.10 (Basic metatheory of the $\lambda\Lambda$-calculus)

1. (Church–Rosser) All well-typed terms are Church–Rosser.
2. (Structural Admissibilities) Exchange, weakening, dereliction, contraction and (two forms of) cut are admissible.
3. (Unicity of Types and Kinds) If $\mathbf{N}$ proves $\Gamma \vdash_\Sigma U{:}V$ and $\mathbf{N}$ proves $\Gamma \vdash_\Sigma U{:}V'$, then $V \equiv V'$.
4. (Extended Unicity of Domains) If $\mathbf{N}$ proves $\lambda\, x{\in}A\,.U$ inhabits $\Lambda\, x{\in}B\,.V$, then $A \equiv B$.
5. (Subject Reduction) If $\mathbf{N}$ proves $\Gamma \vdash_\Sigma U{:}V$ and $U \rightarrow_1 U'$, then $\mathbf{N}$ proves $\Gamma \vdash_\Sigma U'{:}V$.
6. (Strong Normalization) If $\mathbf{N}$ proves $\Gamma \vdash_\Sigma U{:}V$, then U is strongly normalizing.
7. (Predicativity) If $\mathbf{N}$ proves $\Gamma \vdash_\Sigma M{:}A$, then $erase(M)$ (the type erasure of $M{:}A$) can be typed in the Curry type-assignment system.

TABLE 4. Parallel nested reduction

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

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

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

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

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

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

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

8. (Decidability) All assertions of the $\lambda\Lambda$-calculus are decidable.

The proof of this theorem, presented in [17], is obtained by adapting the techniques of Harper *et al.* [10] to this setting. The proof of the Church–Rosser property is shown by proving confluence for the one-step reduction relation and then inducting on the number of reduction steps. The proof of strong normalization is by giving a 'dependency- and linearity-less' translation of the $\lambda\Lambda$-calculus into the Curry-typable untyped λ-calculus. The translation is faithful and consistent and allows us to 'reflect' the strong normalization property of the λ-calculus back to the $\lambda\Lambda$-calculus.

We use this technique, of giving a translation to prove a property, to obtain Church–Rosser for η-reduction. This was the main difficulty in Harper *et al.*'s metatheoretic study of the $\lambda\Pi$-calculus. One solution, due to Salvesen, is to use van Daalen's technique of label-conversion [39]. We exploit that result by giving a faithful and consistent translation of the $\lambda\Lambda$-calculus into the $\lambda\Pi$-calculus and appealing to the reduction properties of the latter.

2.7 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 also very close to **BI**, the logic of bunched implications. In **BI**, we have two kinds of function spaces, the linear one $-\!\circ$ and the intuitionistic one $\to$. Correspondingly, there are 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 weaken-

ing and contraction, and the ',', which doesn't. We can add the rule of Dereliction, the full propositional form being

$$\frac{\Gamma(\Delta, \Delta') \vdash \phi}{\Gamma(\Delta; \Delta') \vdash \phi},$$

to **BI**; this allows consequences such as $\phi \multimap \psi \vdash \phi \to \psi$ to be provable. Neither **BI**'s nor the $\lambda\Lambda$-calculus' rule for Dereliction relies on the presence of a general $!$ modality.

It follows that the $\lambda\Lambda$-calculus stands in propositions-as-types with the $\{\&, \to, \multimap, \forall, \forall_{\mathrm{new}}\}$-fragment of **BI** with the rule of Dereliction, the full predicate form being

$$\frac{(X(Y, Y'))\Gamma(\Delta, \Delta') \vdash \phi}{(X(Y; !Y'))\Gamma(\Delta; !\Delta') \vdash \phi}\, D\,,$$

where each $!Z$ denotes Z with each $x{:}A$ replaced by $x!A$ and each ',' replaced by ';', but without the unit operation taking a bunch $\Delta(\Gamma)$ to $\Delta(I, \Gamma)$.[8] This operation changes the status of Γ within a derivation, so that a variable which starts out in additive combination with its neighbours can bind multiplicatively. No corresponding operation is possible in the $\lambda\Lambda$-calculus, so that this proposition-as-types correspondence does not properly generalize that for propositional **BI** and its associated simply-typed λ-calculus, $\alpha\lambda$. The addition of Dereliction represents the extent to which the $\lambda\Lambda$-calculus corresponds to linear logic: the basic context-extension operation in $\lambda\Lambda$ adds a type to the right-hand end of a context, $\Gamma, x{:}A$ or $\Gamma, x!A$, so that, in the given fragment, the relationship between these two cases may be seen either as an instance of **BI**'s Dereliction or as an instance of linear logic's Dereliction.[9]

The basic idea for the correspondence between **BI** and the $\lambda\Lambda$-calculus is to consider ';' as intuitionistic extension and ',' as linear extension. This is implemented by giving a translation of **BI** contexts which relies, to a certain extent, on the notion of dereliction. The idea of viewing the **BI** context joining connectives as context extension operators necessarily restricts the correspondence to a fragment, though a non-trivial one, of **BI**. The correspondence between the connectives is given by the following table:

BI	$\lambda\Lambda$
$\wedge$	$\&$
$\to$	$\to$
$\multimap$	$\multimap$
$\forall$	$\Lambda{-}!{-}$
$\forall_{\mathrm{new}}$	$\Lambda{-} : {-}$

In fact, as mentioned in Section 2.1, we could take the type-constructor Π as a primitive, with the relationship between Λ and Π being given by **BI**'s dereliction rule.

One view of this correspondence is, then, that the RLF meta-logic uses this fragment of **BI**, just as the LF meta-logic uses the $\{\to, \forall\}$-fragment of Intuitionistic Logic. A detailed account of the correspondence is given in [13, 16].

It remains a challenging and open problem to give a systematic analysis of the relationship between substructural logics and dependent type theories. In particular, it remains to formu-

[8] Though note that, for the purposes of completeness later, we will take both units in the type theory; in our present context, this is a minor matter.

[9] This observation suggests a way to have both additive and multiplicative quantifiers in linear logic. The basic quantifier would be the multiplicative, or linear, one and the additive, or intuitionistic, one would be recovered via the exponential. See [37] for more discussion of this point.

late a dependent type theory in correspondence to a proper fragment of **BI**, i.e. in which the structural rules are unaltered.

2.8 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 objects 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 arises.

DEFINITION 2.11
Let Σ be a signature. The base category $\mathcal{C}(\Sigma)$ of contexts and realizations is defined as follows:

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

The following proposition follows easily from the definition:

PROPOSITION 2.12
$\mathcal{C}(\Sigma)$ is a category.

PROOF. We must check that 1_Δ is an identity morphism and that composition is associative. We omit the details. ∎

In the judgements $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$ and $\Gamma \vdash_\Sigma M{:}A$, the context Γ, which is an object of $\mathcal{C}(\Sigma)$ according to Definition 2.11, 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 $\mathcal{C}(\Sigma)$ to be the base for the following.

DEFINITION 2.13
We inductively define a strict indexed category $\mathcal{E}(\Sigma)$ over the base category $\mathcal{C}(\Sigma)$

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

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

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

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

We remark that each $\mathcal{C}(\Sigma)(\Gamma)$ is a category. Note that the identity arrow $A \stackrel{1}{\to} A$ over Γ is given by the term $\lambda\, x{:}A\,.x$, corresponding to the definition of morphisms above. To see that this construction is correct, consider that the axiom sequent is of the form $\Gamma, x{:}A \vdash_\Sigma x{:}A$, with the side-condition that $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$, thereby using the variables in Γ.

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 $\mathbf{N}$.

THEOREM 2.14 (Classifying category)
Let Σ be a signature and let Γ, M and A be in $\alpha\beta\eta$-normal form.

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

PROOF. By induction: in the forward direction on the structure of the proofs in $\mathbf{N}$; and in the backward direction on the complexity of expressions. The arguments are straightforward and we omit the details. ∎

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

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 $\{\to, \Pi\}$-fragment of the $\lambda\Lambda$-calculus (i.e. the $\lambda\Pi$-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 [42].

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

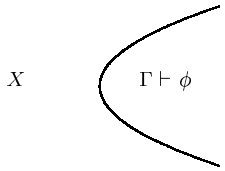
A similar pair of clauses defines the forcing relation for the two **BI** quantifiers. Here $D{:}\mathcal{M}^{op} \to \mathbf{Set}$ is a domain of individuals and $u \in [\![X]\!]r$ is an environment appropriate to the bunch of variables X at world r, where $[\![X]\!]$ is the interpretation of the bunch of variables X in $\mathbf{Set}^{\mathcal{M}^{op}}$:

1. $(X)u, r \models \forall x.\phi$ if and only if, for all $r \sqsubseteq s$ and all $d \in Ds$,
$$(X; x)([\![X]\!](r \sqsubseteq s)u, d), s \models \phi\,;$$
2. $(X)u, r \models \forall_{\mathrm{new}} x.\phi$ if and only if, for all s and all $d \in Ds$,
$$(X, x)[u, d], r \cdot s \models \phi\,.$$

Here $(-,-)$ 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 [19, 42]. Further details are in [24, 36].

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. Firstly, 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 $\{\to, \forall\}$-fragment of the internal logic and so correspond to Kripke-style models for intuitionistic logic. Secondly, we index $[\mathcal{W}, \mathcal{E}]$ by a resource monoid R. Thus, we obtain R-indexed sets of Kripke functors $\{\mathcal{J}_r{:}[\mathcal{W}, \mathcal{E}] \mid r \in R\}$. We remark that the separation of worlds from resources considered in this structure emphasizes a sort of 'phase shift' [9, 12]. We briefly reconsider this choice in Section 5.

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, of propositions over the context, X, of variables. 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 [25]. More specifically, in the case of the type theory, the judgement $\Gamma \vdash_\Sigma M{:}A$ is modelled as the arrow $1 \stackrel{[\![M]\!]}{\to} [\![A]\!]$ in the fibre over $[\![\Gamma]\!]$ in the strict indexed category $\mathcal{E}{:}\mathcal{C}^{op} \to \mathbf{Cat}$.

We remark that this is not the only technique for modelling a typing judgement; Cartmell [5], Pitts [26] 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 $\mathcal{C}$ 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.

DEFINITION 3.1 (Doubly monoidal category)
A *doubly monoidal category* is a category $\mathcal{C}$ equipped with two monoidal structures, $(\otimes, I)$ and $(\times, 1)$. $\mathcal{C}$ is called *cartesian doubly monoidal* if $\times$ is cartesian. We will use $\bullet$ to range over both multiplications.

There are a couple of comments we need to make about the monoidal structure on $\mathcal{C}$. Firstly, there is no requirement that the bifunctors $\otimes$ and $\times$ be symmetric, as the contexts which 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 indeed 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 $\mathcal{C}$, as A by itself may not correspond to a well-formed type. For modelling the judgement $\Gamma, x{\in}A \vdash_\Sigma x{:}A$, we do, however, require the existence of a map $1 \xrightarrow{q} [\![A]\!]$ in the fibre over $[\![\Gamma]\!] \bullet [\![A]\!]$.

The interaction between projection and other maps in $\mathcal{C}$ is stated by requiring the following pullback in $\mathcal{C}$:

$$\begin{array}{ccc} D \times f^*(A) & \xrightarrow{f \times A} & E \times A \\ {\scriptstyle p_{D,f^*A}}\downarrow & & \downarrow{\scriptstyle p_{E,A}} \\ D & \xrightarrow[f]{} & E \end{array}$$

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 $D \xrightarrow{\langle 1_D \otimes M \rangle} D \otimes A$ in $\mathcal{C}$, the tuples being given by the bifunctoriality of $\otimes$.

A doubly monoidal category $\mathcal{C}$ with both exponentials or, alternatively, $\mathcal{C}$ 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 [24]. DCCs provide a class of models of **BI** in which both function spaces are modelled within $\mathcal{C}$. We will work with the barer doubly-monoidal category, requiring some extra structure on the fibres to model the function space. This can be seen as a natural extension to the semantics of bunches to account for dependency. It can be contrasted to the Barber–Plotkin model of DILL [2], which uses a pair of categories, a monoidal one and a cartesian one, together with a monoidal adjunction between them. However, such a model

forces too much of a separation between the linear and intuitionistic parts of a context to be of use to us.

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 $\Lambda_{D,A}$,

$$\frac{Hom_{\mathcal{J}_{r+r'}(W)(D\bullet A)}(1, B)}{Hom_{\mathcal{J}_r(W)(D)}(1, \Lambda_{D,A}\, x{\in}A\,.B)}$$

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 $1 \stackrel{[\![M]\!]}{\to} [\![A]\!]$ 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.

The type theory also contains an additive conjunction connective, &. This is modelled by requiring each category $\mathcal{J}_r(W)(D)$ to have products.

DEFINITION 3.2
Let $(R, +, 0)$ be a commutative monoid (of 'resources'). A *Kripke resource* $\lambda\Lambda$*-structure* is an R-indexed set of functors

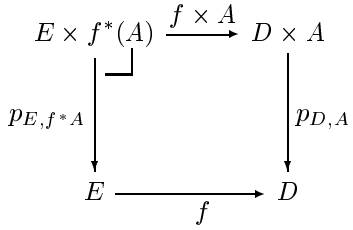
$$\{\mathcal{J}_r \,:\, [\mathcal{W}, [\mathcal{C}^{op}, \mathbf{Cat}]] \mid r \in R\},$$

where $\langle \mathcal{W}, \leq \rangle$ is a poset, $\mathcal{C}^{op} = \amalg_{W\in\mathcal{W}} \mathcal{C}^{op}_W$, where $W \in \mathcal{W}$ and each $\mathcal{C}_W$ is a small doubly monoidal category, with $1 \ncong I$,[10] 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 $f{:}E \to D \in \mathcal{C}_W$.
2. For each $W \in \mathcal{W}$, $D \in \mathcal{C}_W$ and object $A \in \mathcal{J}_r(W)(D)$ there is a $D \bullet A \in \mathcal{C}_W$.

   For the cartesian extension, there are canonical first projections $D \times A \stackrel{p_{D,A}}{\to} D$ and canonical pullbacks

[10] Note that [15] has a bad typographical error, corrected in [16]: It has $1 \cong I$ where it should have $1 \ncong I$.



The pullback indicates, for the cartesian case, how to interpret realizations as tuples. In particular, for each $1 \stackrel{M}{\to} A \in \mathcal{J}_r(W)(D)$, there exists a unique arrow $D \stackrel{\langle 1 \times M \rangle}{\to} D \times A$. It does not cover the case for the monoidal extension. For that, we require there to exist a unique $D(= D \otimes I) \stackrel{\langle 1 \otimes M \rangle}{\to} D \otimes A$, the tuples being given by the bifunctoriality of $\otimes$.

For both extensions, there is a canonical second projection $1 \stackrel{q_{D,A}}{\to} A$ 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 $F \stackrel{g}{\to} E$ and $E \stackrel{f}{\to} D$ in $\mathcal{C}_W$. 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 $\Lambda_{D,A}$,

$$\frac{Hom_{\mathcal{J}_{r+r'}(W)(D \bullet A)}(1, B)}{Hom_{\mathcal{J}_r(W)(D)}(1, \Lambda_{D,A}\, x \in A\, .B)}\ ,$$

in which the extended context is defined in the $r + r'$-indexed functor. This natural isomorphism is required to satisfy the Beck-Chevalley condition: For each $E \stackrel{f}{\to} D$ 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.

Our approach is modular enough to also provide a categorical semantics for the intuitionistic fragment of the $\lambda\Lambda$-calculus, the $\lambda\Pi$-calculus. For that, we work with a Kripke $\lambda\Pi$-structure which consists of a single functor $\mathcal{J}_r{:}[\mathcal{W}, [\mathcal{D}^{op}, \mathbf{Cat}]]$, where $\mathcal{D}$ is (essentially) a category with only the (modified) cartesian structure $(\times, 1)$ on it [35]. The definition of Π as right adjoint to weakening can be recovered from the natural isomorphism.

This can be seen in the Lemma 3.3, which is motivated by the propositions-as-types correspondence discussed earlier. 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 just that given by $\Pi_{D,A}(B)$ in the $\lambda\Pi$-structure case. Such a result 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 (essentially) a category equipped with just the (modified) cartesian closed structure, plus the usual coherence conditions.

LEMMA 3.3
The natural isomorphism

$$cur_W \ : \ Hom_{\mathcal{J}(W)(D \bullet A)}(p^*_{D,A}(1), B) \quad \cong \quad Hom_{\mathcal{J}(W)(D)}(1, \Pi_{D,A}(B)) \ : \ cur^{-1}_W,$$

in the Kripke λΠ-structure $\mathcal{J}$, is just the $\Lambda_{D,A}$ natural isomorphism in the $D \times A$ case in the Kripke resource λΛ-structure.

PROOF. Fix a $\mathcal{J}_r$ to work in. Then define a translation, !, from the λΛ-structure to the λΠ-structure. Informally, the translation can be seen as follows:

$$\mathcal{J}_r(W)(A \bullet \ldots \bullet B) \quad \overset{1 \xrightarrow{f} D}{\frown} \qquad \mapsto \qquad \mathcal{J}(W)(A' \times \ldots \times B') \quad \overset{1 \xrightarrow{f'} D'}{\frown}$$

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 is to forget the linear-intuitionistic context extension operator difference, translating both to the context extension operator $\times$. The units are dealt with similarly: both are translated to the $\mathcal{D}$ unit context 1. The translation has some similarity with Girard's translation of $- \rightarrow -$ into $!- \multimap -$ [9].

To show that the natural isomorphisms are the same, we start with the conclusion of the natural isomorphism $\Lambda_{D,A}$ and compute:

$$\dfrac{\dfrac{\dfrac{Hom_{\mathcal{J}_r(W)(D)}(1, \Lambda\, x!A\,.B)}{Hom_{\mathcal{J}(W)(!D)}(1, \Pi_{!D,!A}(!B))}\,!}{}}{Hom_{\mathcal{J}(!W)(!D \times !A)}(p^*_{!D,!A}(1), (!B))}\, cur^{-1},$$

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 $\mathcal{C}$. ∎

Syntactically, the Lemma 3.3 should be seen as a translation from the λΛ-calculus to the λΠ-calculus (and so the reverse of the translation used to show the strong normalization part of Theorem 2.10). More semantically, it should perhaps be seen in a 2-categorical setting. The statement of the lemma would be that in some category which has those structures as its objects, one should be able to construct an arrow from a λΛ-structure to a λΠ-structure.

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 $\mathcal{C}$-functors, and a satisfaction relation. Except for the structure, the components are defined, due to interdependences,

simultaneously by induction on raw syntax. This explains the long and complex formal definition of the model (below).

Ignoring these interdependencies for a moment, we explain the purpose of each component of the model. Firstly, the Kripke resource structure provides the abstract domain in which the type theory is interpreted. The Σ-operations provide the points to interpret constants in the signature. Secondly, the interpretation $[\![-]\!]$ is a partial function mapping raw (that is, not necessary well-formed) contexts Γ to objects of $\mathcal{C}$, 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. Thirdly, the $\mathcal{C}$-functors maintain the well-formedness of contexts with regard to joining and sharing. The model must also be constrained so that multiple occurrences of variables in the context get the same interpretation. Finally, 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 [43]. In our case, the definition is given more directly. More remarks on the definition of a model will be in order following 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 [41]. The analogous development in our setting is similar and we omit the details.

DEFINITION 3.4
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 $\mathcal{C}$-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

$$[\![\Xi]\!]^W_{\mathcal{J}_r} = share\ join([\![\Gamma_m]\!]^W_{\mathcal{J}_m}, \ldots, share\ join([\![\Gamma_1]\!]^W_{\mathcal{J}_r}, [\![!\Phi]\!]^W_{\mathcal{J}_r}) \ldots) ;$$

   (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

$$\mathcal{J}_r(W)([\![!\Phi, x_1{\in}A_1, \ldots, x_m{\in}A_m]\!]^W_{\mathcal{J}_r})$$

   an arrow $1_{\mathcal{J}_r(W)(D)} \xrightarrow{op_c} [\![A]\!]^W_{\mathcal{J}_r}$, where $D = [\![!\Phi, x_1{\in}A_1, \ldots, x_m{\in}A_m]\!]^W_{\mathcal{J}_r}$.

2. An *interpretation* $[\![-]\!]^{-}_{\mathcal{J}_r}$, in each such $\mathcal{J}_r$, satisfies, at each W:

   (a) $[\![\langle\rangle]\!]^W_{\mathcal{J}_r} \simeq 1_{\mathcal{C}}$;

   (b) $[\![\Gamma, x{:}A]\!]^W_{\mathcal{J}_{r+r'}} \simeq [\![\Gamma]\!]^W_{\mathcal{J}_r} \otimes [\![A_\Gamma]\!]^W_{\mathcal{J}_{r'}}$;

   (c) $[\![\Gamma, x{!}A]\!]^W_{\mathcal{J}_r} \simeq [\![\Gamma]\!]^W_{\mathcal{J}_r} \times [\![A_\Gamma]\!]^W_{\mathcal{J}_r}$;

   (d) $[\![\Gamma \xrightarrow{\langle M_1,\ldots,M_n\rangle} \Delta]\!]^W_{\mathcal{J}_r} \simeq [\![\Gamma]\!]^W_{\mathcal{J}_r} \xrightarrow{\langle [\![M_{1\,\Gamma_1}]\!]^W_{\mathcal{J}_1},\ldots,[\![M_{n\,\Gamma_n}]\!]^W_{\mathcal{J}_n}\rangle} [\![\Delta]\!]^W_{\mathcal{J}_r}$,
   where $[\![\Gamma]\!]^W_{\mathcal{J}_r} = share\ join([\![\Gamma_n]\!]^W_{\mathcal{J}_n}, \ldots, share\ join([\![\Gamma_2]\!]^W_{\mathcal{J}_2}, [\![\Gamma_1]\!]^W_{\mathcal{J}_1})\ldots)$;

   (e) $[\![\langle\rangle_\Gamma]\!]^W_{\mathcal{J}_r} \simeq 1_{[\![\Gamma]\!]^W_{\mathcal{J}_r}}$;

   (f) $[\![(cM_1 \ldots M_n)_\Gamma]\!]^W_{\mathcal{J}_r} \simeq op_c([\![M_{1\,\Gamma_1}]\!]^W_{\mathcal{J}_{r_1}}, \ldots, [\![M_{n\,\Gamma_n}]\!]^W_{\mathcal{J}_{r_n}})$ in $\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})$,
   where $[\![\Gamma]\!]^W_{\mathcal{J}_r} = share\ join([\![\Gamma_n]\!]^W_{\mathcal{J}_n}, \ldots, share\ join([\![\Gamma_2]\!]^W_{\mathcal{J}_2}, [\![\Gamma_1]\!]^W_{\mathcal{J}_1})\ldots)$;

   (g) $[\![\Lambda\, x{:}A\,.B_\Gamma]\!]^W_{\mathcal{J}_r} \simeq \Lambda_{[\![\Gamma]\!]^W_{\mathcal{J}_r},[\![A_\Gamma]\!]^W_{\mathcal{J}_s}}([\![B_{\Gamma,x:A}]\!]^W_{\mathcal{J}_{r+s}})$, where the extended context is defined in the $r+s$-indexed model;

   (h) $[\![\Lambda\, x{!}A\,.B_\Gamma]\!]^W_{\mathcal{J}_r} \simeq \Lambda_{[\![\Gamma]\!]^W_{\mathcal{J}_r},[\![A_\Gamma]\!]^W_{\mathcal{J}_r}}([\![B_{\Gamma,x!A}]\!]^W_{\mathcal{J}_r})$;

   (i) $[\![A\&B_\Gamma]\!]^W_{\mathcal{J}_r} \simeq [\![A_\Gamma]\!]^W_{\mathcal{J}_r} \times [\![B_\Gamma]\!]^W_{\mathcal{J}_r}$;

   (j) $[\![\langle\rangle_\Gamma]\!]^W_{\mathcal{J}_r} \simeq 1_{[\![\Gamma]\!]^W_{\mathcal{J}_r}}$;

   (k) $[\![c_{!\Gamma}]\!]^W_{\mathcal{J}_r} \simeq \Lambda^m(op_c)$;

   (l) $[\![x_{\Gamma,x:A}]\!]^W_{\mathcal{J}_r} \simeq q_{[\![\Gamma,x:A]\!]^W_{\mathcal{J}_r}}$;

   (m) $[\![\lambda\, x{:}A\,.M_\Gamma]\!]^W_{\mathcal{J}_r} \simeq \Lambda_{[\![\Gamma]\!]^W_{\mathcal{J}_r},[\![A_\Gamma]\!]^W_{\mathcal{J}_r}}([\![M_{\Gamma,x:A}]\!]^W_{\mathcal{J}_r})$;

   (n) $[\![\lambda\, x{!}A\,.M_\Gamma]\!]^W_{\mathcal{J}_r} \simeq \Lambda_{[\![\Gamma]\!]^W_{\mathcal{J}_r},[\![!A_\Gamma]\!]^W_{\mathcal{J}_r}}([\![M_{\Gamma,x!A}]\!]^W_{\mathcal{J}_r})$;

   (o) $[\![MN_\Xi]\!]^W_{\mathcal{J}_t} \simeq (\langle 1_{[\![\Gamma]\!]^W_{\mathcal{J}_r}}, [\![N_\Delta]\!]^W_{\mathcal{J}_s}\rangle^*(\Lambda^{-1}_{[\![\Gamma]\!]^W_{\mathcal{J}_r},[\![A_\Gamma]\!]^W_{\mathcal{J}_s}}([\![M_\Gamma]\!]^W_{\mathcal{J}_r})))$,
   where $[\![\Xi']\!]^W_{\mathcal{J}_{r+s}} = join\langle [\![\Gamma]\!]^W_{\mathcal{J}_r}, [\![\Delta]\!]^W_{\mathcal{J}_s}\rangle$ and $[\![\Xi]\!]^W_{\mathcal{J}_t} = share([\![\Xi']\!]^W_{\mathcal{J}_{r+s}})$;

   (p) $[\![\langle M, N\rangle_\Gamma]\!]^W_{\mathcal{J}_r} \simeq \langle [\![M_\Gamma]\!]^W_{\mathcal{J}_r}, [\![N_\Gamma]\!]^W_{\mathcal{J}_r}\rangle$;

   (q) $[\![\pi_i(M)_\Gamma]\!]^W_{\mathcal{J}_r} \simeq \pi_i([\![M_\Gamma]\!]^W_{\mathcal{J}_r})$, where $i \in \{0,1\}$.

   Otherwise the interpretation is undefined.

3. There exists a bifunctor $join$ on $\mathcal{C}$. 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:

$$\begin{array}{rcl} join\langle [\![\langle\rangle]\!]^W_{\mathcal{J}_r}, [\![\langle\rangle]\!]^W_{\mathcal{J}_r}\rangle & = & [\![\langle\rangle]\!]^W_{\mathcal{J}_r} \\ join\langle [\![\Gamma]\!]^W_{\mathcal{J}_r}, [\![\Delta, x{:}A]\!]^W_{\mathcal{J}_{s+t}}\rangle & = & join\langle [\![\Gamma]\!]^W_{\mathcal{J}_r}, [\![\Delta]\!]^W_{\mathcal{J}_s}\rangle \otimes [\![x{:}A]\!]^W_{\mathcal{J}_t} \\ join\langle [\![\Gamma, x{:}A]\!]^W_{\mathcal{J}_{r+t}}, [\![\Delta]\!]^W_{\mathcal{J}_s}\rangle & = & join\langle [\![\Gamma]\!]^W_{\mathcal{J}_r}, [\![\Delta]\!]^W_{\mathcal{J}_s}\rangle \otimes [\![x{:}A]\!]^W_{\mathcal{J}_t} \\ join\langle [\![\Gamma, x{!}A]\!]^W_{\mathcal{J}_r}, [\![\Delta, x{!}A]\!]^W_{\mathcal{J}_s}\rangle & = & join\langle [\![\Gamma]\!]^W_{\mathcal{J}_r}, [\![\Delta]\!]^W_{\mathcal{J}_s}\rangle \times [\![x{!}A]\!]^W_{\mathcal{J}_{r+s}}. \end{array}$$

   The definition of $join$ on morphisms is similar. It is easy to see that $[\![\Xi]\!]^W_{\mathcal{J}_{r+s}} = join\langle [\![\Gamma]\!]^W_{\mathcal{J}_r}, [\![\Delta]\!]^W_{\mathcal{J}_s}\rangle$.

   There exists a functor $share$ on $\mathcal{C}$. The purpose of $share$ is to regulate sharing of multiple occurrences of an object. The definition of $share$ on objects is as follows:

$$
\begin{array}{rcl}
share(\llbracket\langle\rangle\rrbracket^W_{\mathcal{J}_r}) & = & \llbracket\langle\rangle\rrbracket^W_{\mathcal{J}_r} \\
share(\llbracket\Xi\rrbracket^W_{\mathcal{J}_r}) & = & join\langle share(\llbracket\Phi\rrbracket^W_{\mathcal{J}_{s+s'}}), \llbracket\Psi\rrbracket^W_{\mathcal{J}_{t+u+u'}}\rangle \\
 & & \quad \text{if } \llbracket\Xi\rrbracket^W_{\mathcal{J}_t} = join\langle\llbracket\Gamma, x{:}A, \Gamma'\rrbracket^W_{\mathcal{J}_{s+t+u}}, \llbracket\Delta, x{:}A, \Delta'\rrbracket^W_{\mathcal{J}_{s'+t+u'}}\rangle \\
 & & \quad\quad \nexists y{:}B(x) \in \Gamma', \Delta' \\
 & & \quad\quad \llbracket\Phi\rrbracket^W_{\mathcal{J}_{s+s'}} = join\langle\llbracket\Gamma\rrbracket^W_{\mathcal{J}_s}, \llbracket\Delta\rrbracket^W_{\mathcal{J}_{s'}}\rangle \\
 & & \quad\quad \llbracket\Psi\rrbracket^W_{\mathcal{J}_{t+u+u'}} = join\langle\llbracket x{:}A\rrbracket^W_{\mathcal{J}_t}, join\langle\llbracket\Gamma'\rrbracket^W_{\mathcal{J}_u}, \llbracket\Delta'\rrbracket^W_{\mathcal{J}_{u'}}\rangle\rangle \\
 & = & \llbracket\Xi\rrbracket^W_{\mathcal{J}_r} \text{ otherwise.}
\end{array}
$$

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. *Satisfaction* 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 $\llbracket\Gamma, x{\in}A\rrbracket^W_{\mathcal{J}_r}$ 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 $\llbracket\Xi\rrbracket^{W'}_{\mathcal{J}_{r+s}} = join\langle\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r}, \llbracket\Delta\rrbracket^{W'}_{\mathcal{J}_s}\rangle$ and $\llbracket\Xi\rrbracket^{W'}_{\mathcal{J}_t} = share(\llbracket\Xi'\rrbracket^{W'}_{\mathcal{J}_{r+s}})$;
   (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 $\llbracket\Xi\rrbracket^W_{\mathcal{J}_r} = join\langle\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r}, \llbracket!\Delta\rrbracket^W_{\mathcal{J}_r}\rangle$.

We require two further conditions on the model:

1. (*Syntactic monotonicity*) If $\llbracket X\rrbracket^W_{\mathcal{J}_r}$ is defined, then $\llbracket X'\rrbracket^W_{\mathcal{J}_{r'}}$ 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 $\mathcal{C} = \amalg_{W \in \mathcal{W}} C^{op}_{\mathcal{J}_r(W)}$. So that $\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r} \in \mathcal{C}_W$ and $\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r} \in \mathcal{C}_{W'}$. If there is an arrow $W \leq W' \in \mathcal{W}$, then
   (a) there exists a functor $\xi{:}\mathcal{C}_W \to \mathcal{C}_{W'}$ such that $\xi(\llbracket X\rrbracket^W_{\mathcal{J}_r}) = \llbracket X\rrbracket^{W'}_{\mathcal{J}_r}$, where X (recall) ranges over contexts, types and terms;
   (b) $\mathcal{J}_r(W')(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r}) = \mathcal{J}_r(W')(\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r})$ and $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r}) = \mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r})$, for each context Γ; otherwise $\mathcal{J}_r(W')(\llbracket\Gamma\rrbracket^{W'}_{\mathcal{J}_r})$ is undefined.

A few remarks concerning Definition 3.4 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 the functor $share$ on $\mathcal{C}$.

The second accessibility condition on the model 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 certain sense, as a 'rigid designator', that is, one whose interpretation is the same over different worlds, for a semantic object. For example, suppose $\mathbf{N}$ 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 the partiality plays two other rôles too. Firstly, there is dependent typing partiality to 'bootstrap' the definition. 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 [41], Pym [35], and Mitchell and Moggi [22] for some comments regarding these matters.

The following lemma follows easily from the definition:

LEMMA 3.5
$join$ and $share$ are functors.

PROOF. We need to show that both $join$ and $share$ preserve identities and composition. We omit the details. ∎

We now consider various model-theoretic properties of the satisfaction relation.

LEMMA 3.6 (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 3.7 ($\models_\Sigma$-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 $1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \overset{[\![M_\Gamma]\!]^W_{\mathcal{J}_r}}{\longrightarrow} [\![A_\Gamma]\!]^W_{\mathcal{J}_r}$ 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

$$1_{\mathcal{J}_{r+s}(W)(\llbracket\Gamma,x:A\rrbracket^W_{\mathcal{J}_{r+s}})} \overset{\llbracket M_{\Gamma,x:A}\rrbracket^W_{\mathcal{J}_{r+s}}}{\longrightarrow} \llbracket B_{\Gamma,x:A}\rrbracket^W_{\mathcal{J}_{r+s}}$$

is an arrow in $\mathcal{J}_{r+s}(W)(\llbracket\Gamma, x{:}A\rrbracket^W_{\mathcal{J}_{r+s}})$. We then use the natural isomorphism Λ to get the arrow

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \overset{\llbracket\lambda\, x:A\,.M_\Gamma\rrbracket^W_{\mathcal{J}_r}}{\longrightarrow} \llbracket\Lambda\, x{:}A\,.B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$. For the $\Leftarrow$ direction, suppose there exists an arrow

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \overset{\llbracket\lambda\, x:A\,.M_\Gamma\rrbracket^W_{\mathcal{J}_r}}{\longrightarrow} \llbracket\Lambda\, x{:}A\,.B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$. It follows immediately that the existence of an arrow

$$1_{\mathcal{J}_s(W)(\llbracket\Delta\rrbracket^W_{\mathcal{J}_s})} \overset{\llbracket N_\Delta\rrbracket^W_{\mathcal{J}_s}}{\longrightarrow} \llbracket A_\Delta\rrbracket^W_{\mathcal{J}_s}$$

implies the existence of an arrow

$$1_{\mathcal{J}_t(W)(\llbracket\Xi\rrbracket^W_{\mathcal{J}_t})} \overset{\llbracket M\,N_\Xi\rrbracket^W_{\mathcal{J}_t}}{\longrightarrow} \llbracket B[N/x]_\Xi\rrbracket^W_{\mathcal{J}_t},$$

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

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma,x!A\rrbracket^W_{\mathcal{J}_r})} \overset{\llbracket M_{\Gamma,x!A}\rrbracket^W_{\mathcal{J}_r}}{\longrightarrow} \llbracket B_{\Gamma,x!A}\rrbracket^W_{\mathcal{J}_r}$$

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

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \overset{\llbracket\lambda\, x!A\,.M_\Gamma\rrbracket^W_{\mathcal{J}_r}}{\longrightarrow} \llbracket\Lambda\, x!A\,.B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$. For the $\Leftarrow$ direction, suppose there exists an arrow

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \overset{\llbracket\lambda\, x!A\,.M_\Gamma\rrbracket^W_{\mathcal{J}_r}}{\longrightarrow} \llbracket\Lambda\, x!A\,.B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

in $\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})$. It follows immediately that the existence of an arrow

$$1_{\mathcal{J}_r(W)(\llbracket!\Delta\rrbracket^W_{\mathcal{J}_r})} \overset{\llbracket N_\Delta\rrbracket^W_{\mathcal{J}_r}}{\longrightarrow} \llbracket A_\Delta\rrbracket^W_{\mathcal{J}_r}$$

implies the existence of an arrow

$$1_{\mathcal{J}_r(W)(\llbracket\Xi\rrbracket^W_{\mathcal{J}_r})} \overset{\llbracket M\,N_\Xi\rrbracket^W_{\mathcal{J}_r}}{\longrightarrow} \llbracket B[N/x]_\Xi\rrbracket^W_{\mathcal{J}_r},$$

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

$$1_{\mathcal{J}_r(W)(\llbracket\Gamma\rrbracket^W_{\mathcal{J}_r})} \xrightarrow{\llbracket\langle\pi_0(M),\pi_1(M)\rangle_\Gamma\rrbracket^W_{\mathcal{J}_r}} \llbracket A\&B_\Gamma\rrbracket^W_{\mathcal{J}_r}$$

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 3.8 (Substitutivity of $\models_\Sigma$)
Let Σ be a signature and

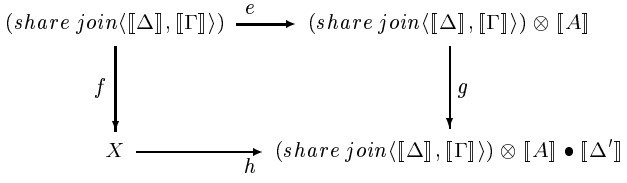
$$\langle\{\mathcal{J}_r \mid r \in R\}, \llbracket-\rrbracket, join, share, \models_\Sigma\rangle$$

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
$\llbracket\Xi'\rrbracket^W_{\mathcal{J}_{s+r'}} = join\langle\llbracket\Gamma\rrbracket^W_{\mathcal{J}_s}, \llbracket\Delta, \Delta'[N/x]\rrbracket^W_{\mathcal{J}_{r'}}\rangle$ 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
$\llbracket\Xi\rrbracket^W_{\mathcal{J}_{r'}} = join\langle\llbracket!\Gamma\rrbracket^W_{\mathcal{J}_{r'}}, \llbracket\Delta, \Delta'[N/x]\rrbracket^W_{\mathcal{J}_{r'}}\rangle$.

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, for simplicity, the parameters on the interpretation, although 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 $\mathcal{C}_W$:



where

$$\begin{array}{rcl} X & = & (share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle) \bullet (\langle 1_{share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle}, [\![N]\!]\rangle)^* [\![\Delta']\!] \\ e & = & \langle 1_{share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle}, [\![N]\!]\rangle \\ f & = & \langle 1_{share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle}, (\langle 1_{share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle}, [\![N]\!]\rangle)^* d'\rangle \\ g & = & \langle \langle 1_{share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle}, [\![N]\!]\rangle, d'\rangle \\ h & = & \langle \langle 1_{share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle}, [\![N]\!]\rangle, 1_{[\![\Delta']\!]}\rangle \end{array}$$

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 $(share\ join\langle [\![\Delta]\!], [\![\Gamma]\!]\rangle) \bullet (\langle 1_{share([\![\Delta]\!] \otimes [\![\Gamma]\!])}, [\![N]\!]\rangle)^* [\![\Delta']\!]$.

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 Soundness and completeness

In this section, we prove the (usual) soundness and completeness theorems for the $\lambda\Lambda$-calculus with respect to Kripke resource models. The proof of soundness uses the definition of interpretation and satisfaction in the model. The proof of completeness, via a term model construction, is more interesting, indicating a view of contexts as resources and worlds.

LEMMA 4.1 (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 $\mathbf{N}$ proves $\vdash_\Sigma \Gamma$ context, then, for those W where $[\![\Gamma]\!]^W_{\mathcal{J}_r}$ is defined, $[\![\Gamma]\!]^W_{\mathcal{J}_r} \in obj(\mathcal{C})$.
2. If $\mathbf{N}$ proves $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$, 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 3.4. The proofs are done by induction on the structure of proofs of system $\mathbf{N}$ and, because of interdependencies, must be done simultaneously with the proof of Theorem 4.2. ∎

THEOREM 4.2 (Soundnes)
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 $\mathbf{N}$ 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 4.1. We do some representative cases.

(Mc) Suppose $\mathbf{N}$ proves $!\Gamma \vdash_\Sigma c : \Lambda\, x_1{\in}A_1 \ldots . \Lambda\, x_m{\in}A_m\, .A$. By Definition 3.4, $\{\mathcal{J}_r \mid r \in R\}$ has enough points to interpret $c : \Lambda\, x_1{\in}A_1 \ldots . \Lambda\, x_m{\in}A_m\, .A$ and $[\![c_{!\Gamma}]\!]^W_{\mathcal{J}_r} = \Lambda^m(op_c)$ (i.e. m applications of the natural isomorphism on op_c) where

$$1_{\mathcal{J}_r(W)([\![!\Gamma, x_1\in A_1, \ldots, x_m\in A_m]\!]^W_{\mathcal{J}_r})} \xrightarrow{op_c} [\![A_{!\Gamma, x_1\in A_1, \ldots, x_m\in A_m}]\!]^W_{\mathcal{J}_r}\,.$$

It can be observed that $[\![c_{!\Gamma}]\!]^W_{\mathcal{J}_r}$ type-checks. By induction, we have that $[\![!\Gamma]\!]^W_{\mathcal{J}_r}$ is defined. So $\mathcal{J}_r, W \models_\Sigma (c : \Lambda\, x_1{\in}A_1 \ldots . \Lambda\, x_m{\in}A_m\, .A)\ [!\Gamma]$ follows.

$(MVar)$ Suppose $\mathbf{N}$ proves $\Gamma, x{:}A \vdash_\Sigma x{:}A$ because $\Gamma \vdash_\Sigma A{:}\mathsf{Type}$. By induction, we have that $[\![\Gamma, x{:}A]\!]^W_{\mathcal{J}_r}$ is defined. According to Definition 3.4, $[\![x_{\Gamma, x:A}]\!]^W_{\mathcal{J}_r} = q_{[\![\Gamma, x:A]\!]^W_{\mathcal{J}_r}}$ and the latter has the correct type. So we have shown

$$1_{\mathcal{J}_r(W)([\![\Gamma, x:A]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![x_{\Gamma, x:A}]\!]^W_{\mathcal{J}_r}} [\![A_\Gamma]\!]^W_{\mathcal{J}_r}$$

and $\mathcal{J}_r, W \models_\Sigma (x{:}A)\ [\Gamma, x{:}A]$ follows.

$(M\lambda\Lambda\mathcal{I})$ 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 $[\![B_{\Gamma, x:A}]\!]^W_{\mathcal{J}_r}$ is defined and that $\mathcal{J}_{r+s}W \vDash_\Sigma (M : B)\ [\Gamma, x : A]$, i.e. that

$$1_{\mathcal{J}_{r+s}(W)([\![\Gamma, x:A]\!]^W_{\mathcal{J}_{r+s}})} \xrightarrow{[\![M_{\Gamma, x:A}]\!]^W_{\mathcal{J}_{r+s}}} [\![B_{\Gamma, x:A}]\!]^W_{\mathcal{J}_{r+s}}\,.$$

We now use the natural isomorphism $\Lambda_{[\![\Gamma]\!]^W_{\mathcal{J}_r}, [\![A_\Gamma]\!]^W_{\mathcal{J}_s}}$ to get

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![\lambda\, x:A\, .M_\Gamma]\!]^W_{\mathcal{J}_r}} [\![\Lambda\, x{:}A\, .B_\Gamma]\!]^W_{\mathcal{J}_r}\,.$$

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 the 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 $\mathcal{J}_s, W \models_\Sigma (N{:}A)\ [\Delta]$, that is,

$$1_{\mathcal{J}_s(W)([\![\Delta]\!]^W_{\mathcal{J}_{r'}})} \xrightarrow{[\![N_\Delta]\!]^W_{\mathcal{J}_s}} [\![A_\Delta]\!]^W_{\mathcal{J}_s}\,.$$

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 $[\![\Xi']\!]^{W'}_{\mathcal{J}_{r+s}} = join\langle [\![\Gamma]\!]^{W'}_{\mathcal{J}_r}, [\![\Delta]\!]^{W'}_{\mathcal{J}_s}\rangle$ and $[\![\Xi]\!]^{W'}_{\mathcal{J}_t} = share([\![\Xi']\!]^{W'}_{\mathcal{J}_{s+r}})$, that is

$$1_{\mathcal{J}_t(W')([\![\Xi]\!]^{W'}_{\mathcal{J}_t})} \xrightarrow{[\![MN_\Xi]\!]^{W'}_{\mathcal{J}_t}} [\![B[N/x]_\Xi]\!]^{W'}_{\mathcal{J}_t}\,.$$

We check that this is the interpretation given by the model. According to Definition 3.4, $[\![M\,N_\Xi]\!]^W_{\mathcal{J}_t}$ is defined and is equal to, using monotonicity,

$$(\langle 1_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r}}, [\![N_\Delta]\!]^{W'}_{\mathcal{J}_s}\rangle^* (\Lambda_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r},[\![A_\Gamma]\!]^{W'}_{\mathcal{J}_s}}([\![M_\Gamma]\!]^{W'}_{\mathcal{J}_r}))),$$

where Ξ is defined as above. We need to check the types. Firstly, 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 $\Lambda_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r},[\![A_\Gamma]\!]^{W'}_{\mathcal{J}_s}}$ gives us

$$\Lambda_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_{r+s}},[\![A_\Gamma]\!]^{W'}_{\mathcal{J}_{r+s}}}([\![M_{\Gamma,x:A}]\!]^{W'}_{\mathcal{J}_{r+s}}) \;:\; 1_{\mathcal{J}_{r+s}(W')([\![\Gamma,x:A]\!]^{W'}_{\mathcal{J}_{r+s}})} \longrightarrow [\![B_{\Gamma,x:A}]\!]^{W'}_{\mathcal{J}_{r+s}}.$$

The functor $\langle 1_{[\![\Gamma]\!]^{W'}_{\mathcal{J}_r}}, [\![N_\Delta]\!]^{W'}_{\mathcal{J}_s}\rangle^*$ performs the required substitution. Finally the action of $join$ and $share$ gives us $[\![\Xi]\!]^{W'}_{\mathcal{J}_t}$.

$(M\&\mathcal{I})$ Suppose $\mathbf{N}$ proves $\Gamma \vdash_\Sigma \langle M, N\rangle{:}A\&B$ because $\mathbf{N}$ proves $\Gamma \vdash_\Sigma M{:}A$ and $\mathbf{N}$ proves $\Gamma \vdash_\Sigma N{:}B$. By the 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

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![N_\Gamma]\!]^W_{\mathcal{J}_r}} [\![B_\Gamma]\!]^W_{\mathcal{J}_r}.$$

Now, each category $\mathcal{J}_r(W)(D)$ in the model has products. We use this property in $\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})$ to construct

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![\langle M,N\rangle_\Gamma]\!]^W_{\mathcal{J}_r}} [\![(A\&B)_\Gamma]\!]^W_{\mathcal{J}_r}$$

and $\mathcal{J}_r, W \models_\Sigma (\langle M, N\rangle{:}(A\&B))\ [\Gamma]$ follows.

$(M\&\mathcal{E}_i)$ Suppose $\mathbf{N}$ proves $\Gamma \vdash_\Sigma \pi_i(M){:}A_i$, for $i \in \{0,1\}$ because $\mathbf{N}$ proves $\Gamma \vdash_\Sigma M{:}A_0\&A_1$. By the induction hypothesis, we have that $\mathcal{J}_r, W \models_\Sigma (M{:}A_0\&A_1)\ [\Gamma]$, that is,

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![M_\Gamma]\!]^W_{\mathcal{J}_r}} [\![(A_0\&A_1)_\Gamma]\!]^W_{\mathcal{J}_r}.$$

Then the definition of satisfaction allows us to construct, for $i \in \{0,1\}$,

$$1_{\mathcal{J}_r(W)([\![\Gamma]\!]^W_{\mathcal{J}_r})} \xrightarrow{[\![\pi_i(M)_\Gamma]\!]^W_{\mathcal{J}_r}} [\![A_{i\Gamma}]\!]^W_{\mathcal{J}_r}$$

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 3.8, allows us to show that if $M =_{\beta\eta} N$, then $[\![M]\!]^W_{\mathcal{J}_r} \simeq [\![N]\!]^W_{\mathcal{J}_r}$. ∎

To see that Dereliction is sound in these models, consider that both $[\![\Gamma, x{:}A]\!]$ and $[\![\Gamma, x!A]\!]$ have corresponding second projection maps (qs) in the fibre over $[\![\Gamma]\!]$. So if a term built out of linear qs is soundly interpreted, then so is the one built out of intuitionistic qs. Note that the converse fails because of the extra properties of the latter.

We now turn to consider completeness. We begin with the appropriate definition of validity for $\models_\Sigma$.

DEFINITION 4.3 ($\models_\Sigma$-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 structure will be defined using the category of contexts and realizations, $\mathcal{C}(\Sigma)$, that we defined previously. We need to following definitions prior to giving a model existence lemma.

PROPOSITION 4.4
$\mathcal{C}(\Sigma)$ is a doubly monoidal category.

PROOF. The two context extension operators are, roughly speaking, taken to be an extension with A and an extension with $!A$. Formally, of course, we have to define arbitrary combinations of objects (contexts, in this case) rather than extensions. So objects of $\mathcal{C}(\Sigma)$ are Δ_Γ, i.e. $\vdash_\Sigma \Gamma, \Delta$ context (we are allowed to extend only with types which are well-formed over contexts). The monoidal operators are defined by induction, using the $join$ operation:

$$\begin{array}{rcl} \Delta \otimes \Gamma & = & join(\Delta, \Gamma) \\ \Delta \times 1 (= 1 \times \Delta) & = & \Delta \\ \Delta \times \Gamma & = & join(!\Delta, !\Gamma). \end{array}$$

The following rules for the units need to be taken in 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 $!A$. ∎

DEFINITION 4.5
The category $\mathcal{P}(\Sigma)$, a full sub-category of $\mathcal{C}(\Sigma)$, 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 $\Gamma \stackrel{\langle 1, M\rangle}{\rightarrow} \Gamma \times A$ in $\mathcal{C}(\Sigma)$, then $\Gamma \times A$ is an object of $\mathcal{P}(\Sigma)$.
- Morphisms: The arrows just considered.

LEMMA 4.6
The tuple consisting of the set of objects in $\mathcal{C}(\Sigma)$, the context joining operation $[-;-;-]$ and the unit context $\langle\rangle = I$ 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 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 the 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 the 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 remark that in logics which include conjunctions and disjunctions, such as intuitionistic logic or **BI**, 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 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 4.7 (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 $\lambda\Lambda$-calculus.

The Kripke Σ-$\lambda\Lambda$ 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 $\mathcal{C}(\Sigma)_\Delta$, where each $\Delta \in ob(\mathcal{P}(\Sigma))$. The indexing monoid is given by the context joining relation $[-;-;-]$, as defined by Lemma 4.6. The functor $\mathcal{T}(\Sigma)_\Delta$, indexed by an element $\Delta \in obj(\mathcal{C}(\Sigma))$, is defined as follows:

$$\mathcal{T}(\Sigma)_\Delta(\Theta)(\Gamma) \quad = \quad \left\{ \begin{array}{ll} Objects & \text{Types } A \text{ such that } \mathbf{N} \text{ proves } \Psi \vdash_\Sigma A{:}\mathsf{Type} \\ Arrows & \mathcal{E}(\Sigma)(\Psi) \text{ arrows} \\ & \text{where } [\Phi';\Theta;\Delta],\ \Phi = \Phi'\backslash\kappa(\Theta,\Delta), \\ & \quad \text{and } [\Psi';\Gamma;\Phi],\ \Psi = \Psi'\backslash\kappa(\Gamma,\Phi). \end{array} \right.$$

The algebraic presentation of the type theory and Theorem 2.14 allows us to see that $\mathcal{T}(\Sigma)_\Delta(\Theta)(\Gamma)$ is a category. We also need to check that $\mathcal{T}(\Sigma)_\Delta$ is a functor.

We next check that $\mathcal{T}(\Sigma)_\Delta$ is a Kripke structure. Each of the following points refers to those of Definition 3.2:

1. The terminal object in each $\mathcal{T}(\Sigma)_\Delta(\Theta)(\Gamma)$ is taken to be the unit additive context 1. We choose this because the proof theory has the judgement $\mathbf{N}$ 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 map $q_{\langle\Gamma,A\rangle}$ is given by the term x where $\Gamma, x{\in}A \vdash_\Sigma x{:}A$. 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 because weakening is admissible in the syntax. We need to check that the appropriate square is a pullback. This can be done using the properties of substitution and we omit the details.

   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.

   We need to check the strictness conditions. This too can be done using the properties of substitution and we omit the details.
3. The natural isomophism is given by the abstraction and application rules of the type theory:

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

   where $x{\in}A$, recall, ranges over both linear $x{:}A$ and intuitionistic $x!A$ declarations. We need to check that these meet the Beck–Chevalley condition. This can be done using the properties of substitution and we omit the details.
4. The products in each $\mathcal{T}(\Sigma)_\Theta(\Delta)(\Gamma)$ are given by the $(M\&\mathcal{I})$ 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; and
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 2.14, 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 4.8 (Completeness)
$\Gamma \vdash_\Sigma M{:}A$ if and only if $\Gamma \models_\Sigma M{:}A$.

PROOF. Theorem 4.2 (Soundness) shows the forward direction. For the other, we assume $\Gamma \not\vdash_\Sigma M{:}A$ and apply Lemma 4.7. ∎

5 A class of set-theoretic models

We describe a class of concrete Kripke resource models, in which the $\lambda\Lambda$-structure

$$\{\mathcal{J}_r{:}[\mathcal{W}, [\mathcal{C}^{op}, \mathbf{Cat}]] \mid r \in R\}$$

is given by $\mathbf{BIFam}{:}[\mathcal{C}, [Ctx^{op}, \mathbf{Set}]]$, where $\mathcal{C}$ 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 [8] 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** within **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 $\mathbf{Fam}$ as a contextual category [5]. The following definition is from Streicher [41]:

DEFINITION 5.1
The contextual category $\mathbf{Fam}$, together with its length and denotation $\mathrm{DEN}{:}\mathbf{Fam} \to \mathbf{Set}$, 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 $\mathbf{Fam}$, 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 a kind of cartesian product and inherits the $\times$'s structural properties. It may be that some suitable category of domains has enough structure to define such a product.) In order to be able to define both $\times$ and $\otimes$, we denote $\mathbf{Fam}$ not in **Set** but in a presheaf $\mathbf{Set}^{\mathcal{C}^{op}}$, where $\mathcal{C}$ is a monoidal category. We emphasize that, in general, $\mathcal{C}$ can be any monoidal category and, therefore, we are actually going to describe a class of set-theoretic models. For simplicity, we take $\mathcal{C}^{op}$ 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 [8] gives us the $\otimes$ context extension.

We note that the restriction of Day's tensor product 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 a general construction for a fibred Day product.

Within the contextual setting, we then have the following definition:

DEFINITION 5.2
The contextual category $\mathbf{BIFam}$, together with its length and denotation $\mathrm{DEN}{:}\mathbf{BIFam} \to \mathbf{Set}^{\mathcal{M}}$, 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 $A{:}\mathrm{DEN}(D)(X) \to \mathbf{Set}^{\mathcal{M}}$ 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 $\mathbf{BIFam}$, then the morphisms between them are the functions between $\mathrm{DEN}(D)(X)$ and $\mathrm{DEN}(E)(Y)$. $\mathbf{BIFam}$ is $\mathbf{Fam}$ parametrized by $\mathcal{M}$; objects that were interpreted in **Set** are now interpreted in $\mathbf{Set}^{\mathcal{M}}$.

Now consider $\mathbf{BIFam}$ in an indexed setting. By our earlier argument relating indexed and contextual presentations of families of sets, $\mathbf{BIFam}$ can be seen as a functor category $\mathbf{BIFam}{:}[Ctx^{op}, \mathbf{Set}^{\mathcal{M}}]$. This is not quite the presheaf setting we require. However, we calculate

$$\begin{array}{rcl}[Ctx^{op}, [\mathbf{Set}^{\mathcal{M}}]] & \cong & [Ctx^{op} \times \mathcal{M}, \mathbf{Set}] \\ & \cong & [\mathcal{M} \times Ctx^{op}, \mathbf{Set}] \\ & \cong & [\mathcal{M}, [Ctx^{op}, \mathbf{Set}]],\end{array}$$

and so restore the indexed setting (and also reiterate the idea that $\mathcal{M}$ parametrizes $\mathbf{Fam}$).

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 [24, 36, 42]). 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 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 5.3
$\mathbf{BIFam}{:}[\mathcal{M}, [Ctx^{op}, \mathbf{Set}]]$ 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 3.2:

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, and 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 must check that $curry$ and $uncurry$ are isomorphisms and also that the Beck–Chevalley condition holds. However, 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, although 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 $M \times N$ of two objects in each fibre is taken to be the cartesian product of the corresponding singleton sets $\{M\} \times \{N\}$. ∎

LEMMA 5.4
BIFam can be extended to a Kripke resource Σ-$\lambda\Lambda$-model.

PROOF. The category **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. $[\![\Gamma, x{:}A]\!]^{X\otimes Y}_{\mathbf{BIFam}} \simeq [\![\Gamma]\!]^X_{\mathbf{BIFam}} \otimes [\![A_\Gamma]\!]^Y_{\mathbf{BIFam}}$;
2. $[\![\Gamma, x!A]\!]^X_{\mathbf{BIFam}} \simeq [\![\Gamma]\!]^X_{\mathbf{BIFam}} \times [\![A_\Gamma]\!]^X_{\mathbf{BIFam}}$.

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 definitions of these are similar to those for the joining relation $[-;-;-]$ and the sharing function κ of the term model. We omit the details.

Satisfaction is a relation over M and $[Ctx^{op}, \mathbf{Set}]$ with the clauses reflecting the properties — in particular, those of application — of the example model:

1. $X \models_\Sigma f{:}\Lambda\, x{:}A\,.B\ [D]$ if and only if $Y \vDash a{:}A\ [E]$ implies
$$X \otimes Y \models_\Sigma f(a){:}B[a/x]\ [D \otimes E];$$
2. $X \models_\Sigma f{:}\Pi\, x{:}A\,.B\ [D]$ if and only if $X \models_\Sigma a{:}A\ [E]$ implies
$$X \models_\Sigma f(a){:}B[a/x]\ [D \times E].$$

The conditions on the model are met due to the bifunctoriality of $\sqsubseteq$ and the definition of the interpretation function. ∎

6 Discussion

We briefly discuss what we see as the main deficiencies of the analysis presented herein. The first important deficiency of this work lies in its level of abstraction. Although our categorical framework provides a clear definition of the structure which must be carried by a model of the $\lambda\Lambda$-calculus, it remains rather closely tied to the syntactic organization of the calculus. [11] We should prefer to have a more abstract definition of a class of models, of which our Kripke resource models would be a (leading) example. We should then seek to show that Kripke resource models would be complete with respect to such a class, perhaps via a covering property. Whilst we should welcome a more abstract semantics for the $\lambda\Lambda$-calculus, we prefer instead to seek a more abstract semantics of a purely bunched calculus, mentioned in Section 2.7 and discussed below, and so to understand $\lambda\Lambda$ in a more general context.

The second important deficiency concerns the relationship between the $\lambda\Lambda$-calculus and bunched logics [37]. The internal logic of $\lambda\Lambda$ admits the bunched version of Dereliction, q.v. in simple propositional form,
$$\frac{\Gamma(\phi, \psi) \vdash \chi}{\Gamma(\phi; \psi) \vdash \chi}.$$
Moreover, as we discussed in Sections 2.7 and 4, our semantics for $\lambda\Lambda$ depends critically upon the admissibility of Dereliction: The formulation of the axiom sequent as
$$\Gamma, x \in A \vdash_\Sigma x : A,$$

[11] This objection is, perhaps, common to descriptions of models based on fibrations.

and its interpretation using second projections from the base category, where $\Gamma, x \in A$ is interpreted, to the fibres, where typing assertions are interpreted, guarantees the admissibility of Dereliction. The syntax and semantics of a purely bunched system, in which Dereliction is not necessarily admissible, is the subject of current research. A (somewhat speculative) discussion of this work is provided in [37].

The third important deficiency concerns the absence of a semantic analysis of RLF, the logical framework associated with $\lambda\Lambda$. Such an analysis should provide an account of the representation of (Kripke) models of object-logics with (Kripke) models of the type-theory (i.e. the meta-logic). Such studies, for a variety of frameworks, including LF, RLF and (the putative) BLF are the subject of current research [30, 31, 32].

Finally, consider the definition of the set-theoretic models in Section 5. It was simpler, here, to work with an unspecified, arbitrary signature. However, if we consider a signature which encoded the judgements of an imperative programming language, so that we had such Σ-operations in each fibre, then we conjecture that $\mathbf{BIFam}$ provides a basis for a good model for an imperative programming language. In particular, we conjecture that $\mathbf{BIFam}$ 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 [14, 23, 28].

Acknowledgements

We are grateful to Peter O'Hearn and to the anonymous referees for their careful and thoughtful comments on this work. We are also grateful both to the UK EPSRC, for their partial support of this work, and to Queen Mary, University of London, at which Ishtiaq and Pym both worked whilst most of this work was carried out. Discussions between Pym, Gordon Plotkin and Dale Miller, in Edinburgh in the early 1990s, provided the background to the formulation by Ishtiaq and Pym [17] of the $\lambda\Lambda$-calculus.

References

[1] A. Avron, F. Honsell, I.A. Mason, and R. Pollack. Using typed lambda calculus to implement formal systems on a machine. *Journal of Automated Reasoning*, **9**, 309–354, 1992.

[2] A. Barber. *Linear Type Theories, Semantics and Action Calculi*. PhD thesis, University of Edinburgh, 1997.

[3] H.P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

[4] P.N. Benton. A mixed linear and non-linear logic: Proofs, terms and models (preliminary report). Technical Report 352, Computer Laboratory, University of Cambridge, 1994.

[5] J. Cartmell. Generalised algebraic theories and contextual categories. *Annals of Pure and Applied Logic*, **32**, 209–243, 1986.

[6] I. Cervesato and F. Pfenning. A linear logical framework. In *Eleventh LICS, New Brunswick, NJ*, E. Clarke, ed., pp. 264–275. IEEE Computer Society Press, 1996.

[7] T. Coquand. An algorithm for testing conversion in type theory. In *Logical Frameworks*, G. Huet and Plotkin, eds, pp. 255–279. Cambridge University Press, 1991.

[8] B.J. Day. On closed categories of functors. In *Reports of the Midwest Category Seminar*, S. Mac Lane, ed., volume 137 of *Lecture Notes in Mathematics*, pp. 1–38. Springer-Verlag, 1970.

[9] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, **50**, 1–102, 1987.

[10] R. Harper, F. Honsell and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, **40**, 143–184, January 1993.

[11] R. Harper, D. Sannella, and A. Tarlecki. Structured theory representations and logic representations. *Annals of Pure and Applied Logic*, **67**, 113–160, 1994.

[12] J.S. Hodas and D. Miller. Logic programming in a fragment of intuitionistic linear logic. *Information and Computation*, **110**, 327–365, 1994.

[13] S. Ishtiaq. *A Relevant Analysis of Natural Deduction*. PhD thesis, Queen Mary and Westfield College, University of London, 1999.

[14] S. Ishtiaq and P.W. O'Hearn. BI as an assertion language for mutable data structures. In *POPL'01*, 2001.

[15] S. Ishtiaq and D.J. Pym. Kripke resource models of a dependently-typed, bunched λ-calculus. In *Computer Science Logic*, M. Rodríguez-Artalejo J. Flum, eds, volume 1683 of *Lecture Notes in Computer Science*, pp. 235–249. Springer-Verlag, 1999.

[16] S. Ishtiaq and D.J. Pym. Corrections and remarks. Research Report RR-00-04, Queen Mary and Westfield College, University of London, 2000.

[17] S.S. Ishtiaq and D.J. Pym. A relevant analysis of natural deduction. *Journal of Logic and Computation*, **8**, 809–838, 1998.

[18] I. Kant. *Immanuel Kants Logik* (Edited by GB Jäsche). Friedrich Nicolovius, Königsberg, 1800. In translation: RS Hartman and W Schwarz, Dover Publications, Inc., 1988.

[19] S. Kripke. Semantic analysis of intuitionistic logic I. In *Formal Systems and Recursive Functions*, J. N. Crossley and M. A. E. Dummett, eds, pp. 92–130. North-Holland, 1965.

[20] S. Mac Lane. *Categories for the Working Mathematician*, 2nd edition. Springer-Verlag, 1998.

[21] P. Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, **1**, 11–60, 1996. (Also: Technical Report 2, Scuola di Specializiazzione in Logica Matematica, Dipartimento di Matematica, Università di Siena, 1982.)

[22] J.C. Mitchell and E. Moggi. Kripke-style models for typed lambda calculus. *Annals of Pure and Applied Logic*, **51**, 99–124, 1991.

[23] P.W. O'Hearn. Resource interpretations, bunched implications and the $\alpha\lambda$-calculus. In *Proceedings of Typed Lambda Calculus and Applications*, J-Y Girard, ed. L'Aquila, Italy, 1999.

[24] P.W. O'Hearn and D.J. Pym. The logic of bunched implications. *Bulletin of Symbolic Logic*, **5**, 215–244, 1999.

[25] R. Paré and D. Schumacher. Abstract families and the adjoint functor theorems. In *Indexed Categories and Their Applications*, T. Johnstone *et al.*, eds, volume 661 of *Lecture Notes in Mathematics*. Springer-Verlag, 1978.

[26] A. Pitts. Categorical logic. In *Handbook of Logic in Computer Science*, volume 6, S. Abramsky, D. Gabbay and T.S.E. Maibaum, eds, Oxford Science Publications, 1992.

[27] D. Prawitz. Proofs and the meaning and completeness of the logical constants. In *Essays on Mathematical and Philosophical Logic*, J. Hintikka, J. Nuniluoto, and E. Saarinen, eds, pp. 25–40. D Reidel, 1978.

[28] D.J. Pym, P.W. O'Hearn, and H. Yang. Possible worlds and resources: the semantics of BI, 2000. Manuscript.

[29] D.J. 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.

[30] D.J. Pym. Functorial Kripke models of the λΠ-calculus, I: Type theory and internal logic. Manuscript.

[31] D.J. Pym. Functorial Kripke models of the λΠ-calculus, II: The LF logical framework. Manuscript.

[32] D.J. Pym. Functorial Kripke models of the λΠ-calculus, III: Logic programming and its semantics. Manuscript.

[33] D.J. 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.)

[34] D.J. Pym. A note on representation and semantics in logical frameworks. In *Proceedings of CADE-13 Workshop on Proof-search in Type-theoretic Languages*, Rutgers University, New Brunswick, NJ, D. Galmiche, ed., 1996. (Also: Technical Report 725, Department of Computer Science, Queen Mary & Westfield College, University of London.)

[35] D.J. Pym. Functorial Kripke models of the λΠ-calculus. Lecture at Isaac Newton Institute for Mathematical Sciences, Semantics Programme, Workshop on Categories and Logic Programming, Cambridge, 1995.

[36] D.J. Pym. On bunched predicate logic. In *Procs LICS 1999*, pp. 183–192. IEEE, 1999.

[37] D.J. Pym. *The Semantics and Proof Theory of the Logic of Bunched Implications*. Applied Logic Series, Kluwer Academic Publishers, Boston/Dordrecht/London, 2002 (to appear).

[38] S Read. *Relevant Logic: A Philosophical Examination of Inference*. Basil Blackwell, 1988.

[39] 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.

[40] P. Schroeder-Heister. Generalized rules for quantifiers and the completeness of the intuitionistic operators $\&, \vee, \supset, \curlywedge, \forall, \exists$. In *Computation and Proof Theory, Logic Colloquim Aachen*, M.M. Richter *et al.*, eds, volume 1104 of *Lecture Notes in Mathematics*, pp. 399–426. Springer-Verlag, 1983.

[41] T. Streicher. *Correctness and Completeness of a Categorical Semantics of the Calculus of Constructions*. PhD thesis, Universität Passau, 1988.

[42] A. Urquhart. Semantics for relevant logics. *Journal of Symbolic Logic*, **37**, 159–169, 1972.

[43] D. van Dalen. *Logic and Structure*. Springer-Verlag, 1994.

Received 12 December 2000