Linear Realizability and Full Completeness for Typed Lambda-calculi

Samson Abramsky

Oxford University Computing Laboratory, Wolfson Building, Parks Road, OX1 3QD, England, tel. +44 (0)1865 283558, fax: +44 (0)1865 273839

Marina Lenisa *,[1]

Dipartimento di Matematica e Informatica, Università di Udine, Via delle Scienze 206, 33100 Udine, ITALY, tel. +39 0432 558417, fax: +39 0432 558499

Abstract

We present the model construction technique called *Linear Realizability*. It consists in building a category of *Partial Equivalence Relations* over a *Linear Combinatory Algebra*. We illustrate how it can be used to provide models, which are *fully complete* for various typed λ-calculi. In particular, we focus on special Linear Combinatory Algebras of *partial involutions*, and we present PER models over them which are fully complete, *inter alia*, w.r.t. the following languages and theories: the fragment of System F consisting of ML-types, the *maximal theory* on the simply typed λ-calculus with *finitely many* ground constants, and the maximal theory on an *infinitary* version of this latter calculus.

Key words: Typed lambda-calculi, ML-polymorphic types, linear logic, hyperdoctrines, PER models, Geometry of Interaction, (axiomatic) full completeness

* Corresponding author.
Email addresses: `Samson.Abramsky@comlab.ox.ac.uk` (Samson Abramsky), `lenisa@dimi.uniud.it` (Marina Lenisa).

[1] Work partially supported by UE Project IST-2000-29001 Types, and by Italian MIUR Projects COFIN 2001013518 Cometa and 20022018192_002 Protocollo.

Introduction

The aim of this paper is to illustrate the technique of *Linear Realizability* for building *fully complete* models for various typed λ-calculi. This paper combines and expands previous works by the authors, [9,11–13].

A *categorical* model of a type theory (or logic) is said to be *fully-complete* [7] if, for all *types* (*formulae*) A, B, all morphisms $f : [\![A]\!] \to [\![B]\!]$, from the interpretation of A into the interpretation of B, are denotations of a *proof-term* of the entailment $A \vdash B$, i.e. if the interpretation function from the category of syntactical objects to the category of denotations is *full*. The notion of full-completeness is a counterpart to the notion of *full abstraction* for programming languages. Besides full completeness, one can ask the question whether the theory induced by a model $\mathcal{M}$ coincides precisely with the syntactical theory or whether more equations are satisfied in $\mathcal{M}$. A model $\mathcal{M}$ is called *faithful* if it realizes exactly the syntactical theory.

A fully complete model indicates that there is a very tight connection between syntax and semantics. Equivalently, one can say that the term model has been made into a mathematically respectable structure.

Over the past decade, Game Semantics has been used successfully by various authors to define fully-complete models for various fragments of Linear Logic, and to give fully-abstract models for many programming languages, including PCF [8,23], and other functional and non-functional languages. In this paper, we propose the technique of *linear realizability* as a valid and less complex alternative to Game Semantics in providing fully complete and fully abstract models for typed λ-calculi.

The linear (linear affine) realizability technique amounts to constructing a category of *Partial Equivalence Relations (PERs)* over a *Linear Combinatory Algebra, LCA*, (Affine Combinatory Algebra, ACA). This category turns out to be *linear* (affine), and to form an *adjoint model* with its co-Kleisli category. The notion of Linear (Affine) Combinatory Algebra introduced by the first author [2] refines the standard notion of Combinatory Algebra, in the same way in which intuitionistic linear (affine) logic refines intuitionistic logic. The construction of PER models from LCA's (ACA's) described in this paper is quite simple and clear, and it yields models with *extensionality* properties, thus avoiding extra quotienting operations, which are often needed in defining game categories and models. Many examples of linear combinatory algebras arise in the context of Abramsky's categorical version of Girard's *Geometry of Interaction*, [6,2,1,5].

In order to illustrate the technique of linear realizability, we present a number of case studies of fully complete models for various typed λ-calculi. These models arise from special ACAs of *partial involutions*, and they are defined in the co-Kleisli category of the category of PERs on them. The models which we study are the following:

(1) A fully complete (and necessarily faithful) model for the fragment of System F (with the $\beta\eta$-theory) consisting of *ML-polymorphic types*. ML-types are universal closures of simple types, i.e. types of the form $\forall X_1 \ldots . X_n.T$, where T is $\forall$-free.
(2) A fully complete model for the *maximal theory* $\approx$ on the simply typed λ-calculus with two ground constans $\bot, \top$. This model turns out to be fully complete also for an *infinitary* version of this calculus.
(3) A fully complete model for the maximal theory on the simply typed λ-calculus with *finitely many* ground constants and *ground permutations*.
(4) A fully complete model for the maximal theory on the simply typed λ-calculus with *finitely many* ground constants.

Some remarks are in order. The full completeness result on ML-types extends the result of full completeness on *algebraic types* of [25] (see Section on Related Work). The maximal theory $\approx$ [36] on the typed languages above can be characterized as follows: $\approx$ equates two closed λ-terms M, N of type $T_1 \to \ldots \to T_n \to o$ if and only if, for all P_1, Q_1 closed λ-terms of type T_1 such that $P_1 \approx Q_1$, ..., P_n, Q_n closed λ-terms of type T_n such that $P_n \approx Q_n$,

$$MP_1 \ldots P_n =_\beta NQ_1 \ldots Q_n .$$

To our knowledge, our fully complete minimal models are the first models of $\approx$ different from the corresponding term models.

The proofs of full completeness make essential use of the linear affine categories which underly the models, and they follow a *common* (by now standard) *pattern* based on a *Decomposition Theorem.* This allows to recover, for all types T, T', the top-level form of the *typed Böhm tree* corresponding to any morphism $f : [\![T]\!] \to [\![T']\!]$. Once we have the Decomposition Theorem, we can apply it iteratively to any $f : [\![T]\!] \to [\![T']\!]$, getting a possibly *infinite* typed Böhm tree. Thus, to finish the full completeness proofs, we need to rule out *infinite* Böhm trees from our models. This part of the proof can be rather difficult.

An interesting parallel line of research is that of abstracting the key lemmata in the proofs of full completeness, and giving *axioms* on categorical models, which ensure full completeness. These axiomatizations provide a useful guide in concrete proofs of full completeness. The first axiomatization of this kind has been given in [3], where axioms on models of PCF are given in order to guarantee full abstraction, and axioms on models of the simply typed λ-calculus are given, in order to guarantee full and faithful completeness w.r.t. the $\beta\eta$-theory. The axioms for PCF are abstracted from the key lemmas in the proof of full abstraction of the game model of [8]. This proof makes essential use of the underlying *linear* structure of the game category. Correspondingly, the axiomatization in [3] applies to models of PCF which arise as co-Kleisli categories of some *linear category*. These kind of models, where we have a linear category and a cartesian closed category, together with a *monoidal adjunction* between the two categories, are called *adjoint models*, following [18,17].

Our proof of full completeness for ML-types is based, in fact, on an axiomatization of fully-complete models for ML-types. This axiomatization is given on the models of system F, originated by Lawvere, which are called *hyperdoctrines* (see also [33]). As in [3], our axiomatization works in the context of *adjoint models*, and, although the full completeness result applies to *intuitionistic* types, it makes essential use of the *linear* decomposition of these types. Our axiomatization consists of two crucial steps. First, we axiomatize the fact that every morphism inhabiting an ML-type generates, under *decomposition*, a possibly *infinite* typed Böhm tree. Then, we introduce an axiom which rules out infinite trees from the model. Our concrete PER model of partial involutions satisfies the axioms in our axiomatization. In particular, proving that the model does not contain infinite typed Böhm trees is quite difficult, and it requires the study of an intermediate model, which offers our second case study. This is the model of the simply typed λ-calculus generated by a suitable *Sierpinski PER*. We prove that this model is fully complete and *minimal*, i.e. it realizes the maximal theory for the case of two ground constants $\bot, \top$. Actually, it is fully complete and minimal for an *infinitary* version of this calculus, including possibly infinite typed Böhm trees (i.e. supremums of coherent chains of infinite typed trees). Immediate generalizations of the Sierpinski PER to k equivalence classes fail to give fully complete models for the case of k ground constants. However, a quite interesting and crucial fact is that an appropriate generalization of the Sierpinski PER gives rise to a fully complete minimal model for the calculus extended with *ground permutations* (our third case study). Finally, in order to get fully complete minimal models exactly for the simply typed λ-calculus with finitely many ground constants, we show how we can cut down the algebra of involutions by putting an extra constraint, which rules out permutations from the model.

The authors are thankful to F. Honsell, R. Jagadeesan, J. Laird, J. Longley, S. Martini and A. Simpson for useful discussions on some of the issues of the paper. We are grateful for the anonymous journal referee's detailed comments, which led to a number of clarifications and minor corrections.

Related Work

The problem of *full completeness* for second order (polymorphic) λ-calculus, i.e. Girard's System F ([21]), is a very important problem, which has been extensively studied.

Here are some results in the literature.

In [25], the category of Partial Equivalence Relations (PER) over the open term model of the untyped λ-calculus has been proved to be fully (and faithfully) complete for *algebraic types*. These are *ML-types* of rank less or equal than 2, like for instance the type $\forall X.(X \to X) \to X \to X$ of *Church's numerals*. The rank of an ML-type $\forall \vec{X}.T$ is the *nesting level* of negative occurrences of $\to$ in the simple type T. A fully-complete model for the whole System F has been provided in [16], but this model is built by means of a quotient on terms, and therefore it is not compositional and not sufficiently abstract. More recently, in [24], a fully and faithfully complete game model for System F has been given. But, although this is a game model, it still has a somewhat syntactical flavour, and the construction of the model is extremely complex.

Notation. Vectors are written in bold. For $f : \mathbf{N} \rightharpoonup \mathbf{N}$ a partial function from $\mathbf{N}$ to $\mathbf{N}$, and $n \in \mathbf{N}$, we denote by $f(n)\uparrow$ the fact that f is not defined (diverges) on input n. Let X, Y be sets, then (l, x) and (r, y), where $x \in X$ and $y \in Y$, denote elements of $X + Y$. Let $X_1, \ldots, X_n$ be sets, often we will omit the parentheses in writing $((X_1 + \cdots) + X_{n-1}) + X_n$. Moreover, we will use the abbreviated notation (i, x) for denoting the element of $X_1 + \ldots + X_n$ coming from an element $x \in X_i$. Finally, an isomorphism $\phi : A \to B$ in a category $\mathcal{C}$, whose inverse is ϕ', is denoted by $\phi : A \simeq B : \phi'$.

1 Simply Typed λ-calculus, Maximal Theory, ML Polymorphism, System F

We start by recalling some definitions, and by introducing some notations concerning the simply typed λ-calculus, the maximal theory over it, System F, and the class of *ML-types* of System F. Then, we present two important results on ML-terms of System F and on the simply typed λ-calculus. The first result that we present is an immediate consequence of Statman's *Typical Ambiguity Theorem*, asserting that the only consistent theory on the fragment of System F consisting of ML-types is precisely the $\beta\eta$-theory. The second result concerns the definability of "convergence tests" in the simply typed λ-calculus with $\perp$-constant. In particular, we prove that, for every normal form, there are convergence test terms, which detect the presence of the $\perp$-constant in the normal form. This result is used in the proof of full completeness of the model for ML-types of Section 3.

Finally, we discuss models of System F based on *hyperdoctrines*. In particular, we introduce the notion of *adjoint hyperdoctrine*, which consists of a co-Kleisli indexed category of a *linear* indexed category.

Definition 1.1 (Simply Typed λ-calculus) *The class* SimType *of simple types over a (possible infinite) set of type variables* TVar *is defined by:*

$$(SimType \ni)\ T \ ::= \ X \ \mid \ T \to T \ ,$$

where $X \in TVar$.

Raw Terms are defined as follows:

$$\Lambda \ni \ M \ ::= c \ \mid \ x \ \mid \ \lambda x : T.M \ \mid \ MM \ ,$$

where $c \in C$ is a set of constants.
We denote by Λ^0 the set of closed *λ-terms.*
We take λ-terms up-to α-conversion*, i.e. up-to renaming of bound variables (see [15] for more details).*

Well-typed terms. *We introduce a* proof system *for deriving* typing judgements *of the form $\Delta \vdash M : T$, where Δ is a* type assignment*, i.e. a finite list $x_1 : T_1, \ldots, x_n : T_n$, where $x_1, \ldots, x_n$ are all distinct. The rules of the proof system are the following:*

$$\frac{}{\Delta \vdash c : T_c} \qquad\qquad \frac{}{\Delta, x : T, \Delta' \vdash x : T}$$

$$\frac{\Delta, x : T, \Delta' \vdash M : S}{\Delta, \Delta' \vdash \lambda x : T.M : T \to S} \qquad\qquad \frac{\Delta \vdash M : T \to S \quad \Delta \vdash N : T}{\Delta \vdash MN : S}$$

$\beta\eta$-conversion. *$\beta\eta$-conversion between well-typed terms is the least relation generated by the following rules and the rules for congruence closure (which we omit):*

β) $\Delta \vdash (\lambda x : T.M)N = M[N/x] \ : \ S$, where $\Delta, x : T \vdash M : S$, and $\Delta \vdash N : T$.

η) $\Delta \vdash \lambda x : T.Mx = M \ : \ T \to S$, where $\Delta \vdash M : T \to S$, and $x \notin dom(\Delta)$.

Definition 1.2 (Typed λ-theory) *We define a* typed λ-theory $\approx$ *as a subset of $\Lambda^0 \times \Lambda^0$, respecting types and closed under $\beta\eta$-conversion.*

Notation.

- We call λ_C^{TVar} the simply typed λ-calculus with type variables in TVar, $|\mathrm{TVar}| \geq 1$, and constants in C of type o, for all $o \in \mathrm{TVar}$. In particular, we will simply denote by $\lambda_\perp^{TVar}$ the calculus $\lambda_{\{\perp\}}^{TVar}$, and by $\lambda_{\perp,\top}^{TVar}$ the calculus $\lambda_{\{\perp,\top\}}^{TVar}$.
- The simply typed λ-calculus $\lambda_C^{\{o\}}$ with one type variable o will be simply denoted by λ_C. Special cases which we will consider are: $\lambda_\emptyset$, which will be simply denoted by λ; $\lambda_{\{\perp\}}$, which will be denoted by $\lambda_\perp$; $\lambda_{\{\perp,\top\}}$, which will be denoted by $\lambda_{\perp,\top}$; $\lambda_{\{c_1,\ldots,c_k\}}$, for $k \geq 0$, which will be denoted by λ_k.

It is well-known that the *maximal theory* over λ_k can be characterized as follows:

Definition 1.3 (Maximal Theory on λ_k, [36]) *Let $M, N \in \Lambda$. We define the equivalence $\approx \subseteq \Lambda^0 \times \Lambda^0$ by induction on types as follows:*
$M \approx_{T_1 \to \ldots T_n \to o} N$ *iff* $\vdash M : T_1 \to \ldots T_n \to o$, $\vdash N : T_1 \to \ldots T_n \to o$, *and* $\forall P_1 \approx_{T_1} Q_1, \ldots, P_n \approx_{T_n} Q_n.\ MP_1 \ldots P_n =_\beta NQ_1 \ldots Q_n$.

Notice that, actually $\lambda_\perp$ and $\lambda_{\perp,\top}$ are special cases of λ_k, since ground constants are indistinguishable in the maximal theory. But in the semantics the constant $\perp$ is interpreted as the *undefined* constant, while the other constants remain indistinguishable.

In this paper, we will focus on categorical models for the simply typed λ-calculus λ_k. As usual, categorical models of λ_k are cartesian closed categories, in which types are interpreted by objects and terms in contexts are interpreted by morphisms.

Definition 1.4 (Fully Complete Model for λ_k) *A categorical model $(\mathcal{C}, [\![\ ]\!])$ of λ_k is* fully complete *if, for all simple types T and for all $h : 1 \to [\![T]\!]$ in $\mathcal{C}$, there exists $M \in \Lambda^0$ such that $h = [\![\vdash M : T]\!]$, where $[\![\ ]\!]$ is the interpretation function in the model.*

Definition 1.5 (System F) *The class* Type *of System F types over an infinite set of type variables* TVar *is defined by:*

$$(Type \ni)\ T \ ::=\ X \ \mid\ T \to T \ \mid\ \forall X.T\ ,$$

where $X \in TVar$.

System F raw terms are defined as follows:

$$M \ ::=\ x \ \mid\ \lambda x : T.M \ \mid\ MM \ \mid\ \Lambda X.M \ \mid MT\ ,$$

where $x \in Var$.

We take both types and terms up-to α-conversion.

Well-typed terms. *The proof system for deriving* typing judgements *is defined as follows. A typing judgement has the form* $\Gamma; \Delta \vdash M : T$*, where* Γ *is a* context*, i.e. a finite list of type variables, and* Δ *is a* type assignment*, i.e. a finite list* $x_1 : T_1, \ldots, x_n : T_n$*, such that each* T_i *is* legal *in* Γ*. The rules for deriving the judgement* $\Gamma \vdash T$*, read as "*T *is legal in* Γ*", are the following:*

$$\frac{}{\Gamma, X, \Gamma' \vdash X} \qquad \frac{\Gamma \vdash T \quad \Gamma \vdash S}{\Gamma \vdash T \to S} \qquad \frac{\Gamma, X \vdash T}{\Gamma \vdash \forall X.T}$$

The rules for deriving the typing judgement $\Gamma; \Delta \vdash M : T$ *are the following:*

$$\frac{}{\Gamma; \Delta, x : T, \Delta' \vdash x : T} \qquad \frac{\Gamma; \Delta, x : T \vdash M : S}{\Gamma; \Delta \vdash \lambda x : T.M : T \to S}$$

$$\frac{\Gamma; \Delta \vdash M : T \to S \quad \Gamma; \Delta \vdash N : T}{\Gamma; \Delta \vdash MN : S}$$

$$\frac{\Gamma, X; \Delta \vdash M : T}{\Gamma; \Delta \vdash \Lambda X.M : \forall X.T} \; X \notin FV(ran(\Delta)) \qquad \frac{\Gamma; \Delta \vdash M : \forall X.T \quad S \textit{ is legal in } \Gamma}{\Gamma; \Delta \vdash MS : T[S/X]}$$

where $ran(\Delta)$ *denotes the range (codomain) of* Δ*.*

$\beta\eta$**-conversion.** *The* $\beta\eta$*-conversion between well-typed terms is the least relation generated by the following rules and the rules for congruence closure (which we omit):*

β) $\Gamma; \Delta \vdash (\lambda x : T.M)N = M[N/x] : S$*, where* $\Gamma; \Delta, x : T \vdash M : S$*, and* $\Gamma; \Delta \vdash N : T$*.*

η) $\Gamma; \Delta \vdash \lambda x : T.Mx = M : T \to S$*, where* $\Gamma; \Delta \vdash M : T \to S$*, and* $x \notin dom(\Delta)$*.*

β_2) $\Gamma; \Delta \vdash (\Lambda X.M)T = M[T/X] : S$*, where* $\Gamma, X; \Delta \vdash M : S$*, and* $X \notin FV(ran(\Delta))$*.*

η_2) $\Gamma; \Delta \vdash \Lambda X.MX = M : \forall X.S$*, where* $\Gamma; \Delta \vdash M : \forall X.S$*, and* $X \notin FV(ran(\Delta))$*.*

Now we introduce the class of *ML-polymorphic types*, which correspond to the limited kind of polymorphism allowed in the language ML.

Definition 1.6 (ML-types) *The class* ML-Type *of ML-types is defined by:*

$$\textit{ML-Type} = \{\forall \vec{X}.T \mid T \in \textit{SimType}\ ,\ n \geq 0\}\ .$$

Definition 1.7 (ML-terms) *The class* ML-term *of ML-terms is defined by:*

$$\mathit{ML\text{-}Term} = \{M \mid \exists \Gamma, \Delta.\ \Gamma; \Delta \vdash M : T \quad \mathit{and}\ T \in \mathit{ML\text{-}Type}\}\ .$$

Notice that the class of ML-terms of System F as defined above includes terms whose subterms have possibly a *non*-ML-type. However, if we restrict ourselves to normal forms, then one can easily check that all subterms of a normal form of ML-type have ML-types. This, together with the fact that System F is strongly normalizing, allow us to restrict ourselves to normal forms, when discussing full completeness of System F at ML-types (see Definition 1.11 below and the comment after it).

1.1 Statman's Typical Ambiguity Theorem

The following is a result about simply typed λ-terms from [37], rephrased for ML-terms:

Theorem 1.1 (Statman's Typical Ambiguity) *Let T be a closed ML-type. If $\not\vdash M =_{\beta\eta} N : T$, then there exists a closed term L such that $\vdash L : T \to \mathit{Bool}$, where $\mathit{Bool} = \forall Y.Y \to Y \to Y$ and Y is fresh, s.t.*

$$\vdash LM =_{\beta\eta} \mathtt{true} : \mathit{Bool} \quad \wedge \quad \vdash LN =_{\beta\eta} \mathtt{false} : \mathit{Bool}\ ,$$

where $\mathtt{true} = \Lambda Y.\lambda x : Y.\lambda y : Y.x$ *and* $\mathtt{false} = \Lambda Y.\lambda x : Y.\lambda y : Y.y$.

Corollary 1.1 *The maximal consistent theory on the fragment of System F consisting of ML-types is the $\beta\eta$-theory.*

As will be clear from the definition of full completeness (see Definition 1.11 below), by Corollary 1.1, any non-trivial fully-complete model for ML-types of System F is necessarily *faithful*, i.e. it realizes exactly the $\beta\eta$-theory at ML-types.

1.2 λ-definability of Convergence Tests in the Simply Typed λ-calculus

We show that "convergence tests" are λ-definable in $\lambda_\perp^{TVar}$ for a suitable class of types.

Definition 1.8 (Typed Convergence Tests) *i) Let o be a distinguished type variable, and let $\alpha_T \in \mathit{SimType}_{o\to o}$ be a type such that α_T is obtained from some $T \in \mathit{SimType}$ by substituting every occurrence of every variable X in T by*

$o \to o$, *i.e.* $\alpha_T = T[o \to o/\vec{X}]$. *We define, by induction on* T, *the* convergence test *term* $\vdash S_{\alpha_T} : \alpha_T$ *as follows:*

- *if* $T = X$, *then* $S_{o\to o} = I_{o\to o}$,
- *otherwise, let* $T = T_1 \to \ldots \to T_n \to X_k$, *where* $T_i = U_{i1} \to \ldots \to U_{iq_i} \to X_i$, *then*

$$S_{\alpha_T} = \lambda x_1 : \alpha_{T_1} \ldots x_n : \alpha_{T_n}.\lambda z : o.(x_1 S_{\alpha_{U_{11}}} \ldots S_{\alpha_{U_{1q_1}}})(\ldots(x_n S_{\alpha_{U_{n1}}} \ldots S_{\alpha_{U_{nq_n}}} z)) \ .$$

ii) Let σ *be a substitution on* λ*-terms such that, for every term* M *of type* T, σ *gives the term of type* α_T *obtained from* M *by replacing each occurrence of the constant* $\perp$ *by the term* $\lambda x : o.\ \perp$. *Formally,* σ *is defined by induction on terms in context as follows:*

$$\sigma(\Delta \vdash x : T) = \overline{\Delta} \vdash x : \alpha_T, \text{ where } \overline{\Delta} \text{ is defined by } x_i : \alpha_{T_i} \in \overline{\Delta} \text{ iff } x_i : T_i \in \Delta$$

$$\sigma(\Delta \vdash \perp : X) = \overline{\Delta} \vdash \lambda x : o.\ \perp : o \to o, \quad \text{where } x \notin \Delta$$

$$\sigma(\Delta \vdash MN : S) = \overline{\Delta} \vdash M'N' : \alpha_S, \quad \text{where } \overline{\Delta} \vdash M' : \alpha_{T\to S} = \sigma(\Delta \vdash M : T \to S) \text{ and } \overline{\Delta} \vdash N' : \alpha_T = \sigma(\Delta \vdash N : T)$$

$$\sigma(\Delta, \Delta' \vdash \lambda x : T.M : T \to S) = \overline{\Delta}, \overline{\Delta'} \vdash \lambda x : \alpha_T.M' : \alpha_{T\to S}, \quad \text{where } \overline{\Delta}, x : \alpha_T, \overline{\Delta}' \vdash M' : \alpha_S = \sigma(\Delta, x : T, \Delta' \vdash M : S).$$

Notice that the substitution σ above preserves normal forms.

The "convergence test" terms defined above give us a procedure for deciding whether a normal form M of type $T = T_1 \to \ldots \to T_n \to X_k$ contains a divergent subterm, i.e. $\perp$. Namely, let us consider the corresponding normal form M' of type α_T obtained from $\sigma(\Delta \vdash M : T)$. We apply M' to the sequence of convergence tests $S_{\alpha_{T_1}}, \ldots, S_{\alpha_{T_n}}$. The effect of this is that, in the head reduction of $M' S_{\alpha_{T_1}} \ldots S_{\alpha_{T_n}}$, each subterm of M' definitely appears in head position, and it reduces to the identity, until a $\perp$ is detected.

Theorem 1.2 (Typed Separability) *Let* $\vec{y} : \vec{U} \vdash M' : T$, *where* $T = T_1 \to \ldots \to T_n \to X_k$, *let* $\vec{y} : \vec{\alpha}_U \vdash M : \alpha_T = \sigma(\vec{y} : \vec{U} \vdash M' : T)$, *and let* $\widetilde{M} = M[\vec{S}_{\alpha_U}/\vec{y}]$ *be the closed term obtained by saturating all the free variables in* M *by convergence tests of the appropriate types. Then*

$$\widetilde{M} S_{\alpha_{T_1}} \ldots S_{\alpha_{T_n}} = \begin{cases} I_{o\to o} & \text{if the normal form of } M \text{ is } \perp\text{-free} \\ \lambda x : o.\ \perp : o \to o & \text{otherwise} \ . \end{cases}$$

Proof. We proceed by induction on the normal form of M.
Base case: there are two cases: a) $\vec{y} : \vec{\alpha}_U \vdash x : \alpha_T$. We have to prove that $S_{\alpha_T} S_{\alpha_{T_1}} \ldots S_{\alpha_{T_n}} = I_{o\to o}$. But this latter fact can be immediately shown by induction on T. b) $\vec{y} : \vec{\alpha}_U \vdash \lambda z : o.\ \perp : o \to o$. Then $\widetilde{M} = \lambda z : o.\ \perp : o \to o$.

Induction Step: $\vec{y} : \vec{\alpha}_U \vdash \lambda\vec{z}.x_i M_1 \dots M_{q_i} : \alpha_T$. Then either $\widetilde{M} = \lambda\vec{z}.x_i\widetilde{M_1}\dots\widetilde{M_{q_i}}$ or $\widetilde{M} = \lambda\vec{z}.S_{\alpha_{T_i}}\widetilde{M_1}\dots\widetilde{M_{q_i}}$. In any case $\widetilde{M}S_{\alpha_{T_1}}\dots S_{\alpha_{T_n}} = S_{\alpha_{T_i}}\widetilde{M}_1\dots\widetilde{M}_{q_i} = \lambda z : o.(\widetilde{M}_1\vec{S})(\dots(\widetilde{M}_{q_i}\vec{S})z)$. The theorem follows by applying the induction hypothesis to each M_j, for all $j = 1, \dots, q_i$. □

Theorem 1.2 above can be regarded as a *typed version* of Böhm Separability Theorem, in the sense that, if we think of $\perp$ as a generic *unsolvable* term, then Theorem 1.2 allows us to tell normal forms apart from solvable terms which do not reduce to a normal form.

1.3 Models of System F

We focus on *hyperdoctrine models* of System F. First, we recall the notion of $2\lambda\times$-hyperdoctrine (see [33]). This essentially corresponds to the notion of *external model* (see [14]). Then, we give the formal definition of full (and faithful) complete hyperdoctrine model. Finally, we carry out a *linear* analysis of the notion of $2\lambda\times$-hyperdoctrine. This will allow us to express conditions which guarantee full completeness of the model w.r.t. ML-types. In particular, we introduce a categorical notion of *adjoint hyperdoctrine*. Adjoint hyperdoctrines arise as *co-Kleisli* indexed categories of *linear* indexed categories.

In what follows, we assume that all indexed categories which we consider are strict (see e.g. [14,19] for more details on indexed categories).

Definition 1.9 **($2\lambda\times$-hyperdoctrine, [30,33])** *A* $2\lambda\times$-hyperdoctrine *is a triple* $(\mathcal{C}, \boldsymbol{G}, \forall)$*, where:*

- $\mathcal{C}$ *is the* base category*, it has finite products, and it consists of a distinguished object* $\mathcal{U}$ *which generates all other objects using the product operation* $\times$*. We will denote by* $\mathcal{U}^m$*, for* $m \geq 0$*, the objects of* $\mathcal{C}$*.*
- $\boldsymbol{G} : \mathcal{C}^{op} \to$ CCCat *is a* $\mathcal{C}$*-indexed cartesian closed category such that: for all* $\mathcal{U}^m$*, the underlying collection of objects of the cartesian closed fibre category* $\boldsymbol{G}(\mathcal{U}^m)$ *is indexed by the collection of morphisms from* $\mathcal{U}^m$ *to* $\mathcal{U}$ *in* $\mathcal{C}$*, i.e. the objects of* $\boldsymbol{G}(\mathcal{U}^m)$ *are the morphisms in* $Hom_{\mathcal{C}}(\mathcal{U}^m,\mathcal{U})$*, and, for any morphism* $f : \mathcal{U}^m \to \mathcal{U}^n$ *in* $\mathcal{C}^{op}$*, the cartesian closed functor* $\boldsymbol{G}(f) : \boldsymbol{G}(\mathcal{U}^n) \to \boldsymbol{G}(\mathcal{U}^m)$*, called* reindexing functor *and denoted by* f^**, is such that, for any object* $h : \mathcal{U}^n \to \mathcal{U}$*,* $f^*(h) = f; h$*;*
- *For each object* $\mathcal{U}^m$ *of* $\mathcal{C}$*, we are given a functor* $\forall_m : \boldsymbol{G}(\mathcal{U}^m \times \mathcal{U}) \to \boldsymbol{G}(\mathcal{U}^m)$ *such that*
  - · $\forall_m$ *is right adjoint to the functor* $\pi_m^* : \boldsymbol{G}(\mathcal{U}^m) \to \boldsymbol{G}(\mathcal{U}^m \times \mathcal{U})$*, where* $\pi_m : \mathcal{U}^m \times \mathcal{U} \to \mathcal{U}^m$ *is the projection in* $\mathcal{C}$*;*
  - · $\forall_m$ *satisfies the* Beck-Chevalley condition*, i.e.:*

    for any morphism $f : \mathcal{U}^m \to \mathcal{U}^n$ *in* $\mathcal{C}$*, the following diagram of func-*

tors commutes

$$\begin{tikzcd} \boldsymbol{G}(\mathcal{U}^n \times \mathcal{U}) \arrow[r, "\forall_{\mathcal{U}^n}"] \arrow[d, "(f\times id_{\mathcal{U}})^*"'] & \boldsymbol{G}(\mathcal{U}^n) \arrow[d, "f^*"] \\ \boldsymbol{G}(\mathcal{U}^m \times \mathcal{U}) \arrow[r, "\forall_{\mathcal{U}^m}"'] & \boldsymbol{G}(\mathcal{U}^m) \end{tikzcd}$$

for any $f : \mathcal{U}^m \to \mathcal{U}^n$*, the canonical natural transformation* $f^* \circ \forall_{\mathcal{U}^n} \to \forall_{\mathcal{U}^m} \circ (f \times id_{\mathcal{U}})^*$ *is an identity.*

Any $2\lambda\times$-hyperdoctrine can be endowed with a notion of interpretation $[\![\ ]\!]$ for the language of System F. Types with free variables in $X_1, \ldots, X_m$ are interpreted by morphisms from $\mathcal{U}^m$ to $\mathcal{U}$ in the category $\mathcal{C}$, i.e. by objects of $\mathbf{G}(\mathcal{U}^m)$:

$$[\![X_1, \ldots, X_m \vdash T]\!] : \mathcal{U}^m \to \mathcal{U}\ .$$

Well-typed terms, i.e. $X_1, \ldots, X_m; x_1 : T_1, \ldots, x_n : T_n \vdash M : T$, are interpreted by morphisms in the category $\mathbf{G}(\mathcal{U}^m)$:

$$[\![X_1, \ldots, X_m; x_1 : T_1, \ldots, x_n : T_n \vdash M : T]\!] : [\![\vec{X} \vdash T_1]\!] \times \ldots \times [\![\vec{X} \vdash T_n]\!] \to [\![\vec{X} \vdash T]\!]\ .$$

More precisely:

Definition 1.10 *We can endow any* $2\lambda\times$*-hyperdoctrine* $(\mathcal{C}, \boldsymbol{G}, \forall)$ *with an* interpretation function $[\![\ ]\!]$ *for the language of System F as follows.*

$[\![\ ]\!]$ *is defined on types by induction on derivations of the judgement* $\Gamma \vdash T$*:*

- $[\![\Gamma, X, \Gamma' \vdash X]\!] = \pi_i : [\![\Gamma]\!] \times \mathcal{U} \times [\![\Gamma']\!] \to \mathcal{U}$
- $[\![\Gamma \vdash T \to S]\!] = [[\![\Gamma \vdash T]\!] \to [\![\Gamma \vdash S]\!]]$;
- $[\![\Gamma \vdash \forall X.T]\!] = \forall([\![\Gamma, X \vdash T]\!])$.

$[\![\ ]\!]$ *is defined on terms by induction on derivations of the typing judgement* $\Gamma; \Delta \vdash M : T$*:*

- $[\![\Gamma; x_1 : T_1, \ldots, x_i : T_i, \ldots, x_n : T_n \vdash x_i : T_i]\!] =$ $\pi_i : [\![\Gamma \vdash T_1]\!] \times \ldots \times [\![\Gamma \vdash T_i]\!] \times \ldots \times [\![\Gamma \vdash T_n]\!] \to [\![\Gamma \vdash T_i]\!]$
- $[\![\Gamma; \Delta \vdash \lambda x.M : T \to S]\!] = \Lambda([\![\Gamma; \Delta, x : T \vdash M : S]\!])$
- $[\![\Gamma; \Delta \vdash MN : S]\!] = [\![\Gamma; \Delta \vdash M : T \to S]\!] \bullet [\![\Gamma; \Delta \vdash N : T]\!] =$ $\langle [\![\Gamma; \Delta \vdash M : T \to S]\!], [\![\Gamma; \Delta \vdash N : T]\!] \rangle; Ap$
- $[\![\Gamma; \Delta \vdash \Lambda X.M : \forall X.T]\!] = \overline{[\![\Gamma, X; \Delta \vdash M : T]\!]}$,
  where $^-$ *is the bijection given by the adjiunction between* $\forall$ *and* π^* *in Definition 1.9;*
- $[\![\Gamma; \Delta \vdash MS : T[S/X]]\!] =$
  $[\![\Gamma; \Delta \vdash M : \forall X.T]\!]; \langle id_{[\![\Gamma]\!]}, [\![\Gamma \vdash S]\!] \rangle^* (\widehat{id}_{\forall([\![\Gamma, Y \vdash T[Y/X]]\!])})$,
  where $^\wedge$ *is the inverse of* $^-$.

Proposition 1.1 (see [33]) *Any $2\lambda\times$-hyperdoctrine with the interpretation function $[\![\ ]\!]$ of Definition 1.10 is a model of System F.*

Definition 1.11 (Full and Faithful Completeness) *Let $\mathcal{M} = (\mathcal{C}, \boldsymbol{G}, \forall, [\![\ ]\!])$ be a $2\lambda\times$-hyperdoctrine.*
i) $\mathcal{M}$ is fully complete *w.r.t. the class of closed types $\mathcal{T}$ if, for all $T \in \mathcal{T}$,*

$$\forall f \in Hom_{\boldsymbol{G}(1)}(1, [\![\vdash T]\!]).\ \exists M.\ \vdash M : T\ \wedge\ f = [\![\vdash M : T]\!]\ .$$

ii) $\mathcal{M}$ is fully and faithfully complete *w.r.t. the class of closed types $\mathcal{T}$ if, for all $T \in \mathcal{T}$,*

$$\forall f \in Hom_{\boldsymbol{G}(1)}(1, [\![\vdash T]\!]).\ \exists!\beta\eta\text{-normal form } M.\ \vdash M : T\ \wedge\ f = [\![\vdash M : T]\!]\ .$$

Notice that, since System F is strongly normalizing, in discussing full completeness we can restrict ourselves to normal forms. Thus in Definition 1.11(i) above, full completeness is equivalent to saying that all morphisms of appropriate type are λ-definable by normal forms.

1.4 Adjoint Hyperdoctrines

We start by recalling some definitions:

Definition 1.12 (Linear Category, [18,17]) *A* linear category *is a symmetric monoidal closed category $(\mathcal{L}, I, \otimes, \multimap)$ with*

- *a symmetric monoidal comonad $(!, der, \delta, \phi, \phi')$ on $\mathcal{L}$, where $der : !(\ ) \to Id(\ )$, $\delta : !(\ ) \to !!(\ )$, $\phi : !(\ ) \otimes !(\ ) \to !((\ ) \otimes (\ ))$, $\phi' : I \to !I$;*
- *monoidal natural transformations with components $weak_A : !A \to I$ and $con_A : !A \to !A \otimes !A$ such that*
  - *each $(!A, weak_A, con_A)$ is a commutative comonoid,*
  - *$weak_A$ and con_A are $!$-coalgebra maps from $(!A, \delta_A)$ to (I, ϕ'_I), and from $(!A, \delta_A)$ to $(!A \otimes !A, \delta_A \otimes \delta_A; \phi_{!A,!A})$, respectively.*
  - *all coalgebra maps between free $!$-coalgebras preserve the canonical structure.*

Definition 1.13 (Adjoint Model, [17]) *An* adjoint model *is specified by*

(1) a symmetric monoidal closed category $(\mathcal{L}, I, \otimes, \multimap)$;
(2) a cartesian closed category $(\mathcal{C}, 1, \times, \to)$;
(3) a symmetric monoidal adjunction from $\mathcal{C}$ to $\mathcal{L}$.

Now we give the *indexed version* of the notion of adjoint model:

Definition 1.14 (Indexed Adjoint Model) *An* indexed adjoint model *is specified by*

(1) a symmetric monoidal closed indexed category $\boldsymbol{L} : \mathcal{C}^{op} \to \mathrm{SMCCat}$*, where* SMCCat *is the category of symmetric monoidal closed categories and strict monoidal closed functors;*
(2) a cartesian closed indexed category $\boldsymbol{G} : \mathcal{C}^{op} \to \mathrm{CCCat}$*, where* CCCat *is the category of cartesian closed categories and strict cartesian closed functors;*
(3) a symmetric monoidal indexed adjunction from $\boldsymbol{G}$ *to* $\boldsymbol{L}$.

In the following definition, which is inspired by [35], we capture those $2\lambda\times$-hyperdoctrines which arise from a co-Kleisli construction over an *indexed linear category*.

Definition 1.15 (Adjoint Hyperdoctrine) *An* adjoint hyperdoctrine *is a quadruple* $(\mathcal{C}, \boldsymbol{L}, \boldsymbol{G}, \forall)$*, where:*

- $\mathcal{C}$ *is the* base category*, it has finite products, and it consists of a distinguished object* $\mathcal{U}$ *which generates all other objects using the product operation* $\times$*. We will denote by* $\mathcal{U}^m$*, for* $m \geq 0$*, the objects of* $\mathcal{C}$*.*
- $\boldsymbol{L} : \mathcal{C}^{op} \to \mathrm{LCat}$ *is a* $\mathcal{C}$*-indexed linear category, where* LCat *is the category of linear categories and strict monoidal closed functors, such that: for all* $\mathcal{U}^m$*, the underlying collection of objects of the linear fibre category* $\boldsymbol{L}(\mathcal{U}^m)$ *is indexed by the collection of morphisms from* $\mathcal{U}^m$ *to* $\mathcal{U}$ *in* $\mathcal{C}$*.*
- $\boldsymbol{G} : \mathcal{C}^{op} \to \mathrm{CCCat}$ *is the* $\mathcal{C}$*-indexed cartesian closed co-Kleisli category of* $\boldsymbol{L}$*.*
- *For each object* $\mathcal{U}^m$ *of* $\mathcal{C}$*, we are given a functor* $\forall_m : \boldsymbol{G}(\mathcal{U}^m \times \mathcal{U}) \to \boldsymbol{G}(\mathcal{U}^m)$ *such that*
  · $\forall_m : \boldsymbol{G}(\mathcal{U}^m \times \mathcal{U}) \to \boldsymbol{G}(\mathcal{U}^m)$ *is right adjoint to the functor* $\boldsymbol{G}(\pi_m) : \boldsymbol{G}(\mathcal{U}^m) \to \boldsymbol{G}(\mathcal{U}^m \times \mathcal{U})$*, where* $\pi_m : \mathcal{U}^m \times \mathcal{U} \to \mathcal{U}^m$ *is the projection in* $\mathcal{C}$*;*
  · $\forall_m : \boldsymbol{G}(\mathcal{U}^m \times \mathcal{U}) \to \boldsymbol{G}(\mathcal{U}^m)$ *satisfies the* Beck-Chevalley condition.

An adjoint hyperdoctrine is, in particular, an indexed adjoint model, and it gives rise to a $2\lambda\times$-hyperdoctrine:

Theorem 1.3 *Let* $(\mathcal{C}, \boldsymbol{L}, \boldsymbol{G}, \forall)$ *be an adjoint hyperdoctrine. Then*
i) the categories $\boldsymbol{L}$ *and* $\boldsymbol{G}$ *form an indexed adjoint model;*
ii) $(\mathcal{C}, \boldsymbol{G}, \forall)$ *is an hyperdoctrine.*

Remark. In the definition of adjoint hyperdoctrine, we require the indexed categories **L** and **G** to form an adjoint model, but we assume the existence of a family of functors $\forall_m$ only on the fibre categories of **G**. Therefore, we have a model of linear *first order* types, but not of linear *higher order* types, and our definition does not capture models of L/NL System F, i.e. System F with both linear and intuitionistic types. But our notion of model is sufficient for dealing with ML-types, and for expressing axioms for full completeness at ML-types (see Section 3.1).

2 Models of PERs over a Linear Combinatory Algebra

Canonical examples of $2\lambda\times$-hyperdoctrines, and in particular of CCC, arise by considering the *Partial Equivalence Relation* (PER) category over a *combinatory algebra* (see [19], Chapter 5, Section 5.5 for more details). In this section, we show how to build a PER category from a linear combinatory algebra (LCA). Furthermore, we prove that this category forms an adjoint model with its co-kleisli category, and we show how adjoint hyperdoctrines arise from PER categories over a *linear combinatory algebra*. Finally, we present the special LCA of *partial involutions* which we will show to provide a fully-complete model at ML-types (see Section 3.2).

We start by recalling the definition of linear combinatory algebra, [2,5]:

Definition 2.1 (Linear Combinatory Algebra) *A linear combinatory algebra $\mathcal{A} = (A, \bullet, !)$ is an applicative structure $(A, \bullet)$ with a unary (injective) operation !, and distinguished elements (combinators) $B, C, I, K, W, D, \delta, F$ satisfying the following equations:*

Equation	**Principal type**	**Logical rule**
$\boldsymbol{I}x = x$	$\alpha \multimap \alpha$	Identity
$\boldsymbol{B}xyz = x(yz)$	$(\alpha \multimap \beta) \multimap (\gamma \multimap \alpha) \multimap \gamma \multimap \beta$	Cut
$\boldsymbol{C}xyz = (xz)y$	$(\alpha \multimap \beta \multimap \gamma) \multimap \beta \multimap \alpha \multimap \gamma$	Exchange
$\boldsymbol{K}x\,!y = x$	$\alpha \multimap\, !\beta \multimap \alpha$	Weakening
$\boldsymbol{W}x\,!y = x\,!y\,!y$	$(\,!\alpha \multimap\, !\alpha \multimap \beta) \multimap\, !\alpha \multimap \beta$	Contraction
$\boldsymbol{D}!x = x$	$!\alpha \multimap \alpha$	Dereliction
$\delta\,!x = !\,!x$	$!\alpha \multimap\, !\,!\alpha$	Comultiplication
$\boldsymbol{F}!x\,!y = !(xy)$	$!(\alpha \multimap \beta) \multimap\, !\alpha \multimap\, !\beta$	Closed Functoriality

*where, by abuse of notation, combinators **K** and **W** are the linear refinements of the omonymous standard combinators.*

LCA's correspond to a *Hilbert style* axiomatization of the $\multimap$, ! fragment of Linear Logic. Given an LCA $\mathcal{A} = (A, \bullet, !)$, we can form a standard CA $\mathcal{A}_s = (A, \bullet_s)$ by the "combinatory version" of Girard's translation of Intuitionistic

Logic into Linear Logic. We define: $\alpha \bullet_s \beta = \alpha \bullet !\beta$ (standard combinators can be defined in terms of the linear ones, see [5] for details).

2.1 Linear Realizability

We start by considering a *BCI-algebra*, i.e. an applicative structure $(A, \bullet)$ with B, C, I combinators. We define a PER category over a BCI-algebra, and we show that this category is symmetric monoidal closed.

Notice that, as standard combinatory algebras satisfy *combinatory completeness* w.r.t. λ-calculus, BCI-algebras satisfy *combinatory completeness* for purely *linear* λ-terms. In virtue of this, in what follows we will often use λ-notation in place of combinatory notation.

Definition 2.2 *Let $\mathcal{A} = (A, \bullet)$ be a BCI-algebra. We define the category $PER_{\mathcal{A}}$ as follows.*
Objects*:* partial equivalence relations $\mathcal{R} \subseteq A \times A$*, i.e. symmetric and transitive relations.*
Morphisms*: a morphism f from $\mathcal{R}$ to $\mathcal{S}$ is an equivalence class of the PER $\mathcal{R} \multimap \mathcal{S}$, where the PER $\mathcal{R} \multimap \mathcal{S}$ is defined by*

$$\alpha(\mathcal{R} \multimap \mathcal{S})\beta \quad \textit{iff} \quad \forall \gamma \,\mathcal{R}\, \gamma'.\ \alpha \bullet \gamma \,\mathcal{S}\, \beta \bullet \gamma' \ .$$

On BCI-algebras, standard *pairing* gives rise to a *tensor product*, but the definition of tensor product requires some care:

Lemma 2.1 *Let $\mathcal{A} = (A, \bullet)$ be a BCI-algebra. Let P be the* pairing combinator*, i.e. (using λ-notation) $P = \lambda xyz.zxy$. Then, for all PERs $\mathcal{R}, \mathcal{S}$, let $\mathcal{R} \otimes \mathcal{S}$ be the PER defined as the* transitive closure *of the following relation:*

$$\mathcal{R} \otimes' \mathcal{S} = \{(P\alpha\beta, P\alpha'\beta') \mid \alpha \,\mathcal{R}\, \alpha' \ \wedge \ \beta \,\mathcal{S}\, \beta'\} \ .$$

Notice in particular that, if the BCI-algebra is *affine*, i.e. it is a BCK-algebra, then the relation $\mathcal{R} \otimes' \mathcal{S}$ is already transitive, since, using projections, we get: $P\alpha\beta = P\alpha'\beta' \implies \alpha = \alpha' \ \wedge \ \beta = \beta'$.

Proposition 2.1 *Let $\mathcal{A} = (A, \bullet)$ be a BCI-algebra. Then $PER_{\mathcal{A}}$ is a symmetric monoidal closed category.*

Proof. Let $\otimes : \mathrm{PER}_{\mathcal{A}} \times \mathrm{PER}_{\mathcal{A}} \to \mathrm{PER}_{\mathcal{A}}$ be defined on objects as in Lemma 2.1. For all arrows $f : \mathcal{R} \to \mathcal{S}$, $f' : \mathcal{R}' \to \mathcal{S}'$, we define $f \otimes f' : \mathcal{R} \otimes \mathcal{R}' \to \mathcal{S} \otimes \mathcal{S}'$ by $[\lambda z.zA]$, where $A = \lambda xy.P(fx)(f'y)$. The PER $I = \{(\mathbf{I}, \mathbf{I})\}$ plays the role of *tensor identity.*

The following are natural isomorphisms:
$\rho_{\mathcal{R}} : \mathcal{R} \otimes \mathbf{I} \to \mathcal{R}, \quad \rho_{\mathcal{R}} = [\lambda z.z(\lambda xy.yx)]$,
$\alpha_{\mathcal{R}_1,\mathcal{R}_2,\mathcal{R}_3} : (\mathcal{R}_1 \otimes \mathcal{R}_2) \otimes \mathcal{R}_3 \to \mathcal{R}_1 \otimes (\mathcal{R}_2 \otimes \mathcal{R}_3), \quad \alpha_{\mathcal{R}_1,\mathcal{R}_2,\mathcal{R}_3} = [\lambda z.zU]$,
where $U = \lambda x.xZ$, and $Z = \lambda xyz.Px(Pyz)$;
$\sigma_{\mathcal{R}_1,\mathcal{R}_2} : \mathcal{R}_1 \otimes \mathcal{R}_2 \to \mathcal{R}_2 \otimes \mathcal{R}_1, \quad \sigma_{\mathcal{R}_1,\mathcal{R}_2} = [\lambda z.z(\lambda zz'.Pz'z)]$;
$\Lambda : (\mathcal{R}_1 \otimes \mathcal{R}_2 \multimap \mathcal{R}_3) \to (\mathcal{R}_1 \multimap \mathcal{R}_2 \multimap \mathcal{R}_3), \quad \Lambda = [\lambda fxy.f(Pxy)]$. $\Box$

Now we show how an LCA gives rise to a linear category.

Proposition 2.2 *Let $\mathcal{A} = (A, \bullet, !)$ be an LCA. Let $\boldsymbol{!} : PER_{\mathcal{A}} \to PER_{\mathcal{A}}$ be the functor defined by:*

- $\forall\, \mathcal{R}, \quad \boldsymbol{!}\,\mathcal{R} = \{(!\alpha, !\beta) \mid \alpha\, \mathcal{R}\, \beta\}$
- $\forall f : \mathcal{R}_1 \to \mathcal{R}_2, \quad \boldsymbol{!}f = [\boldsymbol{F}!f]$.

Then $(\ \boldsymbol{!}, D, \delta, \phi, \phi'\ )$ is a symmetric monoidal comonad, where

- $\phi_{\mathcal{R}_1,\mathcal{R}_2} : \ \boldsymbol{!}\,\mathcal{R}_1 \otimes \boldsymbol{!}\,\mathcal{R}_2 \to \boldsymbol{!}(\mathcal{R}_1 \otimes \mathcal{R}_2)$ *is defined by* $\phi_{\mathcal{R}_1,\mathcal{R}_2} = [\lambda z.zA]$, *where* $A = \lambda xy.F(F\,!Px)y$;
- $\phi' : I \simeq \boldsymbol{!}I$ *is* $[\delta]_{I \to \boldsymbol{!}I}$.

Notice that the following isomorphisms hold immediately in PER categories over LCA's:

Lemma 2.2 *Let $\mathcal{A} = (A, \bullet, !)$ be an LCA satisfying extensionality of pairs. Then, for all PERs $\mathcal{R}, \mathcal{S}$,*

(1) (Idempotency of !) $[D] : \ \boldsymbol{!}\,\boldsymbol{!}\,\mathcal{R} \simeq \boldsymbol{!}\,\mathcal{R} : [\delta]$;
(2) (Uniformity of Threads) $\psi : \ \boldsymbol{!}\,\mathcal{R} \multimap \boldsymbol{!}\,\mathcal{S} \simeq \boldsymbol{!}\,\mathcal{R} \multimap \mathcal{S} : (\cdot)^{\dagger}$, *where* $\psi = [\lambda x.x; D]$;
Equivalently: $\forall \alpha \in \ \boldsymbol{!}\,\mathcal{R} \multimap \boldsymbol{!}\,\mathcal{S}$, $(\alpha; [D])^{\dagger} = \alpha$;
(3) (Commutativity of $\bigcap$ w.r.t. !) $\bigcap_X !\,\mathcal{R} \simeq !(\bigcap_X \mathcal{R})$.

The second isomorphism in Lemma 2.2 above is relevant for full completeness. In fact, as we will see in Section 2, this isomorphism amounts exactly to the *Uniformity of Threads Axiom* in our axiomatization of full completeness. The isomorphisms of Lemma 2.2 above highlight the fact that the PER category is a "degenerate" model of linear logic.

Theorem 2.1 *Let $\mathcal{A} = (A, \bullet, !)$ be an LCA. Then*

- *The category $PER_{\mathcal{A}}$ is linear.*
- *The co-Kleisli category $(PER_{\mathcal{A}})_{\boldsymbol{!}}$, induced by the comonad $\boldsymbol{!}$ on the category $PER_{\mathcal{A}}$, is cartesian closed.*
- *The categories $PER_{\mathcal{A}}$ and $(PER_{\mathcal{A}})_{\boldsymbol{!}}$ form an adjoint model.*
- *The category $(PER_{\mathcal{A}})_{\boldsymbol{!}}$ is isomorphic to the category $PER_{\mathcal{A}_s}$, where $PER_{\mathcal{A}_s}$ is*

the category obtained by standard realizability from the standard combinatory algebra $\mathcal{A}_s$.

Finally, we show how to build an adjoint hyperdoctrine from an LCA:

Theorem 2.2 (PER Adjoint Hyperdoctrine) *Let $\mathcal{A} = (A, \bullet, !)$ be an LCA satisfying extensionality of pairs. Then $\mathcal{A}$ gives rise to an adjoint hyperdoctrine $(\mathcal{C}, \boldsymbol{L}, \boldsymbol{G}, \forall)$, by defining:*

- $\mathcal{C}$ **:** *Let $\mathcal{U}$ be the set $\{\mathcal{R} \mid \mathcal{R}$ is a PER on $A\}$. The objects of $\mathcal{C}$, $\mathcal{U}^n$, for $n \geq 0$, are the finite products in* Set *of n copies of the set $\mathcal{U}$, in particular $\mathcal{U}^0$ is the terminal object in* Set. *A morphism in $\mathcal{C}$, $f : \mathcal{U}^n \to \mathcal{U}^m$, is a set-theoretic function from $\mathcal{U}^m$ to $\mathcal{U}^n$.*
- $\boldsymbol{L}$ **:** *The morphisms in the fibre category $\boldsymbol{L}(\mathcal{U}^m)$ from $h_1 : \mathcal{U}^m \to \mathcal{U}$ to $h_2 : \mathcal{U}^m \to \mathcal{U}$ are the equivalence classes of the PER $\bigcap_{\vec{X}\in\mathcal{U}^m}(h_1\vec{X} \multimap h_2\vec{X})$. For any object $f : \mathcal{U}^m \to \mathcal{U}$ in $\boldsymbol{L}(\mathcal{U}^m)$, we define $!f$ to be $\lambda\vec{X}.!(f\vec{X})$. For any morphism $f : \mathcal{U}^m \to \mathcal{U}^n$ in $\mathcal{C}$, we define the behaviour of the functor $\boldsymbol{L}(f) : \boldsymbol{L}(\mathcal{U}^n) \to \boldsymbol{L}(\mathcal{U}^m)$ on morphisms by: for any morphism $H : h_1 \to h_2$ in $\boldsymbol{L}(\mathcal{U}^n)$, $H = \Lambda\vec{X}.H' \in \bigcap_{\vec{X}}(h_1\vec{X} \multimap h_2\vec{X})$, let $\boldsymbol{L}(f)(H) : \boldsymbol{L}(f)(h_1) \to \boldsymbol{L}(f)(h_2)$ be $\Lambda\vec{X}.H' \circ f(\vec{X}) \in \bigcap_{\vec{X}}(\boldsymbol{L}(f)(h_1)\vec{X} \multimap \boldsymbol{L}(f)(h_2)\vec{X})$.*
- $\forall$ **:** *The functor $\forall_m : \boldsymbol{L}(\mathcal{U}^m \times \mathcal{U}) \to \boldsymbol{L}(\mathcal{U}^m)$ is defined as follows. For any $h : \mathcal{U}^m \times \mathcal{U} \to \mathcal{U}$, $\forall_m(h) = \lambda\vec{X}.\bigcap_Y h(\vec{X}, Y)$. For any morphism $H : h_1 \to h_2$ in $\boldsymbol{L}(\mathcal{U}^m \times \mathcal{U})$, $\forall_m(H) = H$.*

2.2 The Affine Combinatory Algebra of Partial Involutions

Many examples of LCAs arise from Abramsky's categorical version of Girard's Geometry of Interaction (GoI) construction, based on *traced symmetric monoidal* categories, [2,1,5]. A basic example of GoI LCA, introduced in [2], can be defined on the space $[\mathbf{N} \rightharpoonup \mathbf{N}]$ of partial functions from natural numbers into natural numbers, applying the GoI construction to the the traced category *Pfn* of sets and partial functions. Here we briefly recall the definition of this LCA, without discussing the categorical framework (see [2,1,5] for more details). The LCA of partial involutions, which will be shown to provide a fully-complete model for ML-types (see Section 3.2), arises as subalgebra of this.

Let us consider the space $[\mathbf{N} \rightharpoonup \mathbf{N}]$ of partial functions from natural numbers to natural numbers. For any injective $\alpha \in [\mathbf{N} \rightharpoonup \mathbf{N}]$, we denote by α^{-1} the inverse of α. Now we show how we can endow the space $[\mathbf{N} \rightharpoonup \mathbf{N}]$ with a structure of LCA. Actually, the algebra which we will obtain is *affine*, i.e. it has a full $\boldsymbol{K}$-combinator. We start by fixing the following two injective *coding*

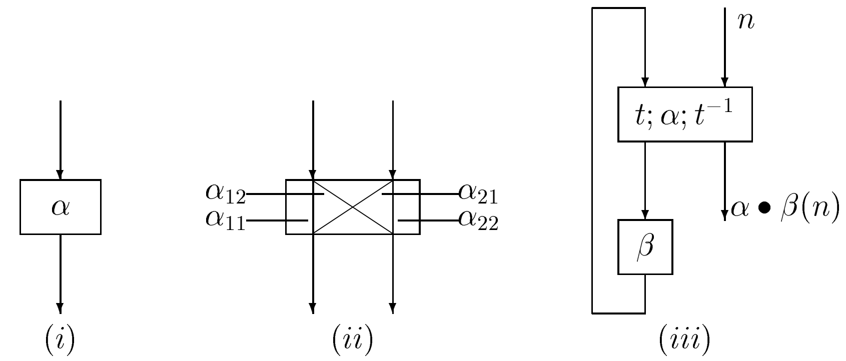


Fig. 1. Geometrical description of linear application.

functions t and p:

$$t : \mathbf{N} + \mathbf{N} \to \mathbf{N} \qquad p : \mathbf{N} \times \mathbf{N} \to \mathbf{N} \ .$$

By abuse of notation, we denote by t^{-1} and p^{-1} the inverses of t and p respectively. The coding t is used in order to define application, it allows to transform an one-input/one-output function into a two-input/two-output function. The coding p is used for creating *infinitely* many copies of an one-input/one-output function α, i.e. for defining $!\alpha$.

We now explain how application is computed *geometrically*, using the language of "boxes and wires" which arises in the general setting of traced symmetric monoidal categories (see [27] for the general categorical treatment).

Let us represent an one-input/one-output function $\alpha \in [\mathbf{N} \rightharpoonup \mathbf{N}]$ by a one-input-port/one-output-port box as in Fig. 1(i).

In order to define the application $\alpha \bullet \beta$, for $\alpha, \beta \in [\mathbf{N} \rightharpoonup \mathbf{N}]$, we regard α as a two-input/two-output function via the coding t. In particular, $t;\alpha;t^{-1} : \mathbf{N} + \mathbf{N} \rightharpoonup \mathbf{N} + \mathbf{N}$ can be described as a matrix of 4 one-input/one-output functions:

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{pmatrix}$$

where $\alpha_{ij} = in_i; t; \alpha; t^{-1}; in_j^{-1} : \mathbf{N} \rightharpoonup \mathbf{N}$ account for the contribution from the i-th input wire into the j-th output wire (see Fig. 1(ii)).

The result of the application $\alpha \bullet \beta$ is the following one-input/one-output function (see Fig. 1(iii)).

$$\alpha \bullet \beta = \alpha_{22} \cup \alpha_{21}; (\beta; \alpha_{11})^\star; \beta; \alpha_{12} \ , \tag{1}$$

where $\cup$ denotes union of graph relations, and $(\beta;\alpha_{11})^\star$ denotes $\bigcup_{n\geq 0}(\beta;\alpha_{11})^n$.

The formula (1) above for computing the application is essentially the *Execution Formula* from Girard's Geometry of Interaction, [22].

The definition of the !-operation on our applicative structure is quite simple. The operation ! is intended to produce, from a single copy of α, *infinitely* many copies of α. These are obtained by simply tagging each of these copies with a natural number, i.e. we define:

$$!\alpha = p^{-1}; (id_{\mathbf{N}} \times \alpha); p \ .$$

Finally, we are left to show that (affine) combinators can be defined on the structure $([\mathbf{N} \rightharpoonup \mathbf{N}], \bullet, !)$. The formal (algebraic) definition of the combinators is the following:

Definition 2.3 (Combinators) *For $\boldsymbol{X} \in \{\boldsymbol{I}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{K}, \boldsymbol{W}, \boldsymbol{D}, \delta, \boldsymbol{F}\}$, let*

$$\boldsymbol{X} = s_{\boldsymbol{X}}^{-1}; f_{\boldsymbol{X}}; s_{\boldsymbol{X}} \ ,$$

where:

$\boldsymbol{I}$:

- $s_{\boldsymbol{I}} = t$
- $f_{\boldsymbol{I}} : \boldsymbol{N} + \boldsymbol{N} \rightharpoonup \boldsymbol{N} + \boldsymbol{N}$ *is defined by:*
  · $\forall n.\ f_{\boldsymbol{I}}(r, n) = (l, n)$
  · $\forall n.\ f_{\boldsymbol{I}}(l, n) = (r, n)$.

$\boldsymbol{B}$:

- $s_{\boldsymbol{B}} : (((\boldsymbol{N} + \boldsymbol{N}) + (\boldsymbol{N} + \boldsymbol{N})) + \boldsymbol{N}) + \boldsymbol{N} \rightharpoonup \boldsymbol{N}$ *is defined by*

$$s_{\boldsymbol{B}} = ((t + t) + id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}; (t + id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}; t + id_{\boldsymbol{N}}; t$$

- $f_{\boldsymbol{B}} : (((\boldsymbol{N} + \boldsymbol{N}) + (\boldsymbol{N} + \boldsymbol{N})) + \boldsymbol{N}) + \boldsymbol{N} \rightharpoonup (((\boldsymbol{N} + \boldsymbol{N}) + (\boldsymbol{N} + \boldsymbol{N})) + \boldsymbol{N}) + \boldsymbol{N}$ *is the function defined by the following equations together with their symmetric closure:*
  · $\forall n.\ f_{\boldsymbol{B}}(r, n) = (l, (l, (l, (r, n))))$
  · $\forall n.\ f_{\boldsymbol{B}}(l, (l, (l, (l, n)))) = (l, (l, (r, (r, n))))$
  · $\forall n.\ f_{\boldsymbol{B}}(l, (l, (r, (l, n)))) = (l, (r, n))$.

$\boldsymbol{C}$:

- $s_{\boldsymbol{C}} : ((((\boldsymbol{N} + \boldsymbol{N}) + \boldsymbol{N}) + \boldsymbol{N}) + \boldsymbol{N}) + \boldsymbol{N} \rightharpoonup \boldsymbol{N}$ *is defined by*

$$s_{\boldsymbol{C}} = (((t+id_{\boldsymbol{N}})+id_{\boldsymbol{N}})+id_{\boldsymbol{N}})+id_{\boldsymbol{N}}; ((t+id_{\boldsymbol{N}})+id_{\boldsymbol{N}})+id_{\boldsymbol{N}}; (t+id_{\boldsymbol{N}})+id_{\boldsymbol{N}}; t+id_{\boldsymbol{N}}; t$$

- $f_{\boldsymbol{C}} : ((((\boldsymbol{N}+\boldsymbol{N})+\boldsymbol{N})+\boldsymbol{N})+\boldsymbol{N})+\boldsymbol{N} \rightharpoonup ((((\boldsymbol{N}+\boldsymbol{N})+\boldsymbol{N})+\boldsymbol{N})+\boldsymbol{N})+\boldsymbol{N}$ *is the function defined by the following equations together with their symmetric closure:*
  · $\forall n.\ f_{\boldsymbol{C}}(r,n) = (l,(l,(l,(r,n))))$
  · $\forall n.\ f_{\boldsymbol{C}}(l,(r,n)) = (l,(l,(l,(l,n))))$
  · $\forall n.\ f_{\boldsymbol{C}}(l,(l,(r,n))) = (l,(l,(l,(r,n))))$.

$\boldsymbol{K}$:

- $s_{\boldsymbol{K}} : (\boldsymbol{N}+\boldsymbol{N})+\boldsymbol{N} \rightharpoonup \boldsymbol{N}$ *is defined by*

$$s_{\boldsymbol{K}} = t + id_{\boldsymbol{N}}; t$$

- $f_{\boldsymbol{K}} : (\boldsymbol{N}+\boldsymbol{N})+\boldsymbol{N} \rightharpoonup (\boldsymbol{N}+\boldsymbol{N})+\boldsymbol{N}$ *is the function defined by the following equations:*
  · $\forall n.\ f_{\boldsymbol{K}}(r,n) = (l,(l,n))$
  · $\forall n.\ f_{\boldsymbol{K}}(l,(l,n)) = (r,n)$.

$\boldsymbol{W}$:

- $s_{\boldsymbol{W}} : ((((\boldsymbol{N}\times\boldsymbol{N})+(\boldsymbol{N}\times\boldsymbol{N}))+\boldsymbol{N})+(\boldsymbol{N}\times\boldsymbol{N}))+\boldsymbol{N} \rightharpoonup \boldsymbol{N}$ *is defined by*

$$s_{\boldsymbol{W}} = ((q + id_{\boldsymbol{N}}) + p) + id_{\boldsymbol{N}}; (((t \times id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}; \\ ((p + id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}; (t + id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}; t + id_{\boldsymbol{N}}; t$$

  where $q : (\boldsymbol{N}\times\boldsymbol{N})+(\boldsymbol{N}\times\boldsymbol{N}) \to (\boldsymbol{N}+\boldsymbol{N})\times\boldsymbol{N}$ *is the isomorphism mapping* $(l,(i,n)) \mapsto ((l,i),n)$ *and* $(r,(i,n)) \mapsto ((r,i),n)$.
- $f_{\boldsymbol{W}} : ((((\boldsymbol{N}\times\boldsymbol{N})+(\boldsymbol{N}\times\boldsymbol{N}))+\boldsymbol{N})+(\boldsymbol{N}\times\boldsymbol{N}))+\boldsymbol{N} \rightharpoonup ((((\boldsymbol{N}\times\boldsymbol{N})+(\boldsymbol{N}\times\boldsymbol{N}))+\boldsymbol{N})+(\boldsymbol{N}\times\boldsymbol{N}))+\boldsymbol{N}$ *is the function defined by the following equations together with their symmetric closure:*
  · $\forall n.\ f_{\boldsymbol{W}}(r,n) = (l,(l,(r,n)))$
  · $\forall n,i.\ f_{\boldsymbol{W}}(l,(r,(t(r,i),n))) = (l,(l,(l,(r,(i,n)))))$
  · $\forall n,i.\ f_{\boldsymbol{W}}(l,(r,(t(l,i),n))) = (l,(l,(l,(l,(i,n)))))$.

$\boldsymbol{D}$:

In order to define $\boldsymbol{D}$*, we need to fix* $i \in \boldsymbol{N}$*. Then*

- $s_{\boldsymbol{D}} : (\boldsymbol{N}\times\boldsymbol{N})+\boldsymbol{N} \rightharpoonup \boldsymbol{N}$ *is defined by*

$$s_{\boldsymbol{D}} = p + id_{\boldsymbol{N}}; t$$

- $f_{\boldsymbol{D}} : (\boldsymbol{N}\times\boldsymbol{N})+\boldsymbol{N} \rightharpoonup (\boldsymbol{N}\times\boldsymbol{N})+\boldsymbol{N}$ *is the function defined by the following equations:*
  · $\forall n.\ f_{\boldsymbol{D}}(r,n) = (l,(i,n))$

· $\forall n.\ f_{\boldsymbol{D}}(l,(i,n)) = (r,n)$.

δ :

In order to define δ, we need to fix $i, j \in \boldsymbol{N}$. Then

- $s_\delta : (\boldsymbol{N} \times \boldsymbol{N}) + (\boldsymbol{N} \times (\boldsymbol{N} \times \boldsymbol{N})) \rightharpoonup \boldsymbol{N}$ *is defined by*

$$s_\delta = p + (id_{\boldsymbol{N}} \times p); id_{\boldsymbol{N}} + p; t$$

- $f_\delta : (\boldsymbol{N} \times \boldsymbol{N}) + (\boldsymbol{N} \times (\boldsymbol{N} \times \boldsymbol{N})) \rightharpoonup (\boldsymbol{N} \times \boldsymbol{N}) + (\boldsymbol{N} \times (\boldsymbol{N} \times \boldsymbol{N}))$ *is the function defined by the following equations:*
  · $\forall n.\ f_\delta(r,(i,(j,n))) = (l,(p(i,j),n))$
  · $\forall n.\ f_\delta(l,(p(i,j),n)) = (r,(i,(j,n)))$.

$\boldsymbol{F}$:

In order to define $\boldsymbol{F}$, we need to fix $i \in \boldsymbol{N}$. Then

- $s_{\boldsymbol{F}} : ((\boldsymbol{N} \times (\boldsymbol{N} + \boldsymbol{N})) + (\boldsymbol{N} \times \boldsymbol{N})) + (\boldsymbol{N} \times \boldsymbol{N}) \rightharpoonup \boldsymbol{N}$ *is defined by*

$$s_{\boldsymbol{F}} = ((id_{\boldsymbol{N}} \times t) + p) + p; (p + id_{\boldsymbol{N}}) + id_{\boldsymbol{N}}; t + id_{\boldsymbol{N}}; t$$

- $f_{\boldsymbol{F}} : ((\boldsymbol{N} \times (\boldsymbol{N} + \boldsymbol{N})) + (\boldsymbol{N} \times \boldsymbol{N})) + (\boldsymbol{N} \times \boldsymbol{N}) \rightharpoonup ((\boldsymbol{N} \times (\boldsymbol{N} + \boldsymbol{N})) + (\boldsymbol{N} \times \boldsymbol{N})) + (\boldsymbol{N} \times \boldsymbol{N})$ *is the function defined by the following equations together with their symmetric closure:*
  · $\forall n.\ f_{\boldsymbol{F}}(r,(i,n)) = (l,(l,(i,(r,n))))$
  · $\forall n.\ f_{\boldsymbol{F}}(l,(r,(i,n))) = (l,(l,(i,(l,n))))$.

There is a simple, intutitive, geometrical explanation of these combinators, which makes use of the language of boxes and wires. For example, let us consider the identity combinator $\boldsymbol{I}$. Since $\boldsymbol{I}$ has to satisfy the equation $\boldsymbol{I}x = x$, in order to define $\boldsymbol{I}$, it is convenient to regard $\boldsymbol{I}$ as a two-input/two-output function, up-to-coding (see Fig. 2). The Identity combinator just copies information from the lefthand input-wire to the righthand output-wire, and vice versa from the righthand input-wire to the lefthand output-wire.

The fact that $\boldsymbol{I}$ satsfies the identity equation has a simple geometrical expla-

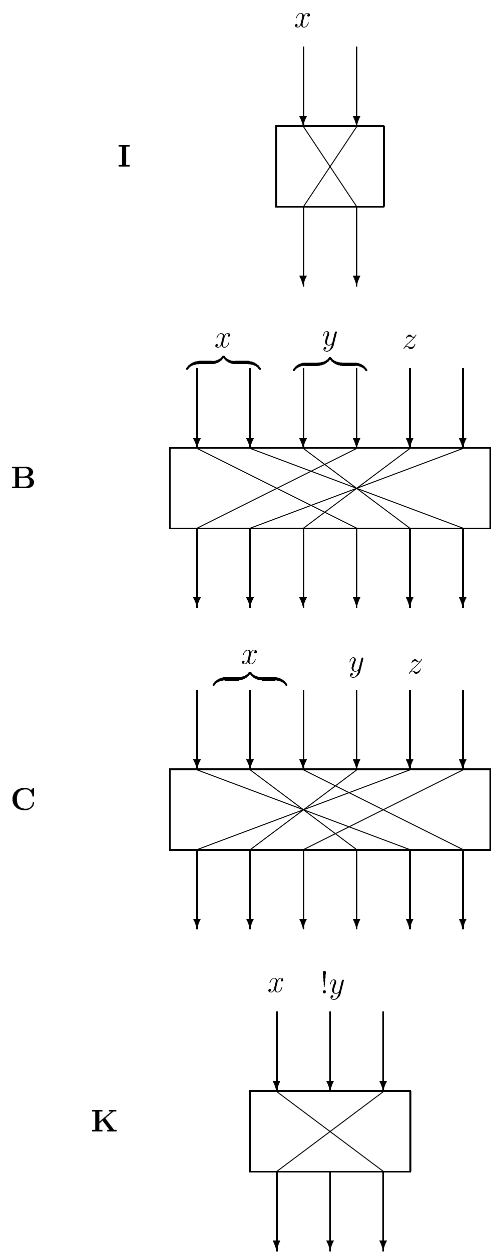
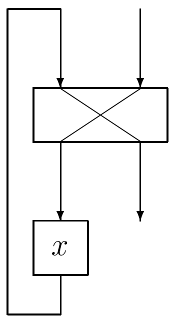


Fig. 2. IBCK-combinators.

nation. Let us apply $\boldsymbol{I}$ to a partial function x:



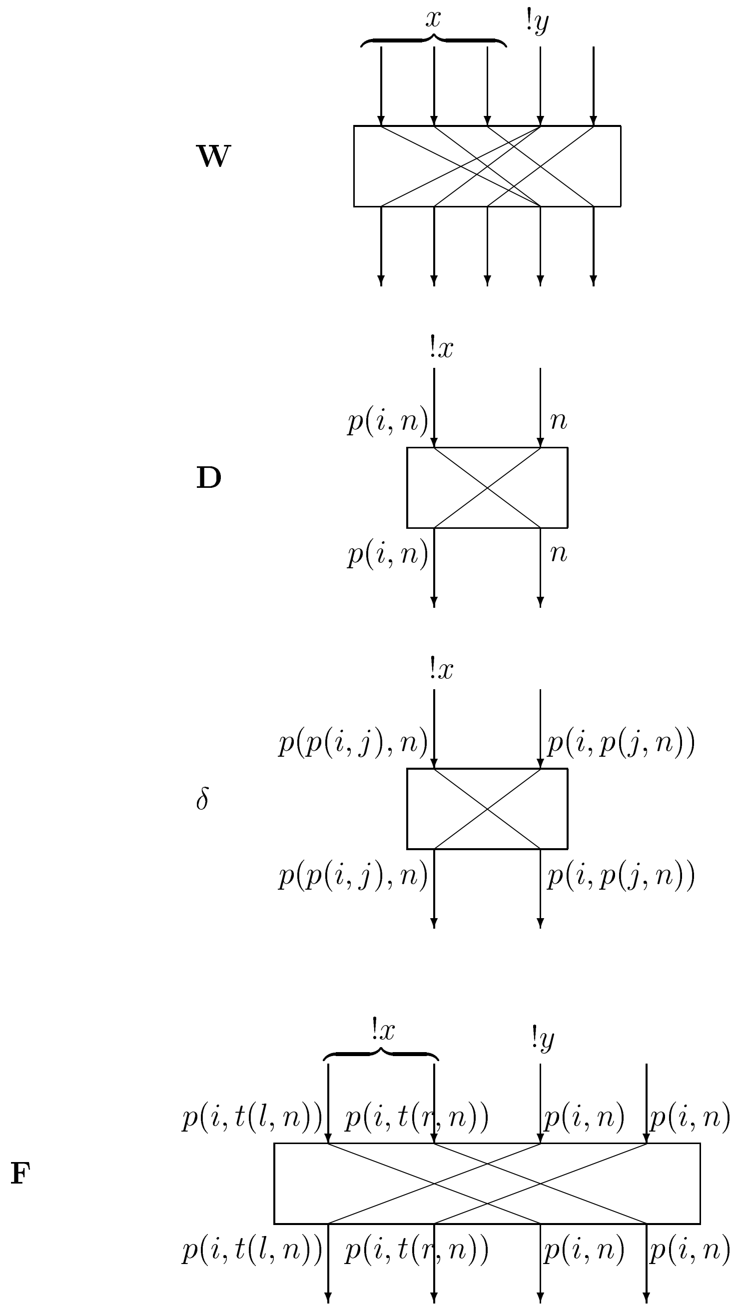
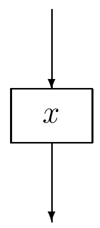


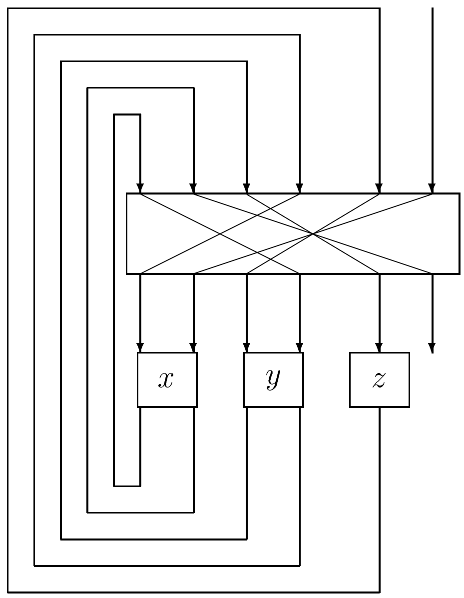
Fig. 3. WDδF-combinators.

Now *yank* the string connecting the input and the output wires of the result of the application, forgetting about the box corresponding to ***I***. This gives us immediately the expected result:

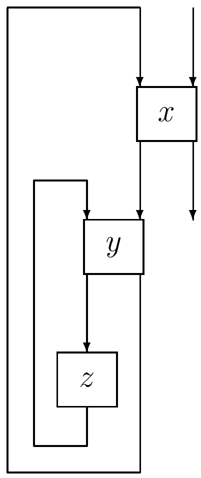


Our argument is based on the *Yanking Property* of the trace on the symmetric monoidal category *Pfn* underlying our combinatory algebra. In particular, *Yanking* is one of the axioms characterizing the trace operation in the general setting of traced symmetric monoidal categories.

Let us now consider the combinator $\boldsymbol{B}$ which satisfies the equation $\boldsymbol{B}xyz = x(yz)$. In order to define the box for $\boldsymbol{B}$ (and that of any other purely linear combinator), we only need to determine how many input wires (and correspondingly output wires) this box should have, and how these wires have to be connected inside the box. The number of input/output wires depends on the number of arguments which the combinator takes, and on the role played by these arguments, i.e. whether they just appear as arguments in the righthand side of the equation satisfied by the combinator or they are used as functions of one or more arguments. Concretely, the box for $\boldsymbol{B}$ (see Fig. 2) has two input (and two output) wires for x and two input (and two output) wires for y, since both x and y are applied to an argument, one input (and one output) wire for z, which appears only as argument, plus one extra input (and one output) wire, along which the input-token (output-token) is intended to enter (exit). The connections of the wires inside the box for $\boldsymbol{B}$ are determined by the control flow between x, y, z in the righthand part of the equation. First of all, the control flow passes from the input port of $\boldsymbol{B}$ to the input port of x. The second port of x is then connected to the input port of y, while the second port of y is connected to the unique port of z. The remaining connections are then obtained by symmetry. Now let us compute the result of the application of $\boldsymbol{B}$ to x, y, z:



Pulling the global input/output string, and forgetting about the box corresponding to $\boldsymbol{B}$, we get the expected result, i.e.:



Now we briefly discuss the remaining combinators. The combinator $\boldsymbol{C}$ (see Fig. 2) can be explained in a similar way as $\boldsymbol{B}$. The affine combinator $\boldsymbol{K}$ simply forgets about its second argument y. In order to define $\boldsymbol{W}$ (see Fig. 3), we use the isomorphism $q : (\mathbf{N} \times \mathbf{N}) + (\mathbf{N} \times \mathbf{N}) \to (\mathbf{N} + \mathbf{N}) \times \mathbf{N}$ to map an element of the form $!y$ to the pair $(\,!y,\,!y)$. The behaviour of $\boldsymbol{D}, \delta, \boldsymbol{F}$ can be explained similarly.

Essentially, all the combinators of Fig. 2 and 3 are functions that mediate the required interactions between the arguments simply by copying information between the various ports.

There are many possible conditions that can be imposed on partial functions in order to cut down the space $[\mathbf{N} \rightharpoonup \mathbf{N}]$, still maintaining closure under application, !, and all the affine combinators. The subalgebra which gives rise to the fully-complete model of Section 3.2 is obtained by considering *partial involutions*:

Definition 2.4 *Let* $f : \boldsymbol{N} \rightharpoonup \boldsymbol{N}$. f *is a* partial involution *if and only if its graph is a symmetric relation. Let us denote by* $[\boldsymbol{N} \rightharpoonup_{\mathtt{Inv}} \boldsymbol{N}]$ *the space of partial involutions from* $\boldsymbol{N}$ *to* $\boldsymbol{N}$.

One can check that partial involutions are closed under the application, the !-operation, and all the combinators of Definition 2.3, i.e.:

Proposition 2.3 $\mathcal{A}_{\mathtt{PInv}} = ([\boldsymbol{N} \rightharpoonup_{\mathtt{Inv}} \boldsymbol{N}], \bullet, !)$ *is an affine combinatory algebra.*

$\mathcal{A}_{\mathtt{PInv}}$ is a highly constrained algebra, in which all computations are *reversible*, [4]. Partial involutions are reminescent of *copy-cat* strategies of game categories, in that the only computational effect that they have is that of *copying* information from input to output wires. Partial involutions f on a set S correspond biuniquely to pair-wise disjoint families of subsets $\{x, y\}$ of S, where $\{x, y\}$ is in the family if and only if $f(x) = y$ (and hence also $f(y) = x$). We can think of these as abstract families of "axiom links" as in the proof-nets of Linear Logic.

In [10], in order to provide a fully-abstract model for PCF, constraints of a different nature are put on the space $[\mathbf{N} \rightharpoonup \mathbf{N}]$, so as to capture only functions representing strategies in the style of [8].

3 System F

This section divides in two parts. In the first part, we provide an *axiomatization* of adjoint hyperdoctrines which are fully complete for ML-types. In the second part, we use this axiomatization for proving that the PER model induced by the LCA of partial involutions $\mathcal{A}_{\mathtt{PInv}}$ is fully complete for ML-types.

3.1 Axiomatizing Models Fully Complete for ML Types

We isolate sufficient conditions on adjoint hyperdoctrine models for System F, in order to guarantee full completeness at ML-polymorphic types. These conditions amount to the six axioms of Subsection 3.1.1. Our axiomatization of full completeness for ML polymorphism is in the line of the work in [3], where an axiomatic approach to full abstraction/full completeness for PCF/simply typed λ-calculus is presented. These axiomatizations are inspired by the proof of full abstraction of the Game Semantics model for PCF of [8]. Our axiomatization of full completeness for ML-types consists of two parts:

(1) Axioms for ensuring the Decomposition Theorem. This theorem allows to recover the *top-level structure* of the (possibly infinite) Böhm tree denoted by morphisms from the terminal object into the interpretation of an ML-type in the fibre category $\mathbf{G}(1)$. The axioms for the Decomposition Theorem (Axioms 1–5 of Section 3.1.1) make essential use of the linear category underlying an adjoint hyperdoctrine. These axioms (apart from the axioms 1 and 3), are expressed by requiring some canonical maps between suitable spaces of morphisms in the fibre categories $\mathbf{L}(\vec{\mathcal{U}})$ to be isomorphisms.

(2) A *Finiteness Axiom*, which allows to rule out infinite Böhm trees from the model.

Notice that, by definition of interpretation function on a hyperdoctrine (Definition 1.10), morphisms f in $\mathbf{G}(1)$ from the terminal object of $\mathbf{G}(1)$ into $[\![\vdash T]\!]$, where $T = \forall \vec{X}.T_1 \to \ldots \to T_n \to X_k$ is a closed ML-type, are λ-definable if and only if morphisms of $\mathbf{G}(\vec{\mathcal{U}})$ from $\times_{i=1}^{n}[\![\vec{X} \vdash T_i]\!]$ into $[\![\vec{X} \vdash X_k]\!]$ are λ-definable. Namely, $f = [\![\vdash \Lambda\vec{X}.\vec{x} : \vec{T}.x_i M_1 \ldots M_{q_i} : \forall\vec{X}.\vec{T} \to X_k]\!]$ if and only if $\vec{\Lambda}^{-1}(\widehat{f}) = [\![\vec{X}; \vec{x} : \vec{T} \vdash x_i M_1 \ldots M_{q_i} : X_k]\!]$. Therefore, from now on we focus on the space of morphisms of $\mathbf{G}(\vec{\mathcal{U}})$ from $\times_{i=1}^{n}[\![\vec{X} \vdash T_i]\!]$ into $[\![\vec{X} \vdash X_k]\!]$, where $T_1, \ldots, T_n$ are simple types.

We start by presenting the main result of this section, i.e. the Decomposition Theorem. The proof of this theorem follows from the Strong Decomposition Theorem 3.2, which is proved in Section 3.1.2.

If a morphism f of $\mathbf{G}(\vec{\mathcal{U}})$ from $\times_{i=1}^{n}[\![\vec{X} \vdash T_i]\!]$ into $[\![\vec{X} \vdash X_k]\!]$ is λ-definable, then $f = [\![\vec{X}; \vec{x} : \vec{T} \vdash x_i M_1 \ldots M_{q_i} : X_k]\!]$, for some $\vec{X}; \vec{x} : \vec{T} \vdash M_1 : U_{i1}$, ..., $\vec{X}; \vec{x} : \vec{T} \vdash M_{q_i} : U_{iq_i}$. I.e., making evident the *top-level* structure of the Böhm tree:

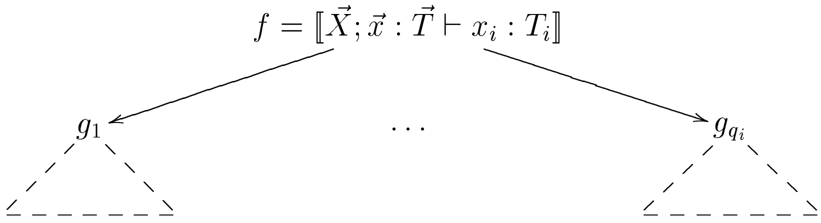
$$f = [\![\vec{X}; \vec{x} : \vec{T} \vdash x_i : T_i]\!] \bullet [\![\vec{X}; \vec{x} : \vec{T} \vdash M_1 : U_{i1}]\!] \bullet \ldots \bullet [\![\vec{X}; \vec{x} : \vec{T} \vdash M_{q_i} : U_{iq_i}]\!] \; .$$

The Decomposition Theorem allows recovering the top-level structure of the Böhm tree corresponding to f in the following sense:

Theorem 3.1 (Decomposition) *Let $(\mathcal{C}, \boldsymbol{L}, \boldsymbol{G}, \forall)$ be an adjoint hyperdoctrine satisfying Axioms 1–5 of Section 3.1.1. Let $T = T_1 \to \ldots \to T_n \to X_k$ be a simple type with $FV(T) \subseteq \{X_1, \ldots, X_n\}$, where, for all $i = 1, \ldots, n$, $T_i = U_{i1} \to \ldots \to U_{iq_i} \to X_i$. Then, for all $f \in Hom_{\boldsymbol{G}(\vec{\mathcal{U}})}(\times_{i=1}^{n}[\![\vec{X} \vdash T_i]\!], [\![\vec{X} \vdash X_k]\!])$,* *there exist* $i \in \{1, \ldots, n\}$ *and* $g_j \in Hom_{\boldsymbol{G}(\vec{\mathcal{U}})}(\times_{i=1}^{n}[\![\vec{X} \vdash T_i]\!], [\![\vec{X} \vdash U_{ij}]\!])$*, for all* $j = 1, \ldots, q_i$*, such that*

$$f = [\![\vec{X}; \vec{x} : \vec{T} \vdash x_i : T_i]\!] \bullet g_1 \ldots \bullet g_{q_i} \ .$$

Since the g's appearing in the Decomposition Thereom still live (up-to-uncurrying) in a space of morphisms denoting a simple type, we could keep on iterating the decomposition, expanding in turn these g's, thus getting a possible infinite tree from f:



If the Decomposition Theorem holds, in order to get the full completeness result, we are only left to rule out morphisms generating trees whose height is infinite, which would correspond to infinite typed Böhm trees. This is expressed in the *Finiteness Axiom* 6 below.

3.1.1 The Axioms

The first axiom is a base axiom, which expresses the fact that the type $\forall \vec{X}.X_k$ is empty, i.e. there are no closed terms typable with $\forall \vec{X}.X_k$.

Axiom 1 (Base)

$$Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(!1, \pi_k) = \emptyset \ ,$$

where 1 *is the terminal object in* $\boldsymbol{G}(\vec{\mathcal{U}})$*, and* $\pi_k : \vec{\mathcal{U}} \to \mathcal{U}$ *denotes the k-th projection in* $\boldsymbol{G}(\vec{\mathcal{U}})$*, i.e.* $\pi_k = weak_1 \otimes \ldots \otimes weak_{k-1} \otimes der_k \otimes weak_{k+1} \otimes \ldots \otimes weak_n$*.*

The following axiom allows the extraction of *one* copy of the type of the head variable, corresponding to the *first* use of this variable. The property expressed by this axiom is *truly* linear. In fact, in order to state it, we are implicitly using the canonical morphism $!A \to A \otimes !A$ to capture the idea of a "first occurrence".

Axiom 2 (Linearization of Head Occurrence)

$$case_i\{\sigma_i\}_{i=1,\ldots,n} \ : \ \coprod_{i=1}^{n} Hom^{t}_{\boldsymbol{L}(\vec{\mathcal{U}})}(h_i, (\vec{h} \multimap \pi_k)) \ \simeq \ Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\vec{h}, \pi_k) \ ,$$

where

- $\coprod$ *denotes coproduct in* Set*;*
- $\vec{h} = \otimes_{i=1}^{n} !h_i$ *and* $\forall i \in \{1,\ldots,n\}$. $h_i = \otimes_{j=1}^{q_i} !l_{ij} \multimap \pi_{p_i}$*;*
- $Hom^t_{\boldsymbol{L}(\vec{U})}(h_i, (\vec{h} \multimap \pi_k))$ *is a suitable subset of* $Hom_{\boldsymbol{L}(\vec{U})}(h_i, (\vec{h} \multimap \pi_k))$*, intended to contain only* total elements *(i.e. strict and not divergent);*
- $\sigma_i : Hom_{\boldsymbol{L}(\vec{U})}(h_i, (\vec{h} \multimap \pi_k)) \to Hom_{\boldsymbol{L}(\vec{U})}(\vec{h}, \pi_k)$ *is the following canonical morphism:*

$$
\begin{array}{c}
Hom_{\boldsymbol{L}(\vec{U})}(h_i, (\vec{h} \multimap \pi_k)) \\
\Big\downarrow {\scriptstyle \Lambda^{-1}} \\
Hom_{\boldsymbol{L}(\vec{U})}(h_i \otimes \vec{h}, \pi_k) \\
\Big\downarrow {\scriptstyle Hom_{\boldsymbol{L}(\vec{U})}(\pi_i;\tau, id_{\pi_k})} \\
Hom_{\boldsymbol{L}(\vec{U})}(\vec{h} \otimes \vec{h}, \pi_k) \\
\Big\downarrow {\scriptstyle Hom_{\boldsymbol{L}(\vec{U})}(con_{\vec{h}}, id_{\pi_k})} \\
Hom_{\boldsymbol{L}(\vec{U})}(\vec{h}, \pi_k)
\end{array}
$$

where $\tau : I \otimes \ldots \otimes I \otimes h_i \otimes I \ldots \otimes I \simeq h_i$.

The following axiom reflects a form of coherence of the type of the head variable w.r.t. the global type of the term. I.e., if $\vec{T} \to X_k$ is the type of a term, then the type of the head variable must be of the shape $\vec{U} \to X_i$, with $k = i$.

Axiom 3 (Type Coherence)

$$Hom^t_{\boldsymbol{L}(\vec{U})}(l \multimap \pi_i, h \multimap \pi_k) = \emptyset\ ,$$

if $i \neq k$.

The following axiom expresses the fact that the only thing that we can do with a *linear* functional parameter is applying it to an argument *which does not itself depend on the parameter*. Note that, again, linearity is essential here. For example, if copying were allowed, then the argument could itself contain further occurrences of the parameter.

Axiom 4 (Linear Function Extensionality)

$$Hom_{\boldsymbol{L}(\vec{U})}((\cdot), id_{\pi_k}) :\ Hom_{\boldsymbol{L}(\vec{U})}(h, l)\ \simeq\ Hom^t_{\boldsymbol{L}(\vec{U})}(l \multimap \pi_k, h \multimap \pi_k)\ .$$

The following axiom expresses the fact that morphisms from $!f$ to $!g$ in the fibre category $\mathbf{L}(\vec{U})$ have *uniform* behaviour in all threads.

Axiom 5 (Uniformity of Threads)

$$Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(id_{!h}, der_l) : \ Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\,!h,\,!l) \ \simeq \ Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\,!h, l) \ : \ \boldsymbol{\lambda} f \in Hom(\,!h, l).(f)^{\dagger} \ ,$$

where $(\ )^{\dagger}_{h,l} : Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\,!h, l) \rightarrow Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\,!h,\,!l)$ *is the canonical morphism given by the comonad* $!$.

Axioms 1–5 guarantee the validity of a strong Decomposition Theorem (see Section 3.2 below), which allows to decompose in a *unique* way all morphisms in $Hom_{\mathbf{L}(\vec{\mathcal{U}})}(\otimes_{i=1}^{n} !h_i, \pi_k)$, where $h_i = \otimes_{j=1}^{q_i} !l_j {-\!\!\circ} \pi_{p_i}$, for any $l_1, \ldots, l_{q_i}$, even if $Hom_{\mathbf{L}(\vec{\mathcal{U}})}(\otimes_{i=1}^{n} !h_i, \pi_k)$ is not the space of morphisms from $\otimes_{i=1}^{n} ![\![\vec{X} \vdash T_i]\!]$ into $[\![\vec{X} \vdash X_k]\!]$, for some simple types $T_1, \ldots, T_n$.

The final axiom in our axiomatization guarantees that the tree generated via repeated applications of the Decomposition Theorem 3.2 to morphisms in $Hom_{\mathbf{L}(\vec{\mathcal{U}})}(\otimes_{i=1}^{n} !h_i, \pi_k)$, where $h_i = \otimes_{j=1}^{q_i} !l_{ij} {-\!\!\circ} \pi_{p_i}$, is *finite*.

Axiom 6 (Finiteness) *There exists a* size function

$$\mathcal{H} : \bigcup \{ Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\otimes_{i=1}^{n} !h_i, \pi_k) \mid k \in \boldsymbol{N},\ h_i = \otimes_{j=1}^{q_i} !l_{ij} {-\!\!\circ} \pi_{p_i} \} \longrightarrow \boldsymbol{N} \ ,$$

such that

$$\forall j \in \{1, \ldots, q_i\}.\ \mathcal{H}(\vec{\Lambda}^{-1}(g_j)) < \mathcal{H}(f) \ ,$$

where the g_j*'s are defined in the Decomposition Theorem 3.2.*

3.1.2 Axiomatic Full Completeness

By Axioms 1–5, all morphisms in $\mathrm{Hom}_{\mathbf{L}(\vec{\mathcal{U}})}(\vec{h}, \pi_k)$, where $\vec{h} = \otimes_{i=1}^{n} !h_i$ and, for all $i = 1, \ldots, n$, $h_i = \otimes_{j=1}^{q_i} !l_{ij} {-\!\!\circ} \pi_{p_i}$, have a *unique* decomposition:

Theorem 3.2 (Strong Decomposition) *Let* $(\mathcal{C}, \boldsymbol{G}, \boldsymbol{L}, \forall)$ *be an adjoint hyperdoctrine satisfying Axioms 1–5 of Section 3.1.1. Let* $f \in Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\vec{h}, \pi_k)$*, where* $\vec{h} = \otimes_{i=1}^{n} !h_i$ *and, for all* $i = 1, \ldots, n$*,* $h_i = \otimes_{j=1}^{q_i} !l_{ij} {-\!\!\circ} \pi_{p_i}$*. Then there exist a unique* i *and unique* $g_1, \ldots, g_{q_i}$ *such that, for all* $j = 1, \ldots, q_i$*,* $g_j \in Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\vec{h}, l_{ij})$*, and*

$$f = con_{\vec{h}}; (\pi_k \otimes \langle g_1, \ldots, g_{q_i} \rangle^{\dagger}); Ap \ .$$

Proof. If $\vec{h} = 1$, then, by Axiom 1, we have immediately the thesis. Otherwise, by Axiom 2, there exists a unique $i \in \{1, \ldots, n\}$ and a unique $f' \in \mathrm{Hom}^{t}_{\mathbf{L}(\vec{\mathcal{U}})}(h_i, \vec{h} {-\!\!\circ} \pi_k)$ such that $f = \mathrm{con}_{\vec{h}}; \pi_i \otimes id_{\vec{h}}; \Lambda^{-1}(f')$. By Axiom 3, $\pi_i = \pi_k$. By Axiom 4, there exists a unique $g \in \mathrm{Hom}(\vec{h}, \vec{l_i})$, where $\vec{l_i} = \otimes_{j=1}^{q_i} !l_{ij}$, such that $f' = g \multimap \pi_k = \Lambda((id \otimes g); Ap)$. Then $f = \mathrm{con}_{\vec{h}}; \pi_k \otimes g; Ap$. Finally, by Axiom 5, and by the universal property of the product, we obtain $g = \langle g_1, \ldots, g_{q_i} \rangle^\dagger$. $\Box$

Finally, we can show the main result of this section:

Theorem 3.3 (Axiomatic Full Completeness) *Let $\mathcal{M}$ be an adjoint hyperdoctrine. If $\mathcal{M}$ satisfies Axioms 1–6, then $\mathcal{M}$ is fully and faithfully complete at ML-types.*

Proof. Let $\forall \vec{X}.T = \forall \vec{X}.T_1 \to \ldots \to T_n \to X_k$ be a closed ML-type, and let $f \in \mathrm{Hom}_{\mathbf{L}(1)}(!I, [\![\vdash \forall \vec{X}.T]\!])$. Using the Decomposition Theorem, one can easily prove, by induction on the measure provided by Axiom 6 that there exists $\vec{X}; \vec{x} : \vec{T} \vdash x_i M_1 \ldots M_{q_i} : X_k$ such that $\overline{\Lambda(f)} = [\![\vec{X}; \vec{x} : \vec{T} \vdash x_i M_1 \ldots M_{q_i} : X_k]\!]$, where $^-$ is the bijection given by the adjiunction between $\forall$ and π^* in Definition 1.9. Then $f = [\![\vdash \Lambda \vec{X}.\lambda \vec{x} : \vec{T}.x_i M_1 \ldots M_{q_i} : \forall \vec{X}.\vec{T} \to X_k]\!]$. $\Box$

As we remarked earlier, the Strong Decomposition Theorem 3.2 is stronger than the Decomposition Theorem 3.1 in two respects. First of all, it provides a decomposition for *all* morphisms of a certain shape, and not just for those whose domains and codomains are *denotations* of types. Secondly, it guarantees the *uniqueness* of the decomposition. Correspondingly, the axioms could be weakened either by considering only spaces of morphisms whose domains and codomains are *denotations* of types, instead of generic objects of appropriate "top-level" shape, or by substituting the isomorphisms requirements by weaker conditions, which only ensure the existence of *a* decomposition. More precisely, the first kind of restriction is obtained by taking, e.g. in the *Linearization of Head Occurrence*, $\vec{h}$ to be $\otimes_{i=1}^{n} ![\![T_i]\!]$, where each T_i is a simple type with free variables in $\vec{X}$. In order to guarantee the existence of *a* decomposition, it is sufficient to ask that the canonical morphisms in Axioms 2 and 4 and the morphism $\boldsymbol{\lambda} f \in \mathrm{Hom}(!h, l).(f)^\dagger$ in Axiom 5 are *surjective* maps. By weakening the axioms in either or both of these ways, we still get a set of sufficient conditions for full completeness. Notice that, if we take the weaker form of the axioms which imply only the existence of a decomposition, we do indeed need Statman's result (Theorem 1.1) to conclude faithfulness. The use of the Typical Ambiguity Theorem, on the other hand, is not necessary, when strong decomposition is available.

As we will see in Section 3.2, in the concrete model of PERs over the LCA of partial involutions, we prove weak variants of Linearization of Head Occurrence and Linear Function Extensionality Axioms, and strong forms for the remaining axioms for Decomposition.

Moreover, notice that the Finiteness Axiom also has a weak form, obtained by requiring the existence of a size function only for morphisms whose domains and codomains are denotations of appropriate types. This is actually the version of the axiom which we prove for our PER model in Section 3.2.

Finally, notice that all the Axioms 1–6 in the strong form are *consistent*, since they are satisfied in the underlying category of the adjoint hyperdoctrine induced by the linear term model. Moreover, Axiom 1 is trivially *necessary*. The question of the necessity of the Axioms 2–6 in their weak or strong form remains open.

3.2 A Fully Complete PER Model

In this section, we prove that the PER category over the LCA $\mathcal{A}_{\mathtt{PInv}}$ of Section 2.2 satisfies the Axioms 1–5 of Section 3.1 (some of them in a weak form), and hence it gives rise to a model which satsfies the Decomposition Theorem. The proof of the Finiteness Axiom 6, which is the most difficult part of the proof of full completeness for the model $\mathrm{PER}_{\mathcal{A}_{\mathtt{PInv}}}$, makes use of the Typed Separability result presented in Section 1.2, and it requires the study of an intermediate model for $\lambda_{\perp,\top}$, which amounts to our second case study. This proof appears in Section 4.3.4.

By definition of PER adjoint models (see Theorem 2.2 of Section 2.1), for all morphisms h, l in the fibre category $\mathbf{L}(\vec{\mathcal{U}})$,

$$Hom_{\mathbf{L}(\vec{\mathcal{U}})}(h, l) = F(\textstyle\bigcap_{\vec{X}} h(\vec{X}) \multimap l(\vec{X})) \ ,$$

where $F : PER_{\mathcal{A}} \to Set$ is the forgetful functor. Therefore, in order to verify the main axioms for the Decomposition Theorem, we are left to establish some isomorphisms between the images in *Set* of suitable closed polymorphic PERs. First of all, notice that Axiom 1 and the *Uniformity of Threads* Axiom hold immediately on PER models. In fact, for the first axiom to hold, we need only to verify that the PER $\bigcap_{\vec{X}} X_k$ is the empty PER. This follows immediately, by instantiating X_k with the empty per. Uniformity of Threads Axiom follows from the isomorphism $\bigcap_{\vec{X}} !\,\mathcal{R} \multimap !S \simeq \bigcap_{\vec{X}} !\,\mathcal{R} \multimap S$, which is an immediate consequence of Lemma 2.2 of Section 2.1.

The rest of this section is devoted to the proof of the validity of the Axioms 2–4. The proof of the validity of Axioms 2–4 is based essentially on the nature of partial involutions, and it requires a careful analysis of their applicative behaviour.

With the following three technical lemmata, we carry out the analysis of the structure of the partial involutions which inhabit the PERs involved in Axioms 2–4. In particular, in Lemma 3.1, we show that the partial involutions in $dom(\bigcap_{\vec{X}} \otimes_{i=1}^{n} !\, \mathcal{R}_i \multimap X_k)$, where $\forall i.\ \mathcal{R}_i{=}\mathcal{S}_i \multimap X_i$, are "total", in the sense that, for any possible sequence of input values, they always "look" at them, before producing an output, and they are different from the empty partial involution. In Lemma 3.2, we show that any of these partial involutions always "asks" first for the same argument, say the i-th argument, for any possible sequence of input values. This allows us to isolate the *first* use of a copy in $!\ \mathcal{R}_i$. Finally, Lemma 3.3 will be used in order to define the space of total morphisms appearing in Axioms 2–4. This space amounts to a PER of *total* partial involutions (see Definition 3.2 below).

Lemma 3.1 *Let $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ be a closed PER, where $\vec{\mathcal{R}} = \otimes_{i=1}^{n}\ !R_i$, and, for all $i = 1, \ldots, n$, $\mathcal{R}_i{=}\ S_i \multimap X_i$. Let $f \in dom(\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k)$. Then f is* total*, i.e.*

$$\forall m \exists m'.\ f^*(r,m) = (l,m')\ ,$$

where $f^* = t; f; t^{-1} : \boldsymbol{N} + \boldsymbol{N} \rightharpoonup \boldsymbol{N} + \boldsymbol{N}$.

Proof. By contradiction. Assume that $\exists m.\ f^*(r,m)\uparrow$. Then we reach a contradiction by instantiating $\vec{X}$ as follows: $X_k = \{h : \mathbf{N} \rightharpoonup_{\texttt{Inv}} \mathbf{N} \mid h(m)\downarrow\}$, and X_j, for all $j \neq k$, by the PER with only one equivalence class and with domain containing all partial involutions. In fact: $\forall \vec{g} \in \mathrm{dom}(\vec{\mathcal{R}}.\ f\vec{g}(m)\uparrow$, i.e. $f\vec{g} \notin X_k$. Hence $f \notin \mathrm{dom}(\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k)$. In order to conclude, we are left only to check that $\exists \vec{g} \in \mathrm{dom}(\vec{\mathcal{R}})$. I.e., we have to check that $\forall i. \exists g_i \in \mathrm{dom}(\mathcal{R}_i)$. Such g_i's exist, since each g_i can be taken to be the function constantly equal to an element in X_i, i.e., let $h \in X_i$, we define, for all n,m, $t; g; t^{-1}(r,n) = (r,m)$ if and only if $h(n) = m$. Similarly, we can rule out the case $\exists m, m'.\ f^*(r,m) = (r,m')$, by instantiating X_k to the one-equivalence class PER $\{h : \mathbf{N} \rightharpoonup_{\texttt{Inv}} \mathbf{N} \mid h(n)\uparrow\}$. $\square$

The following technical definition will be useful in the sequel. It allows to analyze the behaviour of a partial involution f, when it is applied to a vector $\vec{g}$ of n arguments.

Definition 3.1 *Let*

- $f : \boldsymbol{N} \rightharpoonup_{\texttt{Inv}} \boldsymbol{N}$,
- $D_n = \underbrace{(\boldsymbol{N} \times (\boldsymbol{N} + \boldsymbol{N}) + \ldots + \boldsymbol{N} \times (\boldsymbol{N} + \boldsymbol{N}))}_{n} + \boldsymbol{N}$.

We define $f_n^* : D_n \rightharpoonup_{\texttt{Inv}} D_n$ to be be f "up-to coding", i.e.: $f_n^* =$
$\underbrace{id_{\boldsymbol{N}} \times t + \ldots + id_{\boldsymbol{N}} \times t}_{n} + id_{\boldsymbol{N}}; \underbrace{p + \ldots + p}_{n} + id_{\boldsymbol{N}}; t_{n-1}; f; t_{n-1}^{-1}; \underbrace{p^{-1} + \ldots + p^{-1}}_{n} +$
$id_{\boldsymbol{N}}; \underbrace{id_{\boldsymbol{N}} \times t^{-1} + \ldots + id_{\boldsymbol{N}} \times t^{-1}}_{n} + id_{\boldsymbol{N}}\ ,$

where t_n is defined by induction on n as follows:

$t_0 = t : \boldsymbol{N} + \boldsymbol{N} \rightharpoonup \boldsymbol{N}$
$t_{n+1} = [t_n, id_{\boldsymbol{N}}] : \underbrace{(\boldsymbol{N} + \boldsymbol{N}) + \ldots + \boldsymbol{N}}_{n+3} \rightharpoonup \boldsymbol{N}.$

Lemma 3.2 *Let $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ be a closed PER, where $\vec{\mathcal{R}} = \otimes_{i=1}^{n}\ \boldsymbol{!}R_i$, and, for all $i = 1, \ldots, n$, $\mathcal{R}_i = \mathcal{S}_i \multimap X_i$. Let $f \in dom(\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k)$. Then there exists a unique $\mathcal{R}_i = \mathcal{S}_i \multimap X_i$, $1 \leq i \leq n$, such that:*

- $X_i = X_k$,
- $\forall m.\ f_n^*(r, m) = (l, (i, !(r, m)))$,
  where f_n^* is defined as in Definition 3.1, and $!(r, m)$ denotes any element of $\boldsymbol{N} \times (\boldsymbol{N} + \boldsymbol{N})$, whose second projection is (r, m).

Proof. By Lemma 3.1, if $f \in \mathrm{dom}(\bigcap_{\vec{X}} \vec{R} \multimap X_k)$, then $\forall m.\ \exists a.\ f_n^*(r, m) = (l, a)$. We prove first, by contradiction, that $\forall m. \exists i.\ f_n^*(r, m) = (l, (i, !(r, m)))$. Assume that $\exists i. \exists a'.\ f_n^*(r, m) = (l, (i, !(l, a')))$. We instantiate each X_j in $\vec{X}$ by $\{f : \mathbf{N} \rightharpoonup_{\mathtt{Inv}} \mathbf{N} \mid f(m) = m\}$. Then the partial involution g_j such that $g_j^*(r, m) = (r, m)$, where $g_j^* = t; g_j; t^{-1}$, is in $\mathcal{R}_j$, for all j. In particular, we take g_i such that $g_i^*(l, a')\uparrow$. Then $f \bullet \vec{g} \notin X_k$. Hence we reach a contradiction. Using a similar argument, we rule out the case $f_n^*(r, m) = (l, (i, !(r, m')))$, for $m \neq m'$. Moreover, if $f_n^*(r, m) = (l, (i, !(r, m)))$, then $X_i = X_k$. Because, if $X_i \neq X_k$, then we can instantiate $X_{k'}$, for $k' \neq k$, by the PER with only one equivalence class and with domain containing all partial involutions, and X_k by $\{h : \mathbf{N} \rightharpoonup_{\mathtt{Inv}} \mathbf{N} \mid h(m)\downarrow\}$. But, for $g_i = \emptyset$, this yields $f \bullet \vec{g}(m)\uparrow$, i.e. $f \bullet \vec{g} \notin X_k$. Therefore, we are left to show that $\exists ! i. \forall m.\ f^*(r, m) = (l, (i, (r, m)))$. We prove it by contradiction. Assume that $\exists m, m'$, $\exists i, j$ such that $f_n^*(r, m) = (l, (i, !(r, m)))$ and $f_n^*(r, m') = (l, (j, !(r, m')))$. First of all notice that, by the argument above, $X_i = X_j = X_k$. Then let $X_k = \{f : \mathbf{N} \rightharpoonup_{\mathtt{Inv}} \mathbf{N} \mid f(m) \updownarrow f(m')\}$, and $X_{k'} = \{f \mid f \in [\mathbf{N} \rightharpoonup_{\mathtt{Inv}} \mathbf{N}]\}$, for $k' \neq k$. Let g_i be such that $g_i^*(r, m) = (r, m)$, $g_i^*(r, m') = (r, m')$, then $g_i \in T_i$, and let $g_j = \emptyset \in T_j$. Then, for any $g_l \in T_l$, for $l \neq i, j$, $f\vec{g}(m) = m$, while $f\vec{g}(m')\uparrow$. $\square$

Lemma 3.3 *Let $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ be a PER, where $\vec{\mathcal{R}} = \otimes_{i=1}^{n}\ \boldsymbol{!}R_i$, and, for all $i = 1, \ldots, n$, $\mathcal{R}_i = S_i \multimap X_i$. Let $f, f' \in dom(\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k)$. If $f(\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k)f'$, then*

$$\forall m.\ f_n^*(r, m) = f_n'^*(r, m)\ ,$$

where $f_n^*, f_n'^*$ are defined as in Definition 3.1.

Proof. We proceed by contradiction. Assume that $\forall m.\ f_n^*(r, m) = (l, (i, !(r, m)))$ and $\forall m.\ f_n'^*(r, m) = !(l, (j, !(r, m)))$, for $i \neq j$. Then taking X_k to be the PER with the two equivalence classes $\{h : \mathbf{N} \rightharpoonup_{\mathtt{Inv}} \mathbf{N} \mid h(m)\uparrow\}$ and $\{h : \mathbf{N} \rightharpoonup_{\mathtt{Inv}} \mathbf{N} \mid h(m)\downarrow\}$, and taking $g_i = \emptyset$, and g_j such that $g_j^*(r, m) = (r, m)$, where $g_j^* = t; g_j; t^{-1}$, we get $(f\vec{g}, f'\vec{g}) \notin X_k$. Contradiction. $\square$

Now we introduce the *total* space of morphisms appearing in Axioms 2–4. This is induced by a suitable *subPER* of the PER $\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k)$, which is meant to contain only the equivalence classes of total maps. By *subPER* we intend a PER whose equivalence classes form a subset of the set of equivalence classes of the original PER. Notice that, in general, in PER categories, there is no natural notion of strict/total map, since there are no natural $\perp$-elements in any PER. But, in the special case of our combinatory algebra, there is a natural candidate for $\perp$, i.e. the equivalence class of the empty partial involution. Of course, this makes sense only if we restrict ourselves to PERs to which $\emptyset$ belongs. Then strict maps turn out to be those maps which indeed "look" at their arguments, and total maps can be defined, as usual, as strict maps different from $\perp$. Bearing on this intuition, we can define the space of total polymorphic maps used in Axiom 2 as follows (by Lemma 3.3 we are guaranteed that the following definition yields a subPER, i.e. it identifies a subset of the set of the equivalence classes of the original PER):

Definition 3.2 *Let* $\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k)$ *be a closed PER.*

- *We define the* total *PER* $(\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k))_t$ *to be the subPER of* $\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k)$*, which contains only* total *partial involutions, i.e.:*

  $f \in dom((\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k))_t)$ *iff*
  $f \in dom(\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k)) \ \wedge \ \forall m.\ f^*(r,(r,m)) = (l,(r,m))$,

  where $f^* = t + t; t; f; t^{-1}; t^{-1} + t^{-1}$.
- *We define the space of* total *morphisms* $Hom^t_{\boldsymbol{L}(\vec{\mathcal{U}})}((\Lambda\vec{X}.\, \mathcal{S}) \multimap \pi_i, (\Lambda\vec{X}.\, \mathcal{R}) \multimap \pi_k)$ *to be the set*

$$F((\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k))_t) \ .$$

We start by proving the validity of Axioms 3.

Theorem 3.4 (Type Coherence) *Let* $\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k)$ *be a closed PER such that* $X_i \neq X_k$*. Then*

$$(\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k))_t = \emptyset \ .$$

Proof. Assume by contradiction $f \in \mathrm{dom}(\bigcap_{\vec{X}}(\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k))_t$. Then, since f is total, $\forall m. \exists a.\ f^*(r,(r,m)) = (l,a)$, where $f^* : (\mathbf{N}+\mathbf{N}) + (\mathbf{N}+\mathbf{N}) \to (\mathbf{N}+\mathbf{N}) + (\mathbf{N}+\mathbf{N})$ is $t+t; t; t^{-1}; t^{-1} + t^{-1}$. First of all, by suitably instantiating the X_j's (by mimicking part of the proof of Lemma 3.2), one one can check that $f^*(r,(r,m)) = (l,(r,m))$. But then, instantiating X_k by $\{h : \mathbf{N} \rightharpoonup_{\mathtt{Inv}} \mathbf{N} \mid h(m)\downarrow\}$, and X_j by the one equivalence class PER containing all partial involutions, for all $j \neq k$, we get: $\emptyset \in \vec{\mathcal{S}} \multimap X_i$, but $f \bullet \emptyset = \emptyset \notin X_k$. $\square$

Now we focus on Axiom 2, i.e.:

$$\text{case}_i\{F(\sigma_i)\}_{i=1,\ldots,n} : \coprod_{i=1}^{n} F((\bigcap_{\vec{X}} \mathcal{R}_i \multimap (\vec{\mathcal{R}} \multimap X_k))_t) \simeq F(\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k) ,$$

where σ_i is the appropriate canonical morphism (see Axiom 2 of Section 3.1.1).

Using Lemmata 3.2 and 3.3, one can show that the function $\text{case}_i\{F(\sigma_i)\}_{i=1,\ldots,n}$ is surjective. I.e.:

Theorem 3.5 (Weak Linearization of Head Occurrence) *Let $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ be a closed PER. Then the following morphism is surjective:*

$$case_i\{F(\sigma_i)\}_{i=1,\ldots,n} : \coprod_{i=1}^{n} F((\bigcap_{\vec{X}} \mathcal{R}_i \multimap (\vec{\mathcal{R}} \multimap X_k))_t) \longrightarrow F(\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k) ,$$

where $\sigma_i = \Lambda \vec{X}.\Lambda^{-1}; \Lambda \vec{X}.\pi_i; \tau \multimap id_{X_k}; \Lambda \vec{X}.con_{\vec{\mathcal{R}}} \multimap id_{X_k}$.

Proving injectivity of the above morphism is problematic. In fact, this amounts to showing that, if it is not the case that $f(\bigcap_{\vec{X}} \mathcal{R}_i \multimap (\vec{\mathcal{R}} \multimap X_k))_t f'$, then it is not the case that $\sigma_i(f)(\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k)\sigma_i(f')$. As already remarked at the end of Section 3.1.1, the surjectivity of the function $\text{case}_i\{F(\sigma_i)\}_{i=1,\ldots,n}$ is at any rate sufficient to guarantee that the relevant morphisms have *a* decomposition, and therefore, if the finiteness condition also holds, we have full completeness. The question remains whether the strong version of the *Linearization of Head Occurrence* Axiom holds. What we can say is that we can prove *a posteriori* that the isomorphism holds in the case in which we restrict ourselves to universal PERs denoting ML-types. Namely, using the fact that the weak Decomposition Theorem and the Finiteness Axiom hold in our model (the latter is proved in Section 4.3.4), we can infer that our model is fully-complete, i.e. all morphisms from type interpretations to type interpretations are λ-definable. But then, by Statman Theorem, since the model is non-trivial, any relevant morphism denotes exactly one $\beta\eta$-normal form, and therefore the isomorphism holds in Theorem 3.5, when the PER $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ denotes an ML-type.

A similar analysis can be done for the Linearization of Head Occurrence Axiom. Using Lemmata 3.2 and 3.3, one can easily prove that the canonical morphism in the Axiom is surjective (Theorem 3.6 below). Moreover, *a posteriori*, we can prove that the isomorphism holds in case we are dealing with denotations of ML-types.

Theorem 3.6 (Weak Linear Function Extensionality)
Let $\bigcap_{\vec{X}} (\mathcal{S} \multimap X_k) \multimap (\mathcal{R} \multimap X_k)$ be a closed PER. Then the following morphism is surjective:

$$\Lambda \vec{X}.(\cdot)\vec{X} \multimap id_{X_k} \; : \; \bigcap_{\vec{X}} \mathcal{R} \multimap \mathcal{S} \longrightarrow (\bigcap_{\vec{X}} (\mathcal{S} \multimap X_k) \multimap (\mathcal{R} \multimap X_k))_t .$$

4 Maximal Theory for the Simply Typed Lambda Calculus with $\bot, \top$

In this section, we define a model for $\lambda_{\bot,\top}$ in a suitable subcategory of the co-Kleisli category of $\mathrm{PER}_{\mathcal{A}_{\mathsf{PInv}}}$. We prove that this model is fully complete and minimal, i.e. it realizes the maximal theory on $\lambda_{\bot,\top}$. Actually, it is fully complete and minimal also for an *infinitary* version of $\lambda_{\bot,\top}$. Then, we present some interesting applications of our full completeness result, including the proof of the Finiteness Axiom for the model of ML-types of Section 3.

We start by introducing the *Sierpinski PER* $\mathcal{O}$ on $\mathcal{A}_{\mathsf{PInv}}$. Our model is defined in the co-Kleisli category of the affine category freely generated by $\mathcal{O}$.

Definition 4.1 (The PER $\mathcal{O}$) *Fix $* \in \boldsymbol{N}$. Let $\mathcal{O}$ be the PER on the combinatory algebra $\mathcal{A}_{\mathsf{PInv}}$ consisting of two equivalence classes defined as follows:*

- $\bot = \{f : \boldsymbol{N} \rightharpoonup_{\mathsf{Inv}} \boldsymbol{N} \mid f(*)\uparrow\}$
- $\top = \{f : \boldsymbol{N} \rightharpoonup_{\mathsf{Inv}} \boldsymbol{N} \mid f(*) = *\}$.

Definition 4.2 ($\mathcal{M}_{\mathcal{O}}$) *Let $\mathcal{M}_{\mathcal{O}} = (CCPER_{\mathcal{O}}, \bullet_{\mathcal{O}}, [\![\ ]\!]^{\mathcal{O}})$ be the model of $\lambda_{\bot,\top}$ defined in the co-Kleisli category of the affine category freely generated by $\mathcal{O}$, where:*

$$[\![\bot]\!]^{\mathcal{O}} = \bot \quad \text{and} \quad [\![\top]\!]^{\mathcal{O}} = \top \ .$$

First of all notice that, by the fact that the PER $\mathcal{O}$ has only a finite number of equivalence classes and by extensionality of the PER model, each PER in $\mathrm{CCPER}_{\mathcal{O}}$ has only *finitely many* equivalence classes. Therefore, the model is trivially *not* faithful w.r.t. the $\beta\eta$-theory. As we will see, the fact that it realizes exactly the maximal theory follows as a consequence of the full completeness result.

The proof of full completeness follows the standard general pattern based on a Decomposition Theorem. But, in this case, the Decomposition Theorem holds for the *partial involutions* in the domains of the PERs interpreting a simple type (and not for the equivalence classes, which can identify different typed Böhm trees):

Theorem 4.1 (Decomposition) *Let $T = T_1 \to \ldots \to T_n \to o \in SimType$, $n \geq 0$, where, for all $i = 1, \ldots, n$, $T_i = U_{i1} \to \ldots \to U_{iq_i} \to o$. If $f \in dom(\otimes_{i=1}^{n} ![\![T_i]\!]^{\mathcal{O}} \multimap \mathcal{O})$, then*

- *either $f \in dom([\![\vec{x} : \vec{T} \vdash c_i : o]\!]^{\mathcal{O}})$, for some $c_i \in \{\bot, \top\}$*
- *or $\exists i \in \{1, \ldots, n\}$, and $\exists g_1, \ldots, g_{q_i}$, where*

$\forall j \in \{1,\ldots,q_i\}$. $g_j \in dom(\otimes_{i=1}^n ![\![T_i]\!]^{\mathcal{O}} \multimap [\![U_{ij}]\!]^{\mathcal{O}})$, *such that*

$$f\ (\times_{i=1}^n ![\![T_i]\!]^{\mathcal{O}} \multimap \mathcal{O})\ (con_{\otimes_{i=1}^n ![\![T_i]\!]^{\mathcal{O}}}; (\pi_i^n \otimes \langle g_1,\ldots,g_{q_i}\rangle^\dagger); Ap)\ ,$$

where, by abuse of notation, we denote representatives of equivalence classes of the canonical morphisms in the affine category freely generated by $\mathcal{O}$ by the canonical morphisms themselves.

The proof of Theorem 4.1 is carried out in Section 4.1. As usual, once we have a Decomposition Theorem, in order to get the full completeness result, we are still left to rule out partial involutions generating trees whose height is infinite, which would correspond to infinite typed Böhm trees. To this aim, we prove the following:

Proposition 4.1 (Finite Representative Property) *Let $\mathcal{R}$ be a simple PER over $\mathcal{O}$. Then, for each equivalence class of $\mathcal{R}$, there exists a* finite *representative, i.e. a partial involution which generates, via iterative applications of the Decomposition Theorem, a* finite *tree.*

The proof of Proposition 4.1 above appears in Section 4.2. This concludes the proof of full completeness.

Finally, by the full completeness result and by the definition of the PER model $\mathcal{M}_{\mathcal{O}}$, which is well-pointed, the following proposition follows immediately:

Proposition 4.2 (Minimality) *The model $\mathcal{M}_{\mathcal{O}}$ realizes the maximal theory on λ_o.*

We remark that we have used the full completeness result in order to show that $\mathcal{M}_{\mathcal{O}}$ is minimal.

4.1 Proof of the Decomposition Theorem

Let $T \in SimType$, and let $\mathcal{R} = [\![T]\!]^{\mathcal{O}}$, where $\mathcal{R} = \otimes_{i=1}^n !\ \mathcal{R}_i \multimap \mathcal{O}$, and, for all $i = 1,\ldots,n$, $\mathcal{R}_i = \otimes_{j=1}^{q_i} !S_{ij} \multimap \mathcal{O}$. Let $f \in \mathrm{dom}(\otimes_{i=1}^n !\ \mathcal{R}_i \multimap \mathcal{O})$. We analyze the behaviour of f as operator in an application to arguments $!g_1 \in\ !\mathcal{R}_1,\ldots, !g_n \in !\ \mathcal{R}_n$. That is, let us apply the coding functions t, p of Section 2.2, in order to get
$\overline{f} : ((\mathbf{N}\times\mathbf{N}) + \ldots + (\mathbf{N}\times\mathbf{N})) + \mathbf{N} \rightharpoonup ((\mathbf{N}\times\mathbf{N}) + \ldots + (\mathbf{N}\times\mathbf{N})) + \mathbf{N}$,

where in the domain (codomain) of $\overline{f}$ there are n occurrences of $\mathbf{N}\times\mathbf{N}$, each one corresponding to one of the n arguments to which f applies. In the interaction with the i-th argument $!g_i$ only the i-th occurrence of $\mathbf{N}\times\mathbf{N}$ in the domain (codomain) of $\overline{f}$ is involved. In particular, the lefthand occurrence of

$\mathbf{N}$ in $\mathbf{N} \times \mathbf{N}$ refers to the copy of the argument $!g_i$ used, while the righthand occurence of $\mathbf{N}$ carries the values from (to) g_i.

We have three possibilities, according to the behaviour of $\overline{f}$ on the input $*$ (one can easily show that other input values are not relevant, by definition of the PER $\mathcal{O}$):

(1) $\overline{f}(r,*)\uparrow$. Then $f \in [\![\vec{x} : \vec{T} \vdash \perp : o]\!]^{\mathcal{O}}$.
(2) $\overline{f}(r,*) = *$. Then $f \in [\![\vec{x} : \vec{T} \vdash \top : o]\!]^{\mathcal{O}}$.
(3) $n > 0$ and $\overline{f}$ "interrogates" one of its arguments, say the i-th (in the j-th copy), i.e. $\overline{f}(r,*) = (l,(i,(j,m)))$, where m is the value passed to g_i. Notice that m must be equal to $(r,*)$, otherwise one can easily show that $f \notin \mathrm{dom}(\mathcal{R})$.

In the first two cases we are done. We now concentrate on the third case. By the observations above, we have:

Lemma 4.1 (Linearization of Head Occurrence) *Let* $\mathcal{R} = \otimes_{i=1}^{n} ! \,\mathcal{R}_i \multimap \mathcal{O}$, *where, for all* $i = 1, \ldots, n$, $\mathcal{R}_i = \otimes_{j=1}^{q_i} !S_{ij} \multimap \mathcal{O}$. *For all* $f \in$ *dom*$(\vec{\mathcal{R}} \multimap \mathcal{O})$ *satisfying 3, where* $\vec{\mathcal{R}}$ *is an abbreviation for* $\otimes_{i=1}^{n} ! \,\mathcal{R}_i$, *there exist* $i \in \{1,\ldots,n\}$ *and* $f' \in$ *dom*$(R_i \multimap \vec{\mathcal{R}} \multimap \mathcal{O})$ strict, *i.e.* $t + t; t; f'; t^{-1}; t^{-1} + t^{-1}(r,(r,*)) = (l,(r,*))$, *such that*

$$f \;\; \mathcal{R} \;\; (con_{\vec{\mathcal{R}}}; \pi_i^n \otimes id_{\vec{\mathcal{R}}}; \Lambda^{-1}(f')) \ .$$

□

Now we examine the structure of f'. More in general, one can show Lemma 4.2 below:

Lemma 4.2 (Linear Function Extensionality) *Let* $\mathcal{S}, \mathcal{R}$ *be PERs. Then, for all* $f \in$ *dom*$((\mathcal{S} \multimap \mathcal{O}) \multimap (\mathcal{R} \multimap \mathcal{O}))$ *strict, there exists* $f' \in$ *dom*$(\mathcal{R} \multimap \mathcal{S})$ *such that*

$$f \;\; ((\mathcal{S} \multimap \mathcal{O}) \multimap (\mathcal{R} \multimap \mathcal{O})) \;\; (\Lambda((id_{\mathcal{S} \multimap \mathcal{O}} \otimes f'); Ap)) \ .$$

The last technical lemma that we need in order to prove the Decomposition Theorem is *Uniformity of Threads*, which amounts to Lemma 2.2(2).

Finally, we have:

Proof of the Decomposition Theorem 4.1. If either case 1 or case 2 above applies, then we are done. If case 3 applies, then, by Lemma 4.1, there exists $i \in \{1,\ldots,n\}$ and $f' \in \mathrm{dom}(\mathcal{R}_i \multimap (\vec{\mathcal{R}} \multimap \mathcal{O}))$ such that f $(\vec{\mathcal{R}} \multimap \mathcal{O}$ $)$ $(\mathrm{con}_{\vec{\mathcal{R}}}; \pi_i^n \otimes \mathrm{id}_{\vec{\mathcal{R}}}; \Lambda^{-1}(f'))$. By Lemma 4.2, there exists g such that f' $((\vec{\mathcal{S}}_i \multimap \mathcal{O}$

$)$—$\circ(\vec{\mathcal{R}}$—$\circ\ \mathcal{O}))\ \Lambda((\mathrm{id}_{\vec{\mathcal{S}}_i \multimap \mathcal{O}} \otimes g); \mathrm{Ap})$. Then $f\ (\vec{\mathcal{R}} \multimap \mathcal{O})\ \mathrm{con}_{\vec{\mathcal{R}}}; \pi_i^n \otimes g; \mathrm{Ap}$. Finally, by Proposition 2.2(2), by definition of the product of PERs and by the universality property of the product, we obtain $g\ (\vec{\mathcal{R}} \multimap \vec{\mathcal{S}}_i)\ \langle g_1, \ldots, g_{q_i} \rangle^{\dagger}$, for some $g_1 \in \mathcal{S}_{i1}, \ldots, g_{q_i} \in \mathcal{S}_{iq_i}$. □

4.2 Proof of the Finitary Representative Property

The proof of Proposition 4.1 uses an Approximation Theorem for partial involutions in the domains of PERs interpreting simple types. Approximants of partial involutions are defined using the Decomposition Theorem. By repeatedly applying the Decomposition Theorem to a partial involution f, we obtain a (possibly) infinite typed Böhm tree. The k-th approximant of f is a partial involution obtained by truncating this tree at level k, and by substituting the *empty partial involution* for each possibly erased subtree. This is justified by the fact that the empty partial involution lives in the interpretation of $\lambda \vec{x} : \vec{T}.\ \bot : \vec{T} \to o$ for any type $\vec{T} \to o$. Formally:

Definition 4.3 (Approximants) *Let $f \in dom(\mathcal{R})$, where $\mathcal{R} = [\![T]\!]^{\mathcal{O}}$, for some simple type T.*

- *We define the* k-th tree, *$t_k(f)$, of height at most k, generated from f after iterated applications of the Decomposition Theorem by induction on k as follows:*
  - · *$t_0(f)$ is the tree of height 0 with only a root labeled by f;*
  - · *given the tree $t_k(f)$ of height at most k, the tree $t_{k+1}(f)$ is obtained from the tree $t_k(f)$ by expanding the possible leaves at level k via the Decomposition Theorem.*
- *We define the* k-th approximant *of f, $p_k(f)$, as the partial involution obtained from the tree $t_k(f)$ by substituting any partial involution at level k by the* empty *partial involution.*

We have the following lemma:

Lemma 4.3 *Let $T_1 \to \ldots \to T_n \to o \in SimType$, let $f \in dom(\otimes_{i=1}^n ![\![T_i]\!]^{\mathcal{O}} \multimap \mathcal{O})$. Then, for all $k \geq 0$,*
i) $p_k(f) \in dom((\otimes_{i=1}^n ![\![T_i]\!]^{\mathcal{O}} \multimap \mathcal{O})$.
ii) $p_k(f) \subseteq p_{k+1}(f)$.

Proof. The proof of i) follows from the fact that the empty partial involution belongs to any simple PER. The proof of ii) follows from monotonicity of $\bullet$. □

Theorem 4.2 (Approximation) *Let $T_1 \to \ldots \to T_n \to o$ be a simple type,* *and let* $f \in dom(\otimes_{i=1}^{n} ![\![T_i]\!]^{\mathcal{O}} \multimap \mathcal{O})$. *Then*

$$f \ (\otimes_{i=1}^{n} ![\![T_i]\!]^{\mathcal{O}} \multimap \mathcal{O}) \ (\bigcup_{k\in\omega} p_k(f)) \ .$$

Proof.(Sketch) We have to show that $\forall \vec{g} \in \otimes_{i=1}^{n} ![\![T_i]\!]^{\mathcal{O}}.\ f \bullet \vec{g} \ \mathcal{O} \ (\bigcup_{k\in\omega} p_k(f)) \bullet \vec{g}$. I.e.:
i) $f \bullet \vec{g}(*)\uparrow \iff (\bigcup_{k\in\omega} p_k(f)) \bullet \vec{g}(*)\uparrow$ and
ii) $f \bullet \vec{g}(*) = * \iff (\bigcup_{k\in\omega} p_k(f)) \bullet \vec{g}(*) = *$.
The implications ($\Rightarrow$) in i) and ($\Leftarrow$) in ii) follow from monotonicity of $\bullet$ and from the fact that, for all k the graph of $p_k(f)$ is contained in the graph of f. In order to show i)($\Leftarrow$) and ii)($\Rightarrow$), one can check that, if $f \bullet \vec{g}(*) = *$ and the result is obtained with a thread of length at most $2k$, then $p_k(f) \bullet \vec{g}(*) = *$. □

Proposition 4.1 follows from the following crucial result:

Theorem 4.3 *Let* $T_1 \to \ldots \to T_n \to o \in SimType$, $f \in dom(\otimes_{i=1}^{n} ![\![T_i]\!]^{\mathcal{O}} \multimap \mathcal{O})$. *Then there exists* $K \geq 0$ *such that*

$$f \ \mathcal{R} \ p_K(f) \ .$$

Proof. Since $\mathcal{R}$ has only finitely many equivalence classes, by Lemma 4.3, there exists $K \geq 0$ such that, for all $m_1, m_2 \geq K$, $p_{m_1}(f) \ \mathcal{R} \ p_{m_2}(f)$. Hence, by Theorem 4.2, we have that $f \ \mathcal{R} \ p_K(f)$. □

4.3 Some Applications of the Full Completeness Result

Our fully complete model provides immediate *semantical* proofs of some interesting facts concerning the maximal theory $\approx$ on $\lambda_{\perp,\top}$ and an infinitary version of $\lambda_{\perp,\top}$, and it allows to conclude the proof of full completeness for the model for ML-types of Section 3.

4.3.1 Context Lemma

Definition 4.4 (Applicative Equivalence) *Let* $\approx^{\mathsf{app}} \subseteq \Lambda_o^0 \times \Lambda_o^0$ *be defined by*

$$M \approx^{\mathsf{app}} N \ \Leftrightarrow \ \forall P_1, \ldots, P_n \in \Lambda^0.\ MP_1 \ldots P_n =_\beta NP_1 \ldots P_n : o \ .$$

Using our fully complete model, we can immediately (re)prove the following:

Lemma 4.4 (Context Lemma) *The theory* $\approx$ *admits an* applicative *char-*

acterization, i.e.

$$M \approx N \iff M \approx^{\mathsf{app}} N \ .$$

4.3.2 Infinite Typed Böhm Trees

Our model is actually a fully complete model for the maximal theory on the infinitary $\lambda_{\perp,\top}$, i.e. the simply typed, possibly infinite, Böhm trees:

Definition 4.5 (Infinitary Typed Böhm Trees) *We define the* infinitary *typed Böhm trees as the trees obtained as supremums of typed Böhm trees corresponding to approximants.*

By Theorem 4.3, we have that:

Proposition 4.3 *$\mathcal{M}_{\mathcal{O}}$ is a fully complete minimal model for the infinitary typed Böhm trees.*

4.3.2.1 The Theory on Infinitary Böhm Trees is Conservative.

The theory over infinitary typed Böhm trees $\approx^{\infty}$ is a conservative extension of $\approx$ w.r.t. terms in $\lambda_{\perp,\top}$, i.e.:

Proposition 4.4

$$\approx^{\infty}_{|\mathtt{TBT}} = \approx_{|\mathtt{TBT}} \ ,$$

where $\approx^{\infty}_{|\mathtt{TBT}}$, $\approx_{|\mathtt{TBT}}$ denote the theory $\approx^{\infty}$ and the theory $\approx$ restricted to the finite typed Böhm trees, respectively.

4.3.3 Decidability Results

It is well-known that the theory $\approx$ is decidable, [32,31]. Using our fully complete model, we can give a *semantical* proof of the following decidability results:

Theorem 4.4 *Let $\mathcal{R}$ be a simple PER, i.e. $\mathcal{R}{=}\mathcal{R}_1{\to} \ \dots \ {\to}\mathcal{R}_n{\to}\mathcal{O}$. For all $f : Nat \to Nat$ whose graph is finite, it is decidable whether $f \in dom(\mathcal{R})$.*

Proof. In order to decide whether $f \in \mathrm{dom}(\mathcal{R})$, it is sufficient to check the behaviour of f when applied to the "relevant" $\vec{g} \in \vec{\mathcal{R}}$, i.e. to the g's whose domains (and codomains), roughly, are contained in a suitable subset of the domains of h, h'. More precisely, let
$\overline{h}, \overline{h}' : ((\mathbf{N} \times \mathbf{N}) + \ldots + (\mathbf{N} \times \mathbf{N})) + \mathbf{N} \rightharpoonup ((\mathbf{N} \times \mathbf{N}) + \ldots + (\mathbf{N} \times \mathbf{N})) + \mathbf{N}$
be the partial involutions obtained from h, h' using the coding functions t, p. Then g_i is "relevant" to h, h' if $\mathrm{dom}(g_i) \subseteq \{n \mid \exists k.\ (l, (i, (k, n))) \in \mathrm{dom}(\overline{h}) \cup \mathrm{dom}(\overline{h}')\}$. These $\vec{g}$'s are the only "relevant" ones for h, h' in the sense that, for any other "non-relevant" $\vec{g}$, there exists $\vec{g}'$ "relevant" such that $h \bullet \vec{g}(*) \simeq$

$h \bullet \vec{g}'(*)$ and $h' \bullet \vec{g}(*) \simeq h' \bullet \vec{g}'(*)$. Since h, h' have finite graphs, then there are only finitely many $g_1, \ldots, g_n$ whose graphs are finite. We can easily generate all these "relevant" partial involutions $g_1, \ldots, g_n$. At this point, we have to eliminate the relevant $\vec{g}$'s which are not in $\mathrm{dom}(\vec{\mathcal{R}})$. Moreover, we need to know, for all relevant g_i, g_i', whether g_i is equivalent to g_i'. In order to decide this, we compute, in turn, the partial involutions which are relevant for g_i and g_i'. And, recursively, we have to compute the relevant partial involutions of the relevant partial involutions, until we reach the ground PER $\mathcal{R}$. Once we have eliminated those $\vec{g}$ which are not in $\mathrm{dom}(\vec{\mathcal{R}})$, and we have divided the set of relevant $\vec{g}$'s in equivalence classes, we can check, finally, the applicative behaviour of f. Notice that the computations $h \bullet \vec{g}(*)$ and $h' \bullet \vec{g}(*)$ always terminate, since, by definition of partial involution, and by the fact that the graphs of $h, h', g_1, \ldots, g_n$ are all finite, there cannot be an infinite (possibly cyclic) computation. Namely, the computation $h \bullet \vec{g}(*)$ either converges to $*$ or diverges because h or $\vec{g}$ are not defined on some element. This concludes the proof.

$\square$

Using a similar argument, we can prove:

Proposition 4.5 *Let $\vdash M : T$, $\vdash N : T$ be such that $[\![\vdash M : T]\!]^{\mathcal{O}}$, $[\![\vdash N : T]\!]^{\mathcal{O}}$ have representatives with finite graphs. Then it is decidable whether $M \approx N$.*

However, notice that, unfortunately, there are very few λ-terms whose interpretation is finite in the sense of Proposition 4.5 above. E.g. the identity of type $((o \to o) \to o) \to ((o \to o) \to o)$ has no representatives with finite graph. Intuitively, this depends on the fact that it's argument can ask for *any* number of copies of *it's* argument.

4.3.4 Proof of the Finiteness Axiom for the Model for ML-types

We prove a weak form of the Finiteness Axiom 6, i.e. we consider only universal PERs which are denotations of ML-types. In particular, we prove that the trees generated by elements of these PERs, via repeated applications of the Decomposition Theorem, have *finite* height. Therefore, the size function in the Finiteness Axiom can be taken directly to be the height of the tree generated via the Decompostion Theorem.

We use approximants, in order to define a measure on partial involutions in $\mathcal{M}_{\mathcal{O}}$, and hence in particular on partial involutions in the model of ML-types:

Definition 4.6 (Size Function) *Let $T = T_1 \to \ldots \to T_n \to o$ be a type of*

$\lambda_{\perp,\top}$, *and let* $f \in dom(\mathcal{R})$, *where* $\mathcal{R} = \otimes_{i=1}^{n} ! [\![T_i]\!]^{\mathcal{O}} \multimap \mathcal{O}$. *We define*

$$\mathcal{H}(f) = sup_k \ \mathcal{H}(t_k(f)) \ ,$$

where $\mathcal{H}(t_k(f))$ *is the height of the tree* $t_k(f)$.

The following lemma follows from the Approximation Theorem 4.2:

Lemma 4.5 (Approximation) *Let* $\bigcap_{\vec{X}} \vec{\mathcal{R}} \to X_k$ *be the interpretation of an ML-type, and let* $f \in dom(\bigcap_{\vec{X}} \vec{\mathcal{R}} \to X_k)$. *Then*
i)

$$f \ (\bigcap_{\vec{X}} \vec{\mathcal{R}} \to X_k) \ \bigcup_{k \in \omega} p_k(f) \ .$$

ii) For all $\vec{X}$, *for all* $g_1 \in \mathcal{R}_1, \ldots, g_n \in \mathcal{R}_n$,

$$(\bigcup_{k \in \omega} \vec{\Lambda}(p_k(f))) \bullet !g_1 \ldots \bullet !g_n = \bigcup_{k \in \omega} (\vec{\Lambda}(p_k(f)) \bullet !g_1 \ldots \bullet !g_n) \ .$$

Lemma 4.6 *Let* $f \in dom(\otimes_{i=1}^{n} ! \ \mathcal{R}_i \multimap \mathcal{O})$, *where* $\mathcal{R}_1 \to \ \ldots \ \mathcal{R}_n \to \ \mathcal{O} = [\![T_1 \to \ldots \to T_n \to o]\!]^{\mathcal{O}}$, *for some simple type* $T_1 \to \ldots \to T_n \to o$. *Then*

$$p_k(f) \in [\![\vec{x} : \vec{T} \vdash M_k : o]\!]^{\mathcal{O}} \ ,$$

where M_k *is the term of* $\lambda_{\perp,\top}$ *whose (typed) Böhm tree corresponds to the tree* $t_k(f)$.

Proof. By induction on k. $\square$

Finally, using the Typed Separability Result of Section 1.2, we have:

Theorem 4.5 (Finiteness) *Let* $f \in dom(\bigcap_{\vec{X}} \vec{\mathcal{R}} \to X_k)$, *where* $\bigcap_{\vec{X}} (\vec{\mathcal{R}} \to X_k)$ *is a closed PER denoting, up-to-uncurrying, the ML-type* $\forall \vec{X}.T_1 \to \ldots \to T_n \to X_k$. *Then* $\mathcal{H}(f) < \infty$.

Proof. We proceed by contradiction. Assume $\mathcal{H}(f) = \infty$. Then,

$$\forall Y. \ \vec{\Lambda}(f) \in dom(\vec{\mathcal{R}}[Y \to Y/\vec{X}] \to (Y \to Y)) \ ,$$

hence, by Lemma 4.5, also

$$\forall Y. \ \bigcup_{k \in \omega} \vec{\Lambda}(p_k(f)) \in dom(\vec{\mathcal{R}}[Y \to Y/\vec{X}] \to (Y \to Y)) \ .$$

Let

$$g_1 \in [\![Y; \vdash S_{\alpha_{T_1}} : \alpha_{T_1}]\!], \ldots, g_n \in [\![Y; \vdash S_{\alpha_{T_n}} : \alpha_{T_n}]\!] \ ,$$

where $S_{\alpha_{T_i}}$, for $i = 1, \ldots, n$, are the convergence tests defined in Section 1.2. Then

$$(\bigcup_{k\in\omega}\vec{\Lambda}(p_k(f)))g_1 \ldots g_n \in dom(\bigcap_Y Y \to Y).$$

By Lemma 4.5(ii), $\vec{\Lambda}(\bigcup_{k\in\omega} p_k(f))g_1 \ldots g_n = \bigcup_{k\in\omega}(\vec{\Lambda}(p_k(f))g_1 \ldots g_n)$. Now we show that $\bigcup_{k\in\omega}(\vec{\Lambda}(p_k(f))g_1 \ldots g_n) \notin dom(\bigcap_Y Y \to Y)$, thus obtaining a contradiction. By Lemma 4.6,

$$(\vec{\Lambda}(p_k(f))g_1 \ldots g_n) \in \epsilon$$

where

$$\begin{aligned}\epsilon = [\![Y; \vdash M_k : \alpha_{T_1} \to \ldots \to \alpha_{T_n} \to [Y \to Y]]\!]^{\mathcal{O}} & \bullet [\![Y; \vdash S_{\alpha_{T_1}} : \alpha_{T_1}]\!]^{\mathcal{O}} \\ & \bullet \cdots \\ & \bullet [\![Y; \vdash S_{\alpha_{T_n}} : \alpha_{T_n}]\!]^{\mathcal{O}})\end{aligned}$$

where Y plays the role of the type variable o in Lemma 4.6 and M_k is the normal form whose Böhm tree corresponds to the tree determined by $p_k(f)$. Then, since M_k contains $\perp$, (because $p_k(f)$ is a truncation of an infinite tree), by the Typed Separability Theorem 1.2,

$$\begin{aligned}(\vec{\Lambda}(p_k(f))g_1 \ldots g_n) \in \quad & [\![Y; \vdash \lambda x : Y.\, \perp : Y \to Y]\!]^{\mathcal{O}} \\ = \; & \{h : \mathbf{N} \rightharpoonup_{\mathtt{Inv}} \mathbf{N} \mid t; h; t^{-1}(r, n)\uparrow\}.\end{aligned}$$

(Note that this invocation of the Typed Separability Theorem is well-defined because of the fact that, as already observed, the empty partial involution lives in the interpretation of $\lambda\vec{x} : \vec{T}.\, \perp : \vec{T} \to o$ for any type $\vec{T} \to o$. Thus the *same* partial involution can serve as the denotation of the original term, *and* its image under the substitution σ used in the statement of the Typed Separability Theorem.) Therefore

$$\bigcup_{k\in\omega}(\vec{\Lambda}(p_k(f))g_1 \ldots g_n) \notin dom(\bigcap_Y Y \to Y) \subseteq \{h : \mathbf{N} \rightharpoonup \mathbf{N} \mid t; h; t^{-1}(r, n)\downarrow\}.$$

Contradiction. $\square$

5 Maximal Theory for the Simply Typed Lambda Calculus Extended with Ground Permutations

Immediate generalizations of the Sierpinski PER $\mathcal{O}$ to k equivalence classes fail to give models fully complete for the simply typed λ-calculus with more than two ground constants. However, a suitable generalization of $\mathcal{O}$ gives rise to a model fully complete w.r.t. λ_k extended with constants for all *transpositions*[2] of type $o \to o$. The proof of full completeness is based on an appropriate version of the Decomposition Theorem for the extended calculus. The proof of λ-definability is rather difficult, and it requires an Approximation Theorem along the lines of Theorem 4.2 and an intermediate model, which turns out to be itself fully complete w.r.t. an extended language.

We start by introducing the PER $\mathcal{O}_k$ on $\mathcal{A}_{\mathsf{PInv}}$, with k distinct equivalence classes and the model induced by it.

Definition 5.1 (The PER $\mathcal{O}_k$) *Let $\mathcal{M}_{\mathcal{O}_k} = (CCPER_{\mathcal{O}_k}, \bullet_{\mathcal{O}_k}, [\![\ ]\!]^{\mathcal{O}_k})$ be the PER model induced by the PER $\mathcal{O}_k$ defined as follows. Fix distinct natural numbers ("moves") $*_1, \ldots *_k$, $a_1, \ldots, a_k$. Let $\mathcal{O}_k$ be the PER on the combinatory algebra $\mathcal{A}_{\mathsf{PInv}}$ consisting of k equivalence classes defined by:*
- $c_1 = \{f : \boldsymbol{N} \rightharpoonup_{\mathsf{Inv}} \boldsymbol{N} \mid f(*_1) = a_1 \ \wedge\ \forall m \neq 1.\ *_m, a_m \notin dom(f)\}$
- $\ldots$
- $c_k = \{f : \boldsymbol{N} \rightharpoonup_{\mathsf{Inv}} \boldsymbol{N} \mid f(*_k) = a_k \ \wedge\ \forall m \neq k.\ *_m, a_m \notin dom(f)\}$.

It is easy to check that the equivalence classes of the PER $\mathcal{O}_k \multimap \mathcal{O}_k$ correspond to the permutations from $\mathcal{O}_k$ to $\mathcal{O}_k$. I.e. an involution f belongs to $\mathrm{dom}(\mathcal{O}_k \multimap \mathcal{O}_k)$ if and only if $\forall c_i \exists c_j.\ f \bullet c_i = c_j$. Different permutations are in different equivalence classes of $\mathcal{O}_k \multimap \mathcal{O}_k$. It is standard that all permutations can be obtained by suitably composing elementary permutations, i.e. *transpositions*. Permutations (transpositions) of ground type are sufficient to λ-define all the elements of $\mathcal{M}_{\mathcal{O}_k}$, i.e. $\mathcal{M}_{\mathcal{O}_k}$ is fully complete. The Decomposition Theorem below allows to recover, for any given partial involution f, the Böhm tree corresponding to f (up-to permutations):

Theorem 5.1 (Decomposition) *Let $T = T_1 \to \ldots \to T_n \to o \in SimType$, $n \geq 0$, where, for all $i = 1, \ldots, n$, $T_i = U_{i1} \to \ldots \to U_{iq_i} \to o$. If $f \in dom(\otimes_{i=1}^n ![\![T_i]\!]^{\mathcal{O}_k} \multimap \mathcal{O}_k)$, then*

- *either $f \in dom([\![\vec{x} : \vec{T} \vdash c_i : o]\!]^{\mathcal{O}_k})$, for some c_i*
- *or $\exists i \in \{1, \ldots, n\}$, $\exists p_{\mathcal{O}_k} : \mathcal{O}_k \multimap \mathcal{O}_k$ permutation, and $\exists g_1, \ldots, g_{q_i}$, where $\forall j \in \{1, \ldots q_i\}$. $g_j \in dom(\otimes_{i=1}^n ![\![T_i]\!]^{\mathcal{O}_k} \multimap [\![U_{ij}]\!]^{\mathcal{O}_k})$, such that*

[2] I.e. permutations which exchange exactly two elements.

$$f\ (\times_{i=1}^{n} ![\![T_i]\!]^{\mathcal{O}_k} \multimap \mathcal{O}_k)\ (con_{\otimes_{i=1}^{n} ![\![T_i]\!]^{\mathcal{O}_k}}; (\pi_i^n \otimes \langle g_1, \ldots, g_{q_i}\rangle^\dagger); Ap); p_{\mathcal{O}_k}\ .$$

The proof of Theorem 5.1 follows the usual standard pattern, and it is carried out in detail in Section 5.1. The proof of λ-definability uses the Decomposition Theorem, and two further ingredients: an *Approximation Theorem*, and an intermediate model. These are discussed in Section 5.2.

5.1 Proof of the Decomposition Theorem

Let $T \in$ SimType, and $\mathcal{R} = [\![T]\!]^{\mathcal{O}_k}$, where $\mathcal{R} = \otimes_{i=1}^{n} !\ \mathcal{R}_i \multimap \mathcal{O}_k$, and, for all $i = 1, \ldots, n$, $\mathcal{R}_i = \otimes_{j=1}^{q_i} !S_{ij} \multimap \mathcal{O}_k$. Let $f \in \mathrm{dom}(\otimes_{i=1}^{n} !\ \mathcal{R}_i \multimap \mathcal{O}_k)$. We analyze the behaviour of f as operator in an application to arguments

$$!g_1 \in !\ \mathcal{R}_1, \ldots, !g_n \in !\ \mathcal{R}_n\ .$$

I.e., let us apply the coding functions t, p of Section 2.2, in order to get

$$\overline{f} : ((\mathbf{N}\times\mathbf{N}) + \ldots + (\mathbf{N}\times\mathbf{N})) + \mathbf{N} \rightharpoonup ((\mathbf{N}\times\mathbf{N}) + \ldots + (\mathbf{N}\times\mathbf{N})) + \mathbf{N}\ ,$$

where in the domain (codomain) of $\overline{f}$ there are n occurrences of $\mathbf{N}\times\mathbf{N}$, each one corresponding to one of the n arguments to which f applies. In the interaction with the i-th argument $!g_i$ only the i-th occurrence of $\mathbf{N}\times\mathbf{N}$ in the domain (codomain) of $\overline{f}$ is involved. In particular, the lefthand occurrence of $\mathbf{N}$ in $\mathbf{N}\times\mathbf{N}$ refers to the copy of the argument $!g_i$ used, while the righthand occurrence of $\mathbf{N}$ carries the values from (to) g_i.

We have two possibilities, according to the behaviour of $\overline{f}$ on the inputs $*_1, \ldots, *_k$ (other input values are not relevant, by definition of the PER $\mathcal{O}_k$):

Lemma 5.1 *Let $f \in dom(\vec{\mathcal{R}} \multimap \mathcal{O}_k)$, where $\mathcal{R} = \otimes_{i=1}^{n} !\ \mathcal{R}_i \multimap \mathcal{O}_k$, and, for all $i = 1, \ldots, n$, $\mathcal{R}_i = \otimes_{j=1}^{q_i} !S_{ij} \multimap \mathcal{O}_k$. Then*

(1) either $\exists *_j\ .\ \overline{f}(r, *_j) = (r, a_j)\ \wedge\ \forall *_k \neq q_j.\ ((r, *_k), (r, a_k)) \notin dom(\overline{f})$.
I.e. $f \in [\![\vec{x} : \vec{T} \vdash c_j : o]\!]^{\mathcal{O}_k}$.
(2) or $\exists i \in \{1, \ldots, n\}$ such that
(a) $\forall *_j\ \ \exists i_0.\ \overline{f}(r, *_j) = (l, (i, (i_0, m)))\ \wedge\ m = (r, y_k)$ where $y_k \in \{*_k, a_k\}$ and
(b) $\forall i, j\ (\overline{f}(r, *_j) = (l, (i, (i_0, (r, *_k))))\ \Rightarrow\ \overline{f}(l, (i, (i_0, (r, a_k)))) = (r, a_j)\ \wedge$ $\overline{f}(r, *_j) = (l, (i, (i_0, (r, a_k))))\ \Rightarrow\ \overline{f}(l, (i, (i_0, (r, *_k)))) = (r, a_j)$.

Proof. Item 1 is easy to prove. We focus on the proof of item 2. As far as item 2a, one can easily show that, for any given $*_j$, $\exists i, i_0$ s.t. $\overline{f}(r, *_j) = (l, (i, (i_0, m)))$ and $m = (r, y_k)$. The proof of the fact that, for all $*_j$, the argument i interrogated by f is the same is based on a "counting argument". If we "split" the responses to initial questions $*_j$ among different arguments, then we lose totality, because of the constraints of being a partial involution. Namely, assume by contradiction that it is not the case that for all initial questions $*_j$ the responses are in the same argument. Then, for all $i = 1, \dots, n$ $\exists p_i$ s.t.

$$\forall *_j \ \ \forall i_0 \in \mathbf{N}.\ \overline{f}(r, *_j) \neq (l, (i, (i_0, (r, x_{p_i})))) \ ,$$

for $x \in \{*, a\}$.
Then consider constants $(\mathbf{K}c_{p_1}), \dots, (\mathbf{K}c_{p_n})$ in $\mathcal{R}_1, \dots, \mathcal{R}_n$. Then we have $\forall *_j \,.\, f \bullet (\mathbf{K}c_{p_1}) \dots (\mathbf{K}c_{p_n})(*_j)\uparrow$, i.e. $f \notin \mathrm{dom}(\vec{\mathcal{R}} \multimap \mathcal{O}_k)$. Contradiction.

Finally, in order to prove item 2b, one can proceed by contradiction, and by case analysis. This concludes the proof of Lemma 5.1. $\square$

Using Lemma 5.1 above, one can easily prove the following two lemmata: *Linearization of Head Occurrence* and *Linear Function Extensionality.* The factorization of the proof of the Decomposition Theorem in these two lemmata is standard, but notice the special form of *Linear Function Extensionality*, where permutations come into play.

Lemma 5.2 (Linearization of Head Occurrence) *Let* $\mathcal{R} = \otimes_{i=1}^{n} !\, \mathcal{R}_i \multimap \mathcal{O}_k$, *where, for all* $i = 1, \dots, n$, $\mathcal{R}_i = \otimes_{j=1}^{q_i} !S_{ij} \multimap \mathcal{O}_k$. *Then, for all* $f \in dom(\vec{\mathcal{R}} \multimap \mathcal{O}_k$ *), where* $\vec{\mathcal{R}}$ *is an abbreviation for* $\otimes_{i=1}^{n} !\, \mathcal{R}_i$, *there exist* $i \in \{1, \dots, n\}$ *and* $f' \in dom(R_i \multimap \vec{\mathcal{R}} \multimap \mathcal{O}_k)$ strict, *i.e.* $t + t; t; f'; t^{-1}; t^{-1} + t^{-1}(r, (r, *)) = (l, (r, *))$, *such that*

$$f \ \ \mathcal{R} \ \ (con_{\vec{\mathcal{R}}}; \pi_i^n \otimes id_{\vec{\mathcal{R}}}; \Lambda^{-1}(f')) \ .$$

$\square$

Now we examine the structure of f'. One can show that:

Lemma 5.3 (Linear Function Extensionality) *Let* $\mathcal{S}, \mathcal{R}$ *be PERs. Then, for all* $f \in dom((\mathcal{S} \multimap \mathcal{O}_k) \multimap (\mathcal{R} \multimap \mathcal{O}_k))$ *strict, there exists* $f' \in dom(\mathcal{R} \multimap \mathcal{S})$ *such that*

$$f \ \ ((\mathcal{S} \multimap \mathcal{O}_k) \multimap (\mathcal{R} \multimap \mathcal{O}_k)) \ \ (\Lambda(((id_{\mathcal{S}} \multimap p_{\mathcal{O}_k}) \otimes f'); Ap)) \ ,$$

where $p_{\mathcal{O}_k}$ *is a permutation in the PER* $\mathcal{O}_k \multimap \mathcal{O}_k$.

Finally, we have:

Proof of the Decomposition Theorem 5.1. If case 1 of Lemma 5.1 applies, then we are done. If case 2 applies, then, by Lemma 5.2, there exist $i \in \{1, \ldots, n\}$ and $f' \in \mathrm{dom}(\mathcal{R}_i \multimap (\vec{\mathcal{R}} \multimap \mathcal{O}_k))$ such that $f\ (\vec{\mathcal{R}} \multimap \mathcal{O}_k)\ (\mathrm{con}_{\vec{\mathcal{R}}}; \pi_i^n \otimes \mathrm{id}_{\vec{\mathcal{R}}}; \Lambda^{-1}(f'))$. By Lemma 5.3, $\exists g, p_{\mathcal{O}_k}$ such that $f'\ ((\vec{\mathcal{S}}_i \multimap \mathcal{O}_k) \multimap (\vec{\mathcal{R}} \multimap \mathcal{O}_k))\ \Lambda(((\mathrm{id}_{\vec{\mathcal{S}}_i} \multimap p_{\mathcal{O}_k}) \otimes g); \mathrm{Ap})$. Then $f\ (\vec{\mathcal{R}} \multimap \mathcal{O}_k)\ \mathrm{con}_{\vec{\mathcal{R}}}; (\mathrm{id}_{\vec{\mathcal{S}}_i} \multimap p_{\mathcal{O}_k}); \pi_i^n \otimes g; \mathrm{Ap}$. Finally, by Proposition 2.2(2), by definition of the product of PERs and by the universality property of the product, we obtain $g\ (\vec{\mathcal{R}} \multimap \vec{\mathcal{S}}_i)\ \langle g_1, \ldots, g_{q_i} \rangle^\dagger$, for some $g_1 \in \mathcal{S}_{i1}, \ldots, g_{q_i} \in \mathcal{S}_{iq_i}$. $\square$

5.2 Proof of λ-definability

The proof of the λ-definability property of $\mathcal{M}_{\mathcal{O}_k}$ is quite involved and it uses an Approximation Theorem in the line of Theorem 4.2 of Section 4, and an intermediate PER model $\mathcal{M}_{\mathcal{O}_k^\perp}$ for the simply typed λ-calculus λ_k with k ground constants plus the extra *undefined* ground constant $\perp$. This model allows for *partial* elements (*approximants*).

5.2.1 The Approximation Theorem.

The notion of *approximant* of a partial involution is defined as in Definition 4.3. The proof of the following theorem is similar to the proof of Theorem 4.2.

Theorem 5.2 (Approximation) *Let $T_1 \to \ldots \to T_n \to o$ be a simple type, and let $f \in dom(\otimes_{i=1}^n ![\![T_i]\!]^{\mathcal{O}_k} \multimap \mathcal{O}_k)$. Then*

$$f\ (\otimes_{i=1}^n ![\![T_i]\!]^{\mathcal{O}_k} \multimap \mathcal{O}_k)\ (\bigcup_{j \in \omega} p_j(f))\ .$$

5.2.2 The PER model $\mathcal{M}_{\mathcal{O}_k^\perp}$.

Definition 5.2 *Let $\mathcal{M}_{\mathcal{O}_k^\perp}$ be the PER model induced by the ground PER $\mathcal{O}_k^\perp$ defined as follows. Fix distinct natural numbers $*_1, \ldots *_k,\ a_1, \ldots, a_k$. Let $\mathcal{O}_k^\perp$ be the PER on the ACA $\mathcal{A}_{\mathsf{PInv}}$ consisting of $k+1$ equivalence classes defined by:*

$$\begin{array}{rcl} \perp & = & \{f : \boldsymbol{N} \rightharpoonup_{\mathsf{Inv}} \boldsymbol{N} \mid \forall i \in \{1, \ldots, k\}.\ *_i, a_i \notin dom(f)\} \\ c_1 & = & \{f : \boldsymbol{N} \rightharpoonup_{\mathsf{Inv}} \boldsymbol{N} \mid f(*_1) = a_1\ \wedge\ \forall m \neq 1.\ *_m, a_m \notin dom(f)\} \\ & \ldots & \\ c_k & = & \{f : \boldsymbol{N} \rightharpoonup_{\mathsf{Inv}} \boldsymbol{N} \mid f(*_k) = a_k\ \wedge\ \forall m \neq k.\ *_m, a_m \notin dom(f)\}. \end{array}$$

$\mathcal{M}_{\mathcal{O}_k^\perp}$ is a model of $\lambda_k^\perp$ plus the extra undefined constant $\perp$. In particular, approximants are in $\mathcal{M}_{\mathcal{O}_k^\perp}$. One can check that $\mathcal{M}_{\mathcal{O}_k^\perp}$ is not fully complete for $\lambda_k + \perp$ (e.g. consider the type $o \to o \to o$). A remarkable fact is that the model $\mathcal{M}_{\mathcal{O}_k^\perp}$, for $k = 1$, should be fully complete for the decidable fragment of PCF called "Unary PCF".

The relationship with the model $\mathcal{M}_{\mathcal{O}_k}$ is given by the following lemma:

Lemma 5.4 *Let* $f \in dom(\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k})$. *Then*
i) $f \in dom(\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k^\perp})$.
ii) $\exists J \geq 0.\ f\ \llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k^\perp}\ p_J(f)$.

Proof. i) Assume that $f \in \mathrm{dom}(\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k})$. By the Approximation Theorem 5.2, $f \sim \bigcup_{j\in\omega} p_j(f)$ in $\mathcal{M}_{\mathcal{O}_k}$. Moreover, for all j, $p_j(f) \in \mathrm{dom}(\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k^\perp})$, and hence also $\bigcup_{j\in\omega} p_j(f) \in \mathrm{dom}(\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k^\perp})$. Then, using an argument similar to the one used in the proof of the Approximation Theorem, one can check that, $\forall \vec{g} \in \llbracket \vec{T} \rrbracket^{\mathcal{O}_k^\perp}.\ \forall *_i\ .\ f \bullet \vec{g}(*_i) \simeq \bigcup_{j\in\omega} p_j(f) \bullet \vec{g}(*_i)$. Hence, in particular, $f \in \mathrm{dom}(\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k^\perp})$.
ii) By the proof of item i) of this lemma, $f \sim \bigcup_{j\in\omega} p_j(f)$ in $\mathcal{M}_{\mathcal{O}_k^\perp}$, and $\forall j.\ p_j(f) \in \mathrm{dom}(\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k^\perp})$. Moreover, since $\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k^\perp}$ has only finitely many equivalence classes, by Lemma 4.3, there exists $J \geq 0$ such that, for all $m_1, m_2 \geq J$, $p_{m_1}(f) \llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k^\perp} p_{m_2}(f)$. Hence $f \sim p_J(f)$ in $\mathcal{M}_{\mathcal{O}_k^\perp}$. $\square$

5.2.3 λ-definability of $\mathcal{M}_{\mathcal{O}_k}$.

Finally, we are in the position of proving that all partial involutions $f \in \mathrm{dom}(\llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k})$ are λ-definable. We proceed by induction on types. The base case is easy. Let us consider the induction step.

By the Approximation Theorem 5.2, $f \llbracket \vec{T} \multimap o \rrbracket^{\mathcal{O}_k} (\bigcup_{j\in\omega} p_j(f))$. By Lemma 5.4ii), there exists J such that $f \sim p_J(f)$ in the model $\mathcal{M}_{\mathcal{O}_k^\perp}$. Hence, using Lemma 5.4i), we have, in particular, that: $\forall \vec{g} \in \llbracket T \rrbracket^{\mathcal{O}_k}.\ \forall *_i\ .\ f \bullet \vec{g}(*_i) \simeq p_J(f) \bullet \vec{g}(*_i)$. Therefore, $p_J(f) \sim f$ in $\mathcal{M}_{\mathcal{O}_k}$. Let us call P_J the λ-term whose interpretation in $\mathcal{M}_{\mathcal{O}_k^\perp}$ is $p_J(f)$. Two cases can arise:
1) $\exists \vec{M} : \vec{T}$, $\vec{M}$ $\perp$-free, such that $P_J \vec{M} =_\beta \perp$.
2) $\forall \vec{M} : \vec{T}$, $\vec{M}$ $\perp$-free, such that $P_J \vec{M} \neq_\beta \perp$.
If case 1 applies, then we have a contradiction, since $\perp \notin \mathcal{O}_k$. If case 2 applies, then one can check that P_J is equivalent in the maximal theory to any λ-term P' obtained from P_J by substituting any constant $c \neq \perp$ for each possible oc-

currence of $\perp$ in P_J. But then, since, by induction hypothesis, $\forall \vec{g} \in [\![T]\!]^{\mathcal{O}_k}$, $\vec{g}$ is λ-definable, $[\![P']\!]^{\mathcal{O}_k} \bullet \vec{g} \sim [\![P_J]\!]^{\mathcal{O}_k} \bullet \vec{g}$ in $\mathcal{M}_{\mathcal{O}_k}$, and hence $[\![P']\!]^{\mathcal{O}_k} \sim p_J(f) \sim f$, i.e. f is λ-definable.

This concludes the proof of full completeness of $\mathcal{M}_{\mathcal{O}_k}$.

6 Maximal Theory for the Simply Typed Lambda Calculus with Finitely Many Ground Constants

One can build a fully complete model for the simply typed λ-calculus λ_k, by getting rid of permutations in the models $\mathcal{M}_{\mathcal{O}_k}$. There are two ways of doing this. The first is "low-level", and it amounts to cutting down the affine combinatory algebra of partial involutions, by placing additional constraints on the partial involutions, similarly to what one does in [10] for getting a fully abstract model for PCF. Alternatively, one can define a suitable *logical relation* and use it to cut down the PER model. We sketch the first technique.

For the sake of simplicitly, let us consider, in place of the set of natural numbers, the following set of inductively defined moves:

Definition 6.1 *Let $k \in \boldsymbol{N}$. We define*

$$(\boldsymbol{M}_k \ni)\ m \ ::= \ *_i \ \mid \ a_i \ \mid \ (l,m) \ \mid \ (r,m) \ \mid \ \langle j,m\rangle \ ,$$

where $i = 1,\ldots,k$ and $j \in \boldsymbol{N}$.
We regard $\boldsymbol{M}_k$ as equipped with the intrinsic coding functions $[l,r] : \boldsymbol{M}_k + \boldsymbol{M}_k \to \boldsymbol{M}_k$, and $\langle\ ,\ \rangle : \boldsymbol{N} \times \boldsymbol{M}_k \to \boldsymbol{M}_k$.

One could equivalently take the moves to be natural numbers (under suitable assumptions on coding functions), but the set $\mathbf{M}_k$ simplifies the argument. We can immediately define a function v on moves, which, for any move m, provides the index i of the basic move $*_i$ or a_i which the move m is made up. I.e.:

Definition 6.2 *Let $v : \boldsymbol{M}_k \to \boldsymbol{M}_k$ be defined as follows. For all $i \in \boldsymbol{N}$, for all $m \in \boldsymbol{M}_k$,*
$v(*_i) = v(a_i) = i$
$v((l,m)) = v((r,m)) = v(\langle i,m\rangle) = v(m)$.

Partial involutions which preserves the function v still form an affine combinatory algebra.

Proposition 6.1 (Full Completeness and Minimality) *Let $\mathcal{A}_{\mathtt{vPInv}}$ be the affine combinatory algebra whose carrier is the set of partial involutions f :*

$\boldsymbol{M}_k \to \boldsymbol{M}_k$ *such that, for all* $m \in dom(f)$*,* $v(f(m)) = v(m)$ *. Then the model induced by the PER* $\mathcal{O}_k$ *over* $\mathcal{A}_{\mathtt{vPInv}}$ *is fully complete and minimal w.r.t.* λ_k*.*

The results of Section 4.3 concerning the Context Lemma and the decidabilty of the maximal theory hold also for the model defined in this section.

7 Conclusions and Future Work

In this paper, we have shown how the technique of Linear Realizability can be used to provide fully complete models for various typed λ-calculi. Here we give a list of remarks and interesting issues which still remain to be addressed.

- In this paper, we have presented a fully-complete model for ML-types. A natural question arises: what happens beyond ML-types. Here is a partial answer. Already at the type $\mathbf{N} \to \mathbf{N}$, where *Nat* is the type of Church's numerals, i.e. $\forall X.(X \to X) \to X \to X$, the PER model of partial involutions is not fully-complete. In fact, not only all recursive functions, but even *all* functions from natural numbers to natural numbers, can be encoded in the type $\mathbf{N} \to \mathbf{N}$. A similar problem arises even if we consider the term combinatory algebra. PER models as they are defined in this paper, do not seem to give full-completeness beyond ML-types. An innovative construction is called for here.
- Another question which arises naturally is whether the PER model over the linear term combinatory algebra is fully-complete at ML-types. We conjecture that this is the case, but a proof of this fact seems difficult. A logical relation technique relating the term algebra and the term subalgebra of partial involutions could be useful here. The interest of linear term algebras lies in the fact that the PER model generated by these is essentially the PER model shown to be fully-complete at algebraic types in [25].
- We have presented a linear realizability technique for building PER categories over an LCA. These PER categories turn out to be linear categories. It would be interesting to carry on the investigation of the general properties of these categories, e.g. define coproducts, products, etc.
- For the case of the simply typed λ-calculus, we could also abstract axioms for full completeness from the lemmata in our proofs. However, these would not imply faithfulness w.r.t. the maximal theory, i.e. that the theory of models would be maximal.
- One could define fully complete Game Models for λ_k by considering strategies in the style of [8]. But, by the intensionality of the Game Semantics, these models would not realize the maximal theory, but rather the $\beta\eta$-theory.
- Models of partial involutions are worthwhile investigating also for untyped λ-calculi. For example, partial involution strategies, i.e. strategies in the [8] style, which are represented by partial involutions from Opponenent moves

to Player moves, could possibly provide fully abstract models, alternative to those in [20,26].

- In [29], a fully abstract translation of "Finitary μ-PCF" into the simply typed lambda calculus with constants is given. An interesting consequence of this result is that our model is also fully abstract for this finitary μ-PCF.
- We feel that the fully complete models based on partial involutions defined in this paper should provide a semantical proof of the decidability of the maximal theory, alternative to those of Padovani and Loader. We should capitalize on the possibility of checking equivalence of involutions by evaluating them on *finite* sets of inputs (moves).
- Finally, it is interesting to remark that partial involutions provide models for *reversible* computations [4]. A very general question is whether this is related to the decidability of the various theories which can be modeled by partial involutions.

References

[1] S. Abramsky, Retracing some paths in Process Algebra, in: U. Montanari, V. Sassone (Eds), Concur'96, Lecture Notes in Computer Science, Vol. 1119, Springer, Berlin, 1996, pp. 1–17.

[2] S. Abramsky, Interaction, Combinators, and Complexity, Notes, Siena (Italy), 1997.

[3] S. Abramsky, Axioms for Definability and Full Completeness, in: G. Plotkin, C. Stirling, M. Tofte (Eds.), Proof, Language and Interaction: Essays in Honour of Robin Milner, MIT Press, 2000, pp. 55–75.

[4] S. Abramsky, A Structural Approach to Reversible Computation, in *LCCS 2001: Proceedings of the International Workshop on Logic and Complexity in Computer Science*, edited by D. Beauquier and Y. Matiyasevich, LACL 2001, 1–16.

[5] S. Abramsky, E. Haghverdi, P. Scott, Geometry of Interaction and Linear Combinatory Algebras, Math.Struct. in Comp.Science 12 (5) (2002) 625–665.

[6] S. Abramsky, R. Jagadeesan, New foundations for the Geometry of Interaction, Inform. Comput. 111 (1) (1994) 53–119.

[7] S. Abramsky, R. Jagadeesan, Games and Full Completeness for Multiplicative Linear Logic, J. of Symbolic Logic 59 (2) (1994) 543–574.

[8] S. Abramsky, R. Jagadeesan, P. Malacaria, Full Abstraction for PCF, Inform. Comput. 163 (2000) 409–470.

[9] S.Abramsky, M.Lenisa, Fully Complete Models for ML Polymorphic Types, Technical Report ECS-LFCS-99-414, LFCS, 1999.

[10] S.Abramsky, J.Longley, Realizability models based on hystory-free strategies, Draft paper, 1999.

[11] S. Abramsky, M. Lenisa, A Fully-complete PER Model for ML Polymorphic Types, in: P. Clote, H. Schwichtenberg (Eds.), CSL'00, Lecture Notes in Computer Science, Vol. 1862, Springer, Berlin, 2000, pp. 140–155.

[12] S. Abramsky, M. Lenisa, Axiomatizing Fully Complete Models for ML Polymorphic Types, in: M. Nielsen, B. Rovan (Eds.), MFCS'00, Lecture Notes in Computer Science, Vol. 1893, Springer, Berlin, 2000, pp. 141-151.

[13] S. Abramsky, M. Lenisa, A Fully Complete Minimal PER Model for the Simply Typed λ-calculus, in: Fribourg (Ed.), CSL'01, Lecture Notes in Computer Science, Vol. 2142, Springer, Berlin, 2001, pp. 443–457.

[14] A. Asperti, G. Longo, Categories, Types ad Structures: An introduction to category theory for the working computer scientist, Foundations of Computing Series, The MIT Press, 1991.

[15] H. Barendregt, The Lambda Calculus, its Syntax and Semantics, North Holland, Amsterdam, 1984.

[16] V. Breazu-Tannen, T. Coquand, Extensional models for polymorphism, Theoret. Comput. Sci. 59 (1988) 85–114.

[17] N. Benton, P. Wadler, Linear Logic, Monads and the Lambda Calculus, in: LICS'96, 1996.

[18] G. Bierman, What is a categorical Model of Intuitionistic Linear Logic?, in: M. Dezani et al (Eds.), TLCA'95, Lecture Notes in Computer Science, Vol. 902, Springer, Berlin, 1995, pp. 78–93.

[19] R. Crole, Categories for Types, Cambridge University Press, 1993.

[20] P. Di Gianantonio, G. Franco, F. Honsell, Game Semantics for Untyped λ-calculus, in: J.Y. Girard (Ed.), TLCA'99, Lecture Notes in Computer Science, Vol. 1581, Springer, Berlin, 1999, pp. 114–128.

[21] J.Y. Girard, Interprétation functionelle et élimunation des coupures de l'arithmètique d'ordre supérieur, Thèse d'Etat, Université Paris VII, 1972.

[22] J.Y. Girard, Towards a Geometry of Interaction, Contemporary Mathematics 92 (1989) 69–108.

[23] M. Hyland, L. Ong, On full abstraction for PCF, Inform. Comput. 163 (2000) 285–408.

[24] D.J.D. Hughes, Hypergame Semantics: Full Completeness for System F, Ph.D. Thesis, University of Oxford, 1999.

[25] J. Hyland, E. Robinson, G. Rosolini, Algebraic types in PER models, in: M.Main et al. (Eds.), MFPS'90, Lecture Notes in Computer Science, Vol. 442, Springer, Berlin, 1990, pp. 333–350.

[26] A. Ker, H. Nickau, L. Ong, More Universal Game Models of Untyped λ-Calculus: The Böhm Tree Strikes Back, in: J. Flum, M. Rodriquez-Artalejo (Eds.), CSL'99, Lecture Notes in Computer Science, Vol. 1683, Springer, Berlin, 1999, pp. 405–419.

[27] A. Joyal, R. Street, D. Verity, Traced monoidal categories, Math. Proc. Comb. Phil. Soc. 119 (1996) 447–468.

[28] J. Laird, Full abstraction for functional languages with control, in: LICS'97, 1997, pp. 58–64.

[29] J. Laird, Games, control and full abstraction, Ph.D. Thesis, University of Edinburgh, 2000.

[30] F. Lawvere, Equality in hyperdoctrines and the comprehension schema as an adjoint functor, in: Proc. Symp. on Applications of Categorical Logic, 1970.

[31] R. Loader, An Algorithm for the Minimal Model, Note, 1997.

[32] V. Padovani, Decidability of all Minimal Models, in: S. Berardi, M. Coppo (Eds.), TYPES'95, Lecture Notes in Computer Science, Vol. 1158, Springer, Berlin, 1996, pp. 201–215.

[33] A. Pitts, Polymorphism is set-theoretic constructively, in: D. Pitt et al. (Ed.), CTCS'88, Lecture Notes in Computer Science, Vol. 283, Springer, Berlin, 1988, pp. 12–39.

[34] R. Seely, Linear logic, $*$-autonomous categories and cofree coalgebras, in: Category theory, computer science and logic, American Math. Society, 1987.

[35] R. Seely, Polymorphic linear logic and topos models, in: Math. Reports, Academy of Science (Canada) XII, 1990.

[36] R. Statman, Completeness, invariance and λ-definability, J. of Symbolic Logic 47 (1) (1982).

[37] R. Statman, λ-definable functionals and $\beta\eta$-conversion, Arch. Math. Logik 23 (1983) 21–26.