Fully Complete Models for ML Polymorphic Types

Samson Abramsky Marina Lenisa*

Laboratory for the Foundations of Computer Science

University of Edinburgh, Scotland.

October 28, 1999

Abstract

We present an *axiomatic* characterization of models *fully-complete* for *ML-polymorphic types* of system F. This axiomatization is given for *hyperdoctrine* models, which arise as *adjoint models*, i.e. *co-Kleisli categories* of suitable *linear categories*. Examples of adjoint models can be obtained from categories of *Partial Equivalence Relations* over *Linear Combinatory Algebras*. We show that a special linear combinatory algebra of *partial involutions* induces an hyperdoctrine which satisfies our axiomatization, and hence it provides a fully-complete model for ML-types.

Introduction

In this paper we address the problem of *full completeness* for system F. A *categorical* model of a type theory (or logic) is said to be *fully-complete* ([AJ94a]) 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 intailment $A \vdash B$. The notion of full-completeness is the counterpart of the notion of *full abstraction*, in the sense that, if the term language is executable, then a fully-complete model is (up-to a possible quotient) fully-abstract. 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.

The importance of fully (and faithfully) complete, and fully-abstract *denotational* models is that they characterize the space of proofs/programs in a *compositional*, *syntax-independent* way. These models can give hindsigths in various directions. E.g., in the context of *static analysis*, Denotational Semantics has suggested interesting typing disciplines (see e.g. [CDHL82, Abr91]). Moreover, these models can also yield new *mathematical* principles for reasoning on syntactical theories (observational equivalences), like for example Scott's Induction Principle.

Recently, Game Semantics has been used to define fully-complete models for various fragment of Linear Logic ([AJ94a, AM99]), and to give fully-abstract models for many programming languages, including PCF [AJM96, HO96, Nic94], richer functional languages

*Work supported by TMR Linear FMRX-CT98-0170.

[AM95, McC96, HY97], and languages with non-functional features such as reference types and non-local control constructs [AM97, AM97a, AM97b, Lai97].

Once many concrete fully-complete and fully-abstract models have been studied, the problem of abstracting and axiomatizing the key properties of these constructions arises naturally. This line of research has been started by [Abr97], 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 completeness.

The axioms for PCF are abstracted from the key lemmas in the proof of full abstraction of the game model of [AJM96]. This proof makes essential use of the underlying *linear* structure of the game category. Correspondingly, the axiomatization in [Abr97] 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 [Bie95, BW96].

The problem of *full completeness* for second order (polymorphic) λ-calculus, i.e. Girard's system F ([Gir72]), is a very important problem, which has been extensively studied.

Here are some results in the literature.

In [HRR90], 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*. 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 and $FV(T) \subseteq \{X_1, \ldots, X_n\}$. The rank of an ML-type 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 [BC88], 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 [Hug97], a fully and faithfully complete game model for system F has been given. But, although this is a game model, it still has a syntactical flavour.

Summarizing the situation, the previous work on the full completeness problem for system F has produced semantically satisfactory models only for algebraic types. In this paper, we provide satisfactory denotational models fully-complete for the whole class of ML-types.

This paper consists of three parts.

In the first part, we provide an axiomatization of models fully-complete for ML-types. This axiomatization is given on the models of system F which are called *hyperdoctrines* ([Cro93]). As in [Abr97], our axiomatization also works in the context of *adjoint models*. It consists of two crucial steps. First, we axiomatize the fact that every morphism $f : 1 \to [\![T]\!]$, where T is 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.

In the second part of the paper, we present a *linear realizability* technique for building hyperdoctrine adjoint models. This technique allows us to construct a PER category over a *Linear Combinatory Algebra*, which turns out to be a linear category, and it forms an adjoint model with its co-Kleisli category. The notion of Linear Combinatory Algebra (LCA) introduced by Abramsky ([Abr96]) refines the standard notion of Combinatory Algebra, in the same way in which intuitionistic linear logic refines intutitionistic logic. The construction of PER models from LCA's presented in this paper is quite simple and clear, and it yields models with *extensionality* properties, thus avoiding cumbersome quotienting operations which are often needed in defining game categories and models. Recently, there has been much interest in realizability techniques, and in particular in linear realizability, especially in connection with full completeness and full abstraction problems. Realizability can be regarded as a powerful tool for mediating between *intensional* and *extensional* aspects of computation, and it has been used for extensionalizing intensional constructions (e.g. in [AM99]), and as a technique for building directly interesting (possibly fully-complete/fully-abstract) models. Examples of this latter use of realizability are in this paper, and also in [AL99], where a fully-abstract PER model for PCF, alternative to the game model of [AJM96], is provided using the linear algebra of *well-bracketed strategies*.

In the third part of the paper, we build an example of a concrete fully-complete model for ML-types. This is built by linear realizability over a special linear combinatory algebra of *partial involutions*. This algebra arises in the context of Abramsky's generalization of Girard's *Geometry of Interaction* ([AJ94, Abr96, Abr96a, AHPS98]). The proof of full completeness consists in showing that this model 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. This is the model generated by the *Sierpinski PER* and it consists of all (possibly infinite) Böhm trees of the typed λ-calculus, with constants $\bot, \top$. A crucial step in our proof consists in proving that, in the simply typed λ-calculus with *typical ambiguity* and $\bot$-constants, "totality tests" are λ-definable by finite typed trees. These totality tests allow us to tell apart terms in which $\bot$ appears from terms in which $\bot$ does not appear. A further ingredient is an *Approximation Lemma*, in the line of [AJM96].

The authors are thankful to R.Jagadeesan, J.Laird, J.Longley, S.Martini for useful discussions on some of the issues of the paper.

Contents

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

First, we recall the syntax of the simply typed λ-calculus with type variables and constants, and of system F, and we introduce some notation. Then, we present two important results on the simply typed λ-calculus with a theory of *Typical Ambiguity*. Typical Ambiguity theories are obtained from theories of the simply typed λ-calculus by requiring that two terms are equated if and only if, for all possible substitutions of type variables, they are equated in the theory on simply typed λ-calculus. The first result that we present is Statman's *Typical Ambiguity Theorem*, which ensures that there is exactly one consistent theory of Typical Ambiguity on the simply typed λ-calculus with infinite type variables, i.e. the $\beta\eta$-theory. An immediate consequence of this is 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 infinite type variables, $\perp$-constants, and a theory of Typical Ambiguity. In particular, we prove that, for any given type, there are covergence test terms, which detect the presence of $\perp$-constants in a term of that type. This implies immediately that, in a theory of Typical Ambiguity over the simply typed λ-calculus with infinite type variables and $\perp$-constants, a term with $\perp$ can never be equated to a term without $\perp$.

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:*

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

where $X \in \mathit{TVar}$.
Raw Terms are defined as follows:

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

where $c \in C$ is a set of constants.

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 $x_1 : T_1, \ldots , x_n : T_n$. The rules of the proof system are the following:*

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

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

$\beta\eta$-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* $x \notin \mathit{dom}(\Delta)$.

Notation. We call:

λ	:	the simply typed λ-calculus with $TVar = \{\iota\}$;
$\lambda_\perp$	:	the simply typed λ-calculus with $TVar = \{\iota\}$ and a base constant $\perp \in C$ of type ι;
$\lambda_{\perp,\top}$	:	the simply typed λ-calculus with $TVar = \{\iota\}$ and two base constants $\perp, \top$ of type ι;
λ^∞	:	the simply typed λ-calculus with infinite type variables in $TVar$ and $C = \emptyset$;
$\lambda^\infty_\perp$	:	the simply typed λ-calculus with infinite type variables in $TVar$ and base constants $\perp$ of type ι, for any $\iota \in TVar$;
$\lambda^\infty_{\perp,\top}$	:	the simply typed λ-calculus with infinite type variables in $TVar$ and base constants $\perp$ of type ι, for any $\iota \in TVar$.

Definition 1.2 (System F) *The class* Type *of system F types over an infinite set of type variables* TVar *is defined by:*

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

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 $x \in Var$.

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:

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

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

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

(*) if $X \notin FV(ran(\Delta))$.

$\beta\eta$-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 1.3 (ML-types) *The class* ML-Type *of ML-types is defined by:*

$$ML\text{-}Type = \{\forall \vec{X}.T \mid T \in \; SimType \;\wedge\; FV(T) \subseteq \vec{X}\} \; .$$

Terms of ML-types have essentially the same "combinatorics" as terms of the simply typed λ-calculus. More precisely, ML-terms can be regarded as the terms of the λ^∞-calculus with a Typical Ambiguity theory.

1.1 Statman's Typical Ambiguity Theorem

The following is a result about simply typed λ^∞ first proved in [Sta88]. An immediate consequence of this theorem is that the theory at ML-types of any non-trivial model of system F is exactly the $\beta\eta$-theory.

Theorem 1.1 (Statman's Typical Ambiguity) *Let T be a type of λ^∞ such that $FV(T) \subseteq \{X_1, \ldots, X_n\}$. If $\not\vdash M =_{\beta\eta} N \;:\; T$, then, there exist types $S_1, \ldots, S_n$, and $Y \in TVar$, and a term L such that $\vdash L[\vec{S}/\vec{X}] : T \to Bool_Y$, where $Bool_Y = Y \to Y \to Y$, such that*

$$\vdash LM_{\vec{S}} =_{\beta\eta} true \;:\; Bool_Y \;\wedge\; \vdash LN_{\vec{S}} =_{\beta\eta} false \;:\; Bool_Y \; ,$$

where $true = \lambda x : Y.y : Y.x$ and $false = \lambda x : Y.y : Y.y$, and $M_{\vec{S}}$ denotes the term M in which the type variables $\vec{X}$ have been substituted by $\vec{S}$.

Corollary 1.1 *The maximal consistent theory of Typical Ambiguity on λ^∞ is the $\beta\eta$-theory.*

Corollary 1.2 *The maximal consistent theory on the fragment of system F consisting of ML-types is the $\beta\eta$-theory.*

1.2 λ-definability of Covergence Tests in the λ-calculus with Typical Ambiguity

We show that "convergence tests" are λ-definable in $\lambda^\infty_\perp$ with a theory of Typical Ambiguity. This result will be used in Section 5, where we prove that the model of PERs over the linear combinatory algebra of partial involutions is fully-complete at ML-types.

Definition 1.4 (Typed Convergence Tests) *Let $T = T_1 \to \ldots \to T_n \to X_k \in SimType$, let ι be a distinguished type variable, and let $\alpha_T = T[\iota \to \iota/\vec{X}]$. We define, by induction on T, the* convergence test *term $\vdash S_{\alpha_T} : \alpha_T$ as follows:*

- *if $T = X$, then*

$$S_{\iota\to\iota} = I_{\iota\to\iota} \ ,$$

- *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*

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

Now we show how the "convergence test" terms defined above give us a procedure for deciding whether a term of $\lambda_\perp^\infty$ contains a divergent subterm. Let M be a term of $\lambda_\perp^\infty$ of type $T_1 \to \ldots \to T_n \to X_k$. We first instantiate all the free variables in M by $\iota \to \iota$, then 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 $MS_{\alpha_{T_1}}, \ldots, S_{\alpha_{T_n}}$, each subterm of M definitely appears in head position, and it reduces to the identity, until a $\perp$ is detected.

For a term $\vec{y} : \vec{U} \vdash M : T$, we denote by $\vec{y} : \vec{\alpha_U} \vdash M_{\alpha_T} : \alpha_T$ (or simply by M_{α_T}) the term of type α_T obtained from $\vec{y} : \vec{U} \vdash M : T$ by instantiating all the type variables free in T by $\iota \to \iota$.

Theorem 1.2 (Typed Separability) *Let $T = T_1 \to \ldots \to T_n \to X_k \in$SimType, and let $\vdash M : T$ be a term of $\lambda_\perp^\infty$. Then*

$$M_{\alpha_T} S_{\alpha_{T_1}} \ldots S_{\alpha_{T_n}} = \begin{cases} I_{\iota\to\iota} & \textit{if } M \textit{ is } \perp\textit{-free} \\ \lambda x : \iota.\ \perp & \textit{otherwise} \ . \end{cases}$$

Proof. Let $\vec{y} : \vec{U} \vdash M : T$ be a (possibly open) term of $\lambda_\perp^\infty$, and let $\widetilde{M}_{\alpha_T} = M_{\alpha_T}[\vec{S}_{\alpha_U}/\vec{y}]$ be the closed term obtained by saturating all the free variables in M_{α_T} by convergence tests of the appropriate types. We prove, by induction on the structure of M, that

$$\widetilde{M}_{\alpha_T} S_{\alpha_{T_1}} \ldots S_{\alpha_{T_n}} = \begin{cases} I_{\iota\to\iota} & \text{if } M \text{ is } \perp\text{-free} \\ \lambda x : \iota.\ \perp & \text{otherwise} \ . \end{cases}$$

Base case: $\vec{y} : \vec{U} \vdash x : T$. We have to prove that $S_{\alpha_T} S_{\alpha_{T_1}} \ldots S_{\alpha_{T_n}} = I_{\iota\to\iota}$. But this latter fact can be immediately shown by induction on T.
Induction Step: $\vec{y} : \vec{U} \vdash \lambda\vec{z}.x_i M_1 \ldots M_{q_i}$. Let $\widetilde{M}_{\alpha_T} = \lambda\vec{z}.x_i(\widetilde{M_1})_{\alpha_{U_{i1}}} \ldots (\widetilde{M_{q_i}})_{\alpha_{U_{iq_i}}}$. Then $\widetilde{M}_{\alpha_T} S_{\alpha_{T_1}} \ldots S_{\alpha_{T_n}} = S_{\alpha_{T_i}}(\widetilde{M_1})_{\alpha_{U_{i1}}} \ldots (\widetilde{M_{q_i}})_{\alpha_{U_{iq_i}}} = \lambda z : \iota.((\widetilde{M_1})_{\alpha_{U_{i1}}} \vec{S})(\ldots (\widetilde{M_{q_i}})_{\alpha_{U_{iq_i}}} \vec{S})z)$. The thesis follows applying the induction hypothesis to each $(\widetilde{M_j})_{\alpha_{U_{ij}}}$, for all $j = 1, \ldots, q_i$.
$\square$

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 apart normal forms from unsolvable terms.

Corollary 1.3 *In any theory of Typical Ambiguity on $\lambda_\perp^\infty$, a term in which the $\perp$-constant appears cannot be equated to a term without $\perp$-constants.*

The typed separability result above can be read by saying that all non-divergent computations can be reduced in some way to the *identity computation*. This is consistent with the view taken in the *Geometry of Interaction/Game Semantics* paradigm, whereby computations in the simply typed λ-calculus can be handled just by operations/strategies which do nothing more than *copying* information, without producing any new result.

2 Models of System F

We focus on *hyperdoctrine models* of system F. First, we recall the notion of $2\lambda\times$-hyperdoctrine (see [Cro93]). This essentially corresponds to the notion of *external model* (see [AL91]). 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. [AL91, Cro93] for more details on indexed categories).

Definition 2.1 ($2\lambda\times$-hyperdoctrine) *A* $2\lambda\times$-hyperdoctrine *is a triple* $(\mathcal{C}, \boldsymbol{G}, \forall)$*, where:*

- $\mathcal{C}$ *is the* base category*, it has with 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* $\mathcal{C}$*.*
- $\boldsymbol{G} : \mathcal{C}^{op} \to \mathrm{CCCat}$ *is a* $\mathcal{C}$*-indexed cartesian closed category, where* CCCat *is the category of cartesian closed categories and strict cartesian closed functors, 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 functors 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 $X_1, \dots, X_m$ are interpreted by morphisms from $\mathcal{U}^m$ to $\mathcal{U}$ in the category $\mathcal{C}$, i.e. by objects of $\mathbf{G}(\mathcal{U}^m)$:

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

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 2.2 *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 $\Gamma; \Delta \vdash M : T$:*

- $[\![\Gamma; x_1 : T_1, \ldots, x_i : T_i, \ldots, x_n : T_n \vdash x_i : T_i]\!] =$
  $\pi_i : [\![\Gamma \vdash T_1]\!] \times \ldots \times [\![\Gamma \vdash T_i]\!] \times \ldots \times [\![\Gamma \vdash T_n]\!] \to [\![\Gamma \vdash T_i]\!]$
- $[\![\Gamma; \Delta \vdash \lambda x.M : T \to S]\!] = \Lambda([\![\Gamma; \Delta, x : T \vdash M : S]\!])$
- $[\![\Gamma; \Delta \vdash MN : S]\!] = \langle [\![\Gamma; \Delta \vdash M : T \to S]\!], [\![\Gamma; \Delta \vdash N : T]\!] \rangle; Ap$
- $[\![\Gamma; \Delta \vdash \Lambda X.M : \forall X.T]\!] = \overline{[\![\Gamma, X; \Delta \vdash M : T]\!]}$,
  where $^-$ is the bijection given by the adjiunction between $\forall$ and π^* in Definition 2.1;
- $[\![\Gamma; \Delta \vdash MS : T[S/X]]\!] =$
  $[\![\Gamma; \Delta \vdash M : \forall X.T]\!]; \langle id_{[\![\Gamma]\!]}, [\![\Gamma \vdash S]\!] \rangle^* (\widehat{id}_{\forall([\![\Gamma, Y \vdash T[Y/X]]\!])})$,
  where $\widehat{}$ is the inverse of $^-$.

Proposition 2.1 ([Cro93]) *Any $2\lambda\times$-hyperdoctrine with the interpretation function $[\![\ ]\!]$ of Definition 2.2 is a model of system F.*

Definition 2.3 (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 $T \in \mathcal{T}$,*

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

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

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

2.1 Adjoint Hyperdoctrines

We start by recalling some definitions:

Definition 2.4 (Linear Category, [Bie95, BW96]) *A* linear category *is a symmetric monoidal closed category* $(\mathcal{L}, I, \otimes, \multimap)$ *with*

- *a symmetric monoidal comonad* $(!, der, \delta, \phi, \phi')$ *on* $\mathcal{L}$*;*
- *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 2.5 (Adjoint Model, [BW96]) *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 2.6 (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, we capture those $2\lambda\times$-hyperdoctrines which arise from a co-Kleisli construction over an *indexed linear category.*

Definition 2.7 (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, which 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 co-Kleisli category of* $\boldsymbol{L}$*, which we assume to be cartesian closed.*
- *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 2.1 *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. Notice that, in the definition of adjoint hyperdoctrine, we require the indexed categories $\mathbf{L}$ and $\mathbf{G}$ to form an adjoint model, but we assume the existence of a family of functors $\forall_m$ only on the fibre categories of $\mathbf{G}$. 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. Hence our notion of model is more general, but it is sufficient for dealing with ML-types, and for expressing axioms for full completeness at ML-types (see Section 3).

3 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. Our axiomatization of full completeness for ML polymorphism is in the line of the work in [Abr97], 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 [AJM96]. 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) 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 2.2), morphisms f in $\mathbf{G}(1)$ from the terminal object of $\mathbf{G}(1)$ into $[\![\vdash T]\!]$, where $\forall\vec{X}.T_1 \to \ldots \to T_n \to X_k$ is an 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, $f = [\![\vdash \Lambda\vec{X}.\vec{x} : \vec{T}.x_i M_1 \dots M_{q_i} : \forall\vec{X}.\vec{T} \to X_k]\!]$ if and only if $\vec{\Lambda}^{-1}(\vec{\hat{f}}) = [\![\vec{X};\vec{x} : \vec{T} \vdash x_i M_1 \dots M_{q_i} : X_k]\!]$. 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 $T_1, \dots, T_n$ 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 3.2.

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

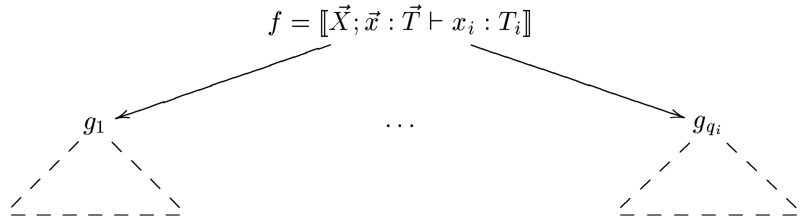
$$f = [\![\vec{X};\vec{x} : \vec{T} \vdash x_i : T_i]\!] \bullet [\![\vec{X};\vec{x} : \vec{T} \vdash M_1 : U_{i1}]\!] \bullet \dots \bullet [\![\vec{X};\vec{x} : \vec{T} \vdash M_{q_i} : U_{iq_i}]\!] \ .$$

The Decomposition Theorem allows to recover 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. Let $T = T_1 \to \dots \to T_n \to X_k$ be a simple type with $FV(T) \subseteq \{X_1, \dots, X_n\}$, where, for all $i = 1, \dots, n$, $T_i = U_{i1} \to \dots \to U_{iq_i} \to X_i$. 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 $i \in \{1, \dots, n\}$ 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 $j = 1, \dots, q_i$, such that*

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

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 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* $\pi_k : \vec{\mathcal{U}} \to \vec{\mathcal{U}}$ *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 to extract *one* copy of the type of the head variable, corresponding to the *first* use of this variable. Notice that the property expressed by this axiom is *truly* linear. In fact, in order to state it, we are implicitly using the isomorphism $A \otimes !A \simeq !A$.

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

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

  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)

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

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.

Axiom 4 (Linear Function Extensionality)

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

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)

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

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 $h_i = \otimes_{j=1}^{q_i} !l_j \multimap \pi_{p_1}$, 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 $h_i = \otimes_{j=1}^{q_i} !l_{ij} \multimap \pi_{p_1}$, for $j = 1, \ldots, q_i$, is *finite*.

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

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

such that

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

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

3.2 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 $i = 1, \ldots, n$, $h_i = \otimes_{j=1}^{q_i} !l_{ij} \multimap \pi_{p_i}$, 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. Let* $f \in Hom_{\boldsymbol{L}(\vec{\mathcal{U}})}(\vec{h}, \pi_k)$, *where* $\vec{h} = \bigotimes_{i=1}^{n} !h_i$ *and, for all* $i = 1, \ldots, n$, $h_i = \bigotimes_{j=1}^{q_i} !l_{ij} \multimap \pi_{p_i}$. *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. By Axiom 2, there exists a unique $i \in \{1, \ldots, n\}$ and a unique $f' \in \mathrm{Hom}^t_{\mathbf{L}(\vec{\mathcal{U}})}(h_i, \vec{h} \multimap \pi_k)$ 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 $\vec{l_i} = \bigotimes_{j=1}^{q_i} !l_{ij}$, 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 $g = \langle g_1, \ldots, g_{q_i} \rangle^{\dagger}$. $\square$

Summarizing the results of this section, we have:

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 $\forall \vec{X}.T = T_1 \rightarrow \ldots \rightarrow T_n \rightarrow X_k$ be an ML-type, and let $f \in \mathrm{Hom}_{\mathbf{L}(1)}(!I, [\![\vdash \forall \vec{X}.T]\!])$. One can easily prove, by induction on $\mathcal{H}(\overline{\Lambda(f)})$ that there exists $\vec{X}; \vec{x} : \vec{T} \vdash x_i M_1 \ldots M_{q_i} : X_k$ such that $\overline{\Lambda(f)} = [\![\vec{X}; \vec{x} : \vec{T} \vdash x_i M_1 \ldots M_{q_i} : X_k]\!]$.
Then $f = [\![\vdash \Lambda\vec{X}.\lambda\vec{x} : \vec{T}.x_i M_1 \ldots M_{q_i} : \forall\vec{X}.\vec{T} \rightarrow X_k]\!]$. $\square$

The Strong Decomposition Theorem is stronger than the Decomposition Theorem 3.1 in two respects. First of all, it guarantees the *unicity* of the decomposition. The unicity condition implies immediately that the model is faithful. We could give alternative forms of the axioms by substituting the isomorphisms requirements by weaker conditions, in order to ensure just the existence of a decomposition. This would be sufficient, since faithfulness of the model follows from Statman's Theorem 1.1. More precisely, 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.

The axioms could be weakened also by considering only morphisms f whose domains and codomains are *denotations* of types, instead of generic objects.

The strong form of the axioms, that we have given, has the advantage of being more readable and concise. Moreover, as we will see in Section 5, the strong axioms for the Decomposition Theorem that we have given hold in our concrete example of fully-complete model, except for the *Linearization of Head Occurrence*, for which we only prove a weak form. But we conjecture that also this axiom holds in its strong form. In Section 5, also the Finiteness Axiom is proved in a weak form. More precisely, we show that there exists a size function for morphisms whose domains and codomains are denotations of appropriate types.

4 Models of PERs over a Linear Combinatory Algebra

Canonical examples of $2\lambda\times$-hyperdoctrines arise by considering the *Partial Equivalence Relation* (PER) category over a *combinatory algebra* (see [Cro93], 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 5).

We start by recalling the definition of linear combinatory algebra ([Abr96, AHPS98]):

Definition 4.1 (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 .

LCA's correspond to *Hilbert style* axiomatization of $\multimap, !$ 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 [AHPS98] for details).

4.1 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.

Definition 4.2 *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 which satisfy *extensionality of pairs*, standard *pairing* gives rise to a tensor product:

Lemma 4.1 *Let $\mathcal{A} = (A, \bullet)$ be a BCI-algebra. Let P be the* pairing combinator*, i.e. (using λ-notation) $P = \lambda zxy.zxy$. If $\mathcal{A}$ satisfies* extensionality of pairs*, i.e.*

$$P\alpha\beta = P\alpha'\beta' \implies \alpha = \alpha' \ \wedge \ \beta = \beta' \ ,$$

then, for all PERs $\mathcal{R}, \mathcal{S}$, the following PER is well defined

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

Notice in particular that, if the BCI-algebra is *affine*, i.e. it is a BCK-algebra, then extensionality of pairs holds.

Proposition 4.1 *Let $\mathcal{A} = (A, \bullet)$ be a BCI-algebra satisfying extensionality of pairs. 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 4.1. For any 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 $[\lambda z.UzA]$, where $U = \lambda xy.xy$ and $A = \lambda xy.P(fx)(f'y)$.
The PER $I = \{(I, I)\}$ plays the role of *tensor identity.*
The following are natural isomorphisms:
$\rho_{\mathcal{R}} : \mathcal{R} \otimes I \to \mathcal{R}$, $\quad \rho_{\mathcal{R}} = [\lambda z.z(\lambda xy.yx)]$,
$\alpha_{\mathcal{R}_1,\mathcal{R}_2,\mathcal{R}_3} : (\mathcal{R}_1 \otimes \mathcal{R}_2) \otimes \mathcal{R}_3 \to \mathcal{R}_1 \otimes (\mathcal{R}_2 \otimes \mathcal{R}_3)$, $\quad \alpha_{\mathcal{R}_1,\mathcal{R}_2,\mathcal{R}_3} = [\lambda z.UzA]$,
where $A = \lambda x.UxZ$, and $Z = \lambda xyz.Px(Pyz)$;
$\sigma_{\mathcal{R}_1,\mathcal{R}_2} : \mathcal{R}_1 \otimes \mathcal{R}_2 \to \mathcal{R}_2 \otimes \mathcal{R}_1$, $\quad \sigma_{\mathcal{R}_1,\mathcal{R}_2} = [\lambda z.Uz(\lambda zz'.Pz'z)]$;
$\Lambda : (\mathcal{R}_1 \otimes \mathcal{R}_2 \multimap \mathcal{R}_3) \to (\mathcal{R}_1 \multimap \mathcal{R}_2 \multimap \mathcal{R}_3)$, $\quad \Lambda = [\lambda fxy.f(Pxy)]$. $\square$

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

Proposition 4.2 *Let $\mathcal{A} = (A, \bullet, !)$ be an LCA satisfying extensionality of pairs. Let $\boldsymbol{!} : PER_{\mathcal{A}} \to PER_{\mathcal{A}}$ be the functor defined by:*

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

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

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

Lemma 4.2 *Let $\mathcal{A} = (A, \bullet, !)$ be an LCA satisfying extensionality of pairs. Then, for all PERs $\mathcal{R}, \mathcal{S}$,*

1. (Idempotency of !) $[D] : \boldsymbol{!!}\,\mathcal{R} \simeq \boldsymbol{!}\,\mathcal{R} : [\delta]$;
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 4.2 above is relevant for full completeness. In fact, as we will see in Section 4, this isomorphism amounts exactly to the *Uniformity of Threads Axiom* in our axiomatization of full completeness. The isomorphisms of Lemma 4.2 above highlight some degeneracies of the present construction of PER models.

Theorem 4.1 *Let $\mathcal{A} = (A, \bullet, !)$ be an LCA satisfying extensionality of pairs. Then*

- *The category $PER_{\mathcal{A}}$ is linear.*
- *The co-Kleisli category $(PER_{\mathcal{A}})_!$, induced by the comonad **!** on the category $PER_{\mathcal{A}}$, is cartesian closed.*
- *The categories $PER_{\mathcal{A}}$ and $(PER_{\mathcal{A}})_!$ form an adjoint model.*
- *The category $(PER_{\mathcal{A}})_!$ 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 4.2 (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 $\mathcal{U}^n$.*

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 $h : \mathcal{U}^m \times \mathcal{U} \to \mathcal{U}$, $\forall_m(h) = \lambda\vec{X}.\bigcap_{\vec{X}} h(\vec{X})$. For any morphism $H : h_1 \to h_2$ in $\boldsymbol{L}(\mathcal{U}^m \times \mathcal{U})$, $\forall_m(H) = H$.*

4.2 Partial Involutions Affine Combinatory Algebra

Many examples of LCAs arise from Abramsky's categorical version of Girard's Geometry of Interaction (GoI) construction, based on *traced symmetric monoidal* categories ([Abr96, Abr96a, AHPS98]). A basic example of GoI LCA, introduced in [Abr96], 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

[Abr96, Abr96a, AHPS98] for more details). The LCA of partial involutions, which will be shown to provide a fully-complete model for ML-types (see Section 5), arises as subalgebra of this.

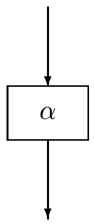
Let us consider the space $[\mathbf{N} \rightharpoonup \mathbf{N}]$ of partial functions from natural numbers to natural numbers. For any $\alpha \in [\mathbf{N} \rightharpoonup \mathbf{N}]$ injective, we denote by α^{-1} the inverse of α. Now we show how we can endow the space $[\mathbf{N} \rightharpoonup \mathbf{N}]$ 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* functions t and p:

$$t : \mathbf{N} + \mathbf{N} \triangleright \mathbf{N} : t^{-1} \qquad p : \mathbf{N} \times \mathbf{N} \triangleright \mathbf{N} : p^{-1} \ .$$

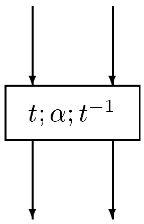
The first is used in order to define application, it allows to transform an one-input/one-output function into a two-input/two-output function, . The latter is used for creating *infinitely* many copies of an one-input/one-output function α, i.e. for defining $!\alpha$.

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 [JSV96] for the general categorical treatment).

Let us represent an one-input/one-output function $\alpha \in [\mathbf{N} \rightharpoonup \mathbf{N}]$ by the following one-input-port/one-output-port box:



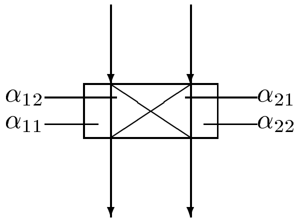
In order to define the application $\alpha \bullet \beta$, for $\alpha, \beta \in [\mathbf{N} \rightharpoonup \mathbf{N}]$, we regard α as a two-input/two-output function via the coding t, i.e.:



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. I.e., graphically:



The result of the application $\alpha \bullet \beta$ is an one-input/one-output function which can be computed as follows. For any given input (*token*) n, this becomes input for $t; \alpha; t^{-1}$ along its lefthand input wire (see Fig. 1). The function $t; \alpha; t^{-1}$ sends it either to the lefthand or to the righthand output wire. In the first case, this is the result, in the latter case the resulting token becomes input for β and then it keeps traveling along the lefthand wires of $t; \alpha; t^{-1}$ and the input wire of β, until $t; \alpha; t^{-1}$ (possibly) sends it to its righthand output wire. Formally:

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

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

The formula 1 for computing the application is essentially the *Execution Formula* from Girard's Geometry of Interaction ([Gir89]).

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:

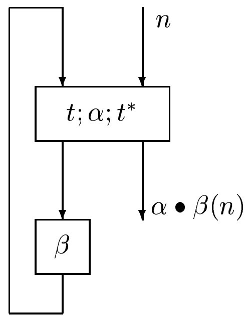


Figure 1: Linear application.

Definition 4.3 (Combinators)

I :

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

where

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

B :

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

where

- $s_{\boldsymbol{B}} : (((\boldsymbol{N} + \boldsymbol{N}) + (\boldsymbol{N} + \boldsymbol{N})) + \boldsymbol{N}) + \boldsymbol{N} \rightharpoonup \boldsymbol{N}$ *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:*
  - $\forall n.\ f_{\boldsymbol{B}}(r, n) = (l, (l, (l, (r, n))))$
  - $\forall n.\ f_{\boldsymbol{B}}(l, (l, (l, (l, n)))) = (l, (l, (r, (r, n))))$
  - $\forall n.\ f_{\boldsymbol{B}}(l, (l, (r, (l, n)))) = (l, (r, n))$.

C :

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

where

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

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

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

K :

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

where

- $s_{\boldsymbol{K}} : (\boldsymbol{N}+\boldsymbol{N}) + \boldsymbol{N} \rightharpoonup \boldsymbol{N}$ *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:*
  - $\forall n.\ f_{\boldsymbol{K}}(r,n) = (l,(r,n))$
  - $\forall n.\ f_{\boldsymbol{C}}(l,(r,n)) = (r,n)$.

W : *In order to define* $\boldsymbol{W}$, *we need first to fix* $i,j \in \boldsymbol{N}$ *such that* $i \neq j$. *Then*

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

where

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

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

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

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

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

where

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

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

where

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

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

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

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

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

where

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

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

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

There is a simple, intutitive, geometrical explanation of these combinators, which makes use of the language of boxes and wires (see Fig. 2 and 3).

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 informations from the lefthand input-wire to the righthand output-wire, and vice versa from the righthand input-wire to the lefthand output-wire.

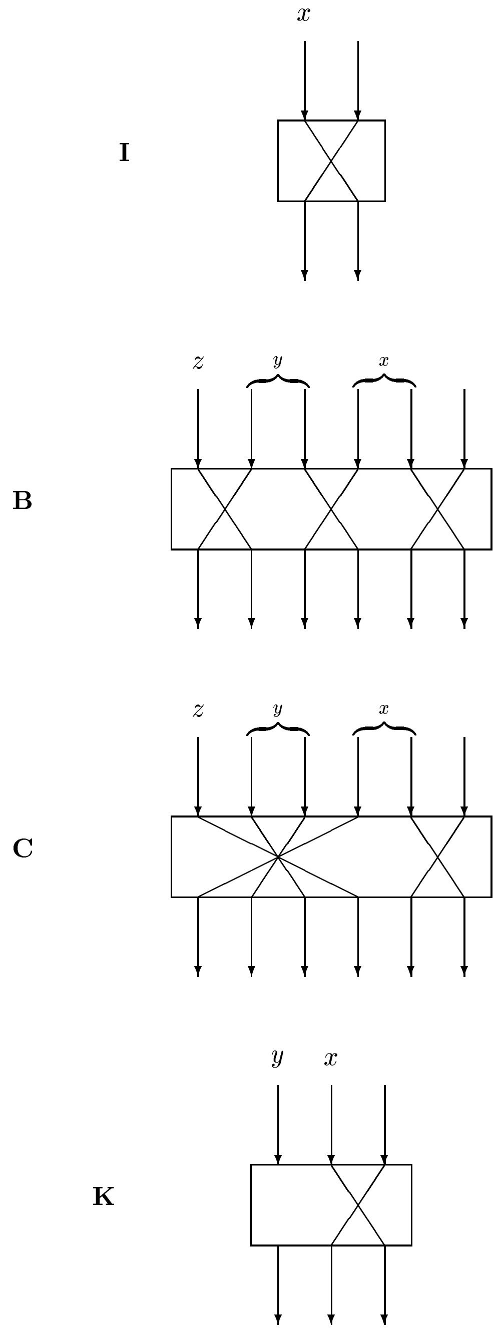


Figure 2: IBCK-combinators.

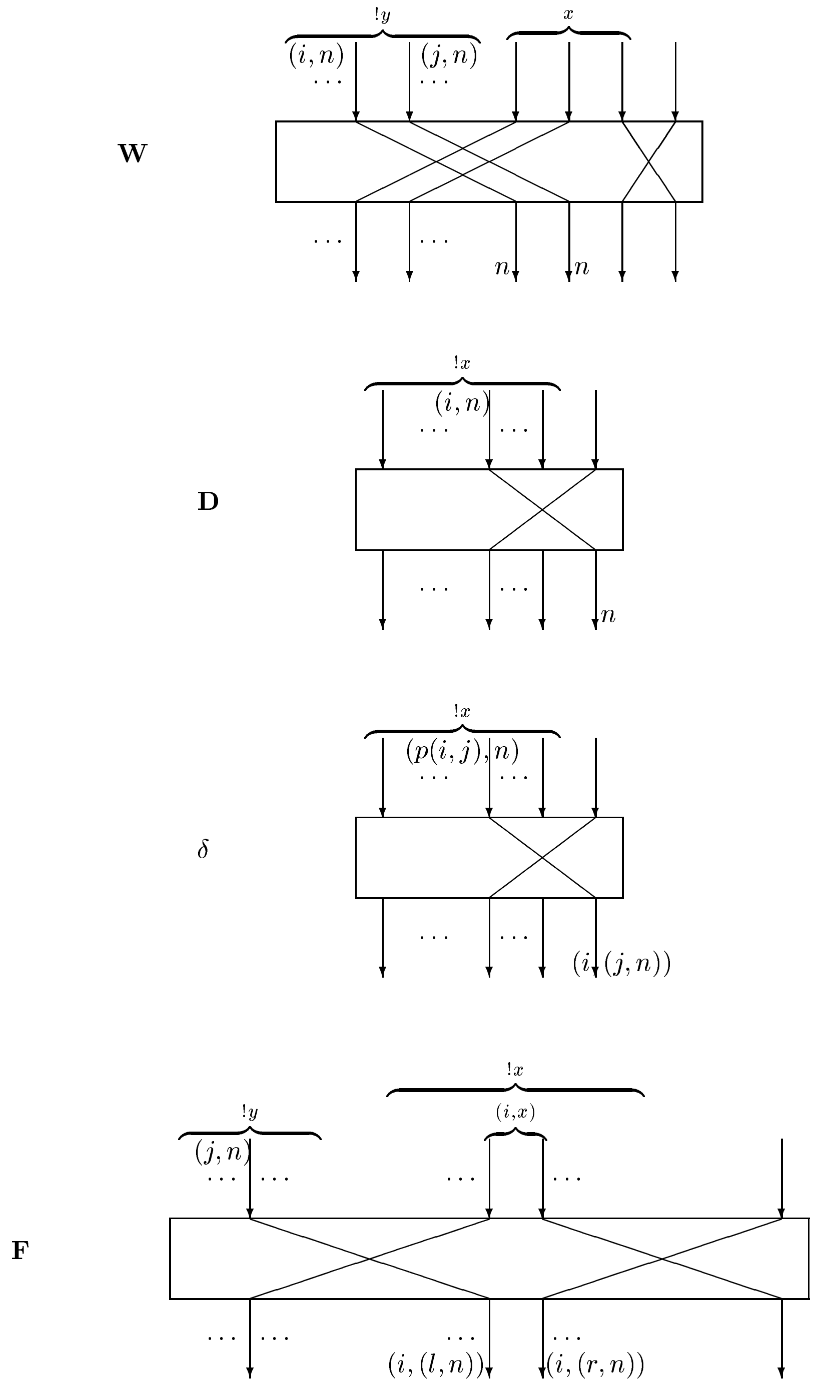
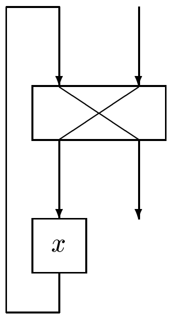
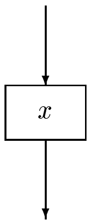


Figure 3: WDδF-combinators.

The fact that $\boldsymbol{I}$ satsfies the identity equation has a simple geometrical explanation. Let us apply $\boldsymbol{I}$ to a partial function x:

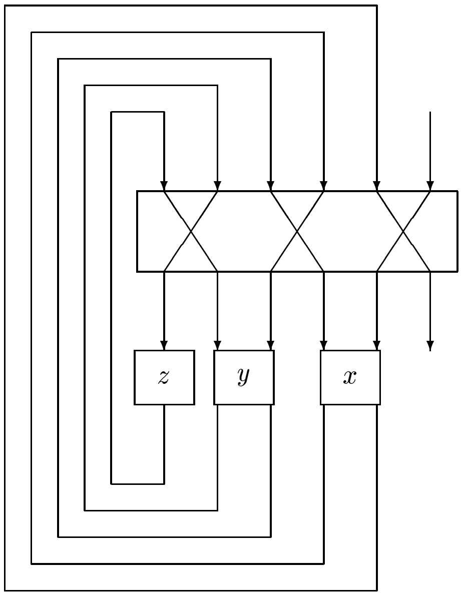


Now *yank* the string connecting the input and the output wires of the result of the application, forgetting about the box corresponding to $\boldsymbol{I}$. This gives us immediately the expected result:

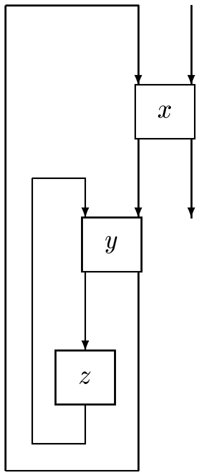


Our argument is based on the *Yanking Property* of the trace on the symmetric monoidal category *Pfn* underlying our combinatory algebra. In particular, *Yanking* is one of the axioms characterizing the trace operation in the general setting of traced symmetric monoidal categories. We do not elaborate more on this here.

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 review through 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 need to fix two different indeces $i, j \in \mathbf{N}$, tagging the copies of y which are used as arguments by x. The remaining copies of y are ignored. 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 informations 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 5 is obtained by considering *partial involutions*:

Definition 4.4 *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* $[\boldsymbol{N} \rightharpoonup_{Inv} \boldsymbol{N}]$ *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 4.3, i.e.:

Proposition 4.3 $\mathcal{A}_{PInv} = ([\boldsymbol{N} \rightharpoonup_{Inv} \boldsymbol{N}], \bullet, !)$ *is an affine combinatory algebra.*

$\mathcal{A}_{PInv}$ is a highly constrained algebra, in which all computations are *reversible*. Partial involutions are reminescent of *copy-cat* strategies of game categories, in that the only computational effect that they have is that of *copying* informations from input to output wires.

A similar idea to the one used to define $\mathcal{A}_{PInv}$ is used in [AL99], in order to provide a fully-abstract model for PCF. Here 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 [AJM96] style.

5 A Fully Complete PER Model

In this section, we prove that the PER category over the LCA $\mathcal{A}_{PInv}$ of Section 4.2 satisfies the Axioms of Section 3 (some of them in a weak form), and hence it gives rise to a fully and faithfully complete PER model for ML-types.

By definition of PER adjoint models (see Theorem 4.2 of Section 4.1), and by the fact that PER categories are well-pointed, for any morphisms h, l in the fibre category $\mathbf{L}(\vec{\mathcal{U}})$,

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

where $F : PER_{\mathcal{A}} \rightarrow Set$ 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 4.2 of Section 4.1.

The rest of this section is devoted to the proof of the validity of the Axioms 2–4, 6.

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. The most difficult part of the proof of full completeness for the model $\mathrm{PER}_{\mathcal{A}_{\mathrm{PInv}}}$ consists in proving the *Finiteness Axiom*, i.e. in ruling out infinite typed trees. The proof of this Axiom makes use of the Typed Separability result presented in Section 1.2, and it requires an *Approximation Lemma*, along the lines of [AJM96].

5.1 Proof of the Axioms 2–4

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 5.1, we show that the partial involutions in $dom(\bigcap_{\vec{X}} \bigotimes_{i=1}^{n} ! \mathcal{R}_i \multimap X_k)$, where $\forall i.\ \mathcal{R}_i{=}\mathcal{S}_i \multimap X_i$, are "total", in the sense that, for any possible sequence of arguments in input, they always "look" at them, before producing an output, and they are different from the empty partial involution. In Lemma 5.2, we show that any of this partial involutions always "ask" first for the same argument, say the i-th argument, for any possible sequence of arguments. This allows us to isolate the *first* use of a copy in $! \mathcal{R}_i$. Finally, Lemma 5.3 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 5.2 below).

Lemma 5.1 *Let $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ be a closed PER, where $\vec{\mathcal{R}} = \bigotimes_{i=1}^{n} !R_i$, 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: $X_k = \{h : \mathbf{N} \rightharpoonup_{Inv} \mathbf{N} \mid h(m)\downarrow\}$, and $X_j = 1$, for all $j \neq k$. 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 $\exists m, m'.\ f^*(r, m) = (r, m')$. $\square$

The following technical definition will be useful in the sequel.

Definition 5.1 *Let $f : \boldsymbol{N} \rightharpoonup_{Inv} \boldsymbol{N}$. Let $f^*_n : D_n \rightharpoonup_{Inv} D_n$, where*
$D_n = \underbrace{(\boldsymbol{N} \times (\boldsymbol{N} + \boldsymbol{N}) + \ldots + \boldsymbol{N} \times (\boldsymbol{N} + \boldsymbol{N})}_{n} + \boldsymbol{N}$,
be defined as follows:

$$
\begin{array}{c}
\underbrace{\boldsymbol{N}\times(\boldsymbol{N}+\boldsymbol{N})+\ldots+\boldsymbol{N}\times(\boldsymbol{N}+\boldsymbol{N})}_{n}+\boldsymbol{N}\\
\Big\downarrow\ \underbrace{id_{\boldsymbol{N}}\times t+\ldots+id_{\boldsymbol{N}}\times t}_{n}+id_{\boldsymbol{N}}\\
\underbrace{\boldsymbol{N}\times\boldsymbol{N}+\ldots+\boldsymbol{N}\times\boldsymbol{N}}_{n}+\boldsymbol{N}\\
\Big\downarrow\ \underbrace{p+\ldots+p}_{n}+id_{\boldsymbol{N}}\\
\underbrace{\boldsymbol{N}+\ldots+\boldsymbol{N}}_{n}+\boldsymbol{N}\\
\Big\downarrow\ t_{n-1}\\
\boldsymbol{N}\\
\Big\downarrow\ t_{n-1}^{-1}\\
\underbrace{\boldsymbol{N}+\ldots+\boldsymbol{N}}_{n}+\boldsymbol{N}\\
\Big\downarrow\ \underbrace{p^{-1}+\ldots+p^{-1}}_{n}+id_{\boldsymbol{N}}\\
\underbrace{\boldsymbol{N}\times\boldsymbol{N}+\ldots+\boldsymbol{N}\times\boldsymbol{N}}_{n}+\boldsymbol{N}\\
\Big\downarrow\ \underbrace{id_{\boldsymbol{N}}\times t^{-1}+\ldots+id_{\boldsymbol{N}}\times t^{-1}}_{n}+id_{\boldsymbol{N}}\\
\boldsymbol{N}
\end{array}
$$

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 5.2 *Let $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ be a closed PER, where $\vec{\mathcal{R}} = \bigotimes_{i=1}^{n} !R_i$, 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 there exists a unique i, $1 \le i \le n$, such that:*

- $X_i = X_k$,
- $\forall m.\ f_n^*(r,m) = (l,(i,!(r,m)))$,
  where f_n^* is defined as in Definition 5.1, and $!(r,m)$ denotes any element of $\boldsymbol{N}\times(\boldsymbol{N}+\boldsymbol{N})$, whose second projection is (r,m).

Proof. By Lemma 5.1, 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 by $\{f : \mathbf{N} \rightharpoonup_{Inv} \mathbf{N} \mid f(m) = m\}$. Then the partial involution g_j such that $g_j^*(r, m) = (r, m)$, where $g_j^* = t; g_j; t^{-1}$, is in $\mathcal{R}_j$. 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 $X_{k'}$ by 1, for $k' \neq k$, and X_k by $\{h : \mathbf{N} \rightharpoonup_{Inv} \mathbf{N} \mid h(m)\downarrow\}$. 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 $X_k = \{f : \mathbf{N} \rightharpoonup_{Inv} \mathbf{N} \mid f(m)\updownarrow f(m')\}$[1], and $X_{k'} = 1$, 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 $f\vec{g}(m')\uparrow$. $\Box$

Lemma 5.3 *Let $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ be a PER, where $\vec{\mathcal{R}} = \bigotimes_{i=1}^{n} !R_i$, 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 5.1.

Proof. 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 $\{h : \mathbf{N} \rightharpoonup_{Inv} \mathbf{N} \mid h(m)\uparrow\}$ and $\{h : \mathbf{N} \rightharpoonup_{Inv} \mathbf{N} \mid h(m)\downarrow\}$, 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. $\Box$

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 equivalences 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 5.3 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 5.2 *Let $\bigcap_{\vec{X}} (\mathcal{S} \multimap X_i) \multimap (\mathcal{R} \multimap X_k)$ be a closed PER.*

[1] $f(m)\updownarrow f(m')$ is the *equiconvergence* predicate, to be read as: $f(m)\downarrow \Leftrightarrow f(m')\downarrow$.

- *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* $Hom^t_{\mathbf{L}(\mathcal{U})}(\Lambda\vec{X}.\ \mathcal{S} \multimap \pi_i, \mathcal{R} \multimap \pi_k)$ *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 and 4, leaving Axiom 2, which is the most problematic, at the end.

Theorem 5.1 (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 5.2), one one can check that $f^*(r,(r,m)) = (l,(r,m))$. But then, instantiating X_k by $\{h : \mathbf{N} \rightharpoonup_{Inv} \mathbf{N} \mid h(m)\downarrow\}$, and $X_j = 1$, for all $j \neq k$, we get: $\emptyset \in \vec{\mathcal{S}}\multimap X_i$, but $f \bullet \emptyset = \emptyset \notin X_k$. $\square$

Theorem 5.2 (Linear Function Extensionality) *Let* $\bigcap_{\vec{X}}(\mathcal{S}\multimap X_k)\multimap(\mathcal{R}\multimap X_k)$ *be a closed PER. Then*

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

Proof. We define the inverse τ of $\Lambda\vec{X}.(\cdot)\vec{X}\multimap id_{X_k}$ to be the equivalence class of $\overline{\tau}$, where
$\overline{\tau} = t^{-1};t^{-1} + t^{-1};(t^{-1}+t^{-1}) + id_{\mathbf{N}+\mathbf{N}};\overline{\tau}^*;(t+t)+t;t+id_{\mathbf{N}};t$, and
$\overline{\tau}^* : ((\mathbf{N}+\mathbf{N}) + (\mathbf{N}+\mathbf{N})) + (\mathbf{N}+\mathbf{N}) \rightharpoonup ((\mathbf{N}+\mathbf{N}) + (\mathbf{N}+\mathbf{N})) + (\mathbf{N}+\mathbf{N})$ is the partial involution such that
$\forall n.\overline{\tau}^*(l,(l,(l,n))) = (r,(r,n))$ and
$\forall n.\overline{\tau}^*(l,(r,(l,n))) = (r,(l,n))$.
Then one can easily check that τ is the inverse of $\Lambda\vec{X}.(\cdot)\vec{X}\multimap id_{X_k}$. $\square$

We are left to show Axiom 2, i.e.:

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

where σ_i is the appropriate canonical morphism.

The function $\mathrm{case}_i\{F(\sigma_i)\}_{i=1,\dots,n}$ is easily shown to be surjective. Proving injectivity 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, the sole surjectivity of the function $\mathrm{case}_i\{F(\sigma_i)\}_{i=1,\dots,n}$ is at any rate sufficient to guarantee that the relevant morphisms have *a* decomposition, and therefore, if also finiteness condition 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 5.2), 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 Stataman Theorem, since the model is non-trivial, any relevant morphism denotes exactly one $\beta\eta$-normal form, and therefore the following isomorphism holds:

Theorem 5.3 (Weak Linearization of Head Occurrence) *Let $\bigcap_{\vec{X}} \vec{\mathcal{R}} \multimap X_k$ denote an ML-type. Then*

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

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

Proof. The proof follows using Theorems 1.1 and 5.4. $\square$

5.2 Proof of the Finiteness Axiom

We only prove a weak form of the Finiteness Axiom, 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.

In order to prove the finiteness result, we need to study an intermediate model, which contains also *approximant* terms of possibly infinite trees live. To this aim, we introduce the *Sierpinski PER*, and the corresponding hierarchy of *simple PERs* over it. This hierarchy gives rise to a model for the simply typed calculus $\lambda_{\perp,\top}$ of Section 1.

Definition 5.3 (Sierpinski PER) • *Let $n \in \boldsymbol{N}$. The* Sierpinski PER $\boldsymbol{O}_n$ *is the two-equivalence classes PER defined as follows:*

- $\perp_{\boldsymbol{O}_n} = \{f : \boldsymbol{N} \rightharpoonup_{Inv} \boldsymbol{N} \mid f(n)\uparrow\}$
- $\top_{\boldsymbol{O}_n} = \{f : \boldsymbol{N} \rightharpoonup_{Inv} \boldsymbol{N} \mid f(n)\downarrow\}$.

- *We define the hierarchy of* simple PERs *over the Sierpinski PER* $\boldsymbol{O}_n$ *as follows:*

$$(\mathrm{SimPer}_{\mathbf{O}_n} \ni)\ \mathcal{R} ::= \ \boldsymbol{O}_n \ \mid \mathcal{R} \rightarrow \mathcal{R} \ .$$

In what follows, we will omit the index n in denoting the Sierpinski PER $\mathbf{O}_n$, since, for all n, all these PERs are isomorphic.

The simple PERs over $\mathbf{O}$ yield a model of $\lambda_{\perp,\top}$. More precisely, this model is the CCC freely generated by the Sierpinski PER. We denote by $[\![\ ]\!]^{\mathbf{O}}$ the interpretation function. Notice that this model is trivially not faithful, i.e. its theory contains properly the $\beta\eta$-theory. In fact, since $\mathbf{O}$ has only a finite number of equivalence classes, then also $(\mathbf{O} \rightarrow \mathbf{O}) \rightarrow (\mathbf{O} \rightarrow \mathbf{O})$ has a finite number of equivalence classes, and therefore, some Church's numerals are identified in the model. More precisely, the model generated by $\mathbf{O}$ induces the *minimal* theory $=_m$ on simply typed λ-calculus defined by induction on types as follows:
$\perp =_m \perp \ : \ \iota \quad \wedge \quad \top =_m \top \ : \ \iota$
$M =_m N \ : \ T \rightarrow U$ iff $\forall P =_m Q. \ : T \ .MP =_m NQ \ : U.$
The model generated by the Sierpinski PER has a remarkable property, i.e. all partial involutions which inhabit a simple PER over $\mathbf{O}$ decompose as in Lemma 5.4 below. The proof of this lemma proceeds along the line of the proof of the Decomposition Theorem 3.2, except for the fact that here we deal directly with partial involutions, and not with equivalence classes. The proof of Lemma 5.4 is omitted.

Lemma 5.4 *Let $f \in dom(\mathcal{R}_1 \rightarrow \ldots \mathcal{R}_n \rightarrow \boldsymbol{O})$, where $\mathcal{R}_1 \rightarrow \ldots \rightarrow \mathcal{R}_n \rightarrow \boldsymbol{O} \in$*SimPer$_{\boldsymbol{O}}$*, and, for all $i = 1, \ldots, n$, $\mathcal{R}_i = \mathcal{S}_{i1} \rightarrow \ldots \rightarrow \mathcal{S}_{iq_i} \rightarrow \boldsymbol{O}$. Then*

- *either $f \in [\![\lambda\vec{x}.\ \perp]\!]^{\boldsymbol{O}}$*
- *or $f \in [\![\lambda\vec{x}.\top]\!]^{\boldsymbol{O}}$*
- *or $\exists! i \in \{1, \ldots, n\}$, $\exists h \in [\![\lambda\vec{z}\lambda\vec{x}.x_i(z_1\vec{x})\ldots(z_{q_i}\vec{x})]\!]$, and $\exists g_1, \ldots, g_{q_i}$, where $\forall j \in \{1, \ldots q_i\}$. $g_j \in dom(\vec{\mathcal{R}} \rightarrow S_{ij})$, such that*

$$f = h \bullet g_1 \bullet \ldots \bullet g_{q_i} \ .$$

Now we define *approximants* for partial involutions in the simple PERs over $\mathbf{O}$. These approximants 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 obtained by truncating at level k this tree, and by substituting the empty partial involution for the possibly erased subtrees. Formally:

Definition 5.4 (Approximants) *Let $f \in dom(\mathcal{R}) \in SimPer_{\boldsymbol{O}}$.*

- *We define the k-th tree, $t_k(f)$, of height at most $k + 1$, generated from f after k applications of the Decomposition Theorem by induction on k as follows:*

- $t_0(f)$ *is the tree of height* 1 *with root* f*;*
- *given the tree* $t_k(f)$ *of height at most* $k+1$*, the tree* $t_{k+1}(f)$ *is obtained from the tree* $t_k(f)$ *by expanding the possible leaves at level* $k+1$ *via the Decomposition Theorem.*

- *We define the k-th* approximant partial involution *of* f*,* $p_k(f) \in dom(\mathcal{R}) \in SimPer_{\boldsymbol{O}}$*, as the partial involution obtained from the tree* $t_k(f)$ *by substituting, if necessary, any partial involution at level* $k+1$ *by the empty partial involution.*

The supremum of the heights of the trees $t_k(f)$ yields a measure on partial involutions:

Definition 5.5 (Size Function) *Let* $f \in dom(\mathcal{R})$*, where* $\mathcal{R} \in SimPer_{\boldsymbol{O}}$*. We define*

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

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

Lemma 5.5 (Approximation) *Let* $f \in dom(\bigcap_{\vec{X}} \vec{\mathcal{R}} \to X_k)$*, where* $(\vec{\mathcal{R}} \to X_k)[\vec{\boldsymbol{O}}/\vec{X}]$ *is a simple PER over* $\boldsymbol{O}$*. 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* $\vec{g} \in dom(\vec{\mathcal{R}})$*,*

$$(\bigcup_{k\in\omega} p_k(f)) \bullet \vec{g} = \bigcup_{k\in\omega} (p_k(f) \bullet \vec{g}) \ .$$

Proof.(Sketch) The proof of ii) follows from the definition of application between partial involutions. The proof of i) follows from the fact that, for all $\vec{X}$, for all $\vec{g} \in \mathrm{dom}(\vec{\mathcal{R}})$, for all n, m,

$$(f \bullet \vec{g})(n) = m \iff ((\bigcup_{k\in\omega} p_k(f)) \bullet \vec{g})(n) = m \ .$$

The implication ($\Leftarrow$) is immediate by definition of approximants. In order to prove the converse, one can check, by induction on k, that, if $f \bullet \vec{g}(n) = m$ with a "thread" of length at most $2k$, then $p_k(f) \bullet \vec{g}(n) = m$. $\square$

Lemma 5.6 *Let* $f \in dom(\mathcal{R})$*, where* $\mathcal{R} \in SimPer_{\boldsymbol{O}}$*. Then*

$$p_k(f) \in [\![M_k]\!] \ ,$$

where M_k *is the term of* $\lambda_{\perp,\top}$ *whose (typed) Böhm tree is obtained from the tree* $p_k(f)$ *by relabeling the nodes, in such a way that the nodes appearing in the* $k+1$*-th level of* $p_k(f)$ *are relabeled by the constant* $\perp$ *(of the appropriate type), and the remaining nodes are labeled by the corresponding projections.*

Proof. By induction on k. $\square$

Theorem 5.4 (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 the ML-type $\forall \vec{X}.\vec{T} \to X_k$. Then $\mathcal{H}(f) < \infty$.*

Proof. By contradiction. Assume $\mathcal{H}(f) = \infty$. Then, $\forall Y.f \in dom(\mathcal{R}\ [Y \to Y/\vec{X}] \to (Y \to Y))$. Let $g_1 \in [\![\Lambda Y.S_{\alpha_{T_1}}]\!], \dots, g_n \in [\![\Lambda Y.S_{\alpha_{T_n}}]\!]$. Then $fg_1 \dots g_n \in dom(\bigcap_Y Y \to Y)$. By Lemma 5.5i), also $(\bigcup_{k\in\omega} p_k(f))g_1 \dots g_n \in dom(\bigcap_Y Y \to Y)$. By Lemma 5.5ii), $(\bigcup_{k\in\omega} p_k(f))g_1 \dots g_n = \bigcup_{k\in\omega} (p_k(f)g_1 \dots g_n)$. Then, by Lemma 5.6,
$(p_k(f)g_1 \dots g_n) \in [\![M_k]\!]^{\mathbf{O}}[\![S_{\alpha_{T_1}}]\!]^{\mathbf{O}} \dots [\![S_{\alpha_{T_n}}]\!]^{\mathbf{O}} =$ by the Typed Separability Theorem 1.2, $[\![\lambda x : \iota.\ \bot : \iota \to \iota]\!]^{\mathbf{O}} = \{h : \mathbf{N} \rightharpoonup \mathbf{N} \mid t; h; t^{-1}(r, n)\uparrow\}$.
Therefore, $((\bigcup_{k\in\omega} p_k(f))g_1 \dots g_n)(n)\uparrow$, and hence $(\bigcup_{k\in\omega} p_k(f))g_1 \dots g_n \notin \bigcap_Y Y \to Y \subseteq \{h : \mathbf{N} \rightharpoonup \mathbf{N} \mid t; h; t^{-1}(r, n)\downarrow\}$. Contradiction. $\square$

6 Final Remarks and Directions for Future Work

Here we give a list of remarks and interesting issues which still remain to be addressed (some of them are currently under investigation).

- 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 Nat $\to$ Nat, 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, all recursive functions, even all functions from natural numbers to natural numbers, can be encoded in the type Nat $\to$ Nat. 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 [HRR90].

- 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..

- Besides full completeness, *parametricity* is another "quality filter" for models of polymorphic functions. In particular, in [Plo93], a logic for linear parametric models has been suggested, in the line of [PA93]. It would be interesting to develop further this approach, and see whether this logic holds on our linear PER models. Longo's *genericity* ([LMS92]) can be viewed as a form of parametricity, in that it amounts to a *uniformity* property of polymorphic functions w.r.t. their input types. This issue and that of parametricity à la Reynolds will be investigated in [AL99a].

- Models of partial involutions are worthwhile investigating also for typed/untyped λ-calculi different from system F. E.g. strategies in the [AJM96] style, which are represented by partial involutions from Opponent moves to Player moves, should provide fully-complete models for simply typed λ-calculus with $\perp, \top$-base constants. In the untyped setting, partial involutions strategies could possibly provide fully-abstract models , alternative to those in [DFH99, KNO99].

- In the category $\mathrm{PER}_{\mathtt{PInv}}$, models of typed Böhm trees naturally arise (e.g. the model induced by the Sierpinski PER in Section 5.2). These are in particular models of the simply typed λ-calculus together with a *fixed point combinator*, as suggested by Alex Simpson. All these "infinite" calculi seem interesting by themselves, but have not yet been propertly investigated.

References

[Abr91] S.Abramsky. Domain Theory in logical form, *Annals of Pure and Applied Logic* **51**, 1991, 1–77.

[Abr96] S.Abramsky. Interaction, Combinators, and Complexity, Notes, Siena (Italy), 1996.

[Abr96a] S.Abramsky. Retracing some paths in Process Algebra, *Concur'96* Conf. Proc., 1996.

[Abr97] S.Abramsky. Axioms for Full Abstraction and Full Completeness, 1997, to appear.

[AHPS98] S.Abramsky, E.Haghverdi, P.Panangaden, P.Scott. Geometry of Interaction and Models of Combinatory Logic, 1998, to appear.

[AJ94] S.Abramsky, R.Jagadeesan. New foundations for the Geometry of Interaction, *Inf. and Comp.* **111**(1), 1994, 53–119.

[AJ94a] S.Abramsky, R.Jagadeesan. Games and Full Completeness for Multiplicative Linear Logic, *J. of Symbolic Logic* **59**(2), 1994, 543–574.

[AJM96] S.Abramsky, R.Jagadeesan, P.Malacaria. Full Abstraction for PCF, 1996, to appear.

[AL99] S.Abramsky, J.Longley. Realizability models based on hystory-free strategies, Draft paper, 1999.

[AL99a] S.Abramsky, M.Lenisa. On Full Completeness, Genericity, Parametric Polymorphism, in preparation.

[AM95] S.Abramsky, G.McCusker. Games and full abstraction for the lazy lambda-calculus, *LICS* Conf. Proc., 234–243, 1995.

[AM97] S.Abramsky, G.McCusker. Full abstraction for idealized Algol with passive expressions, 1997, to appear.

[AM97a] S.Abramsky, G.McCusker. Call-by-value games, *CSL'97* Conf. Proc., LNCS, 1997.

[AM97b] S.Abramsky, G.McCusker. Linearity, sharing and state, in P.O'Hearn and R.Tennent eds., *Algol-like Languages*, Birkhauser, 1997, 297–329.

[AM99] S.Abramsky, P.Mellies. *Concurrent Games and Full Completeness*, *LICS'99* Conf. Proc., 1999.

[AL91] 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.

[BC88] V.Breazu-Tannen, T.Coquand. Extensional models for polymorphism, *TCS* **59**, 1988, 85–114.

[BW96] N.Benton, P.Wadler. Linear Logic, Monads and the Lambda Calculus, *LICS'96* Conf. Proc., 1996.

[Bie95] G.Bierman. What is a categorical Model of Intuitionistic Linear Logic?, *TLCA'95* Conf. Proc., LNCS, 1995.

[CDHL82] M.Coppo, M.Dezani, F.Honsell, G.Longo. Extended Type Structures and Filter Lambda Models, *Logic Colloquium'82* Conf. Proc., G.Longo et al. eds., North Holland, 1983.

[Cro93] R.Crole, *Categories for Types*, Cambridge University Press, 1993.

[DFH99] P.Di Gianantonio, G.Franco, F.Honsell. Game Semantics for Untyped λ-calculus, *TLCA'99* Conf. Proc., LNCS, 1999.

[Gir72] J.Y.Girard. Interprétation functionelle et élimunation des coupures de l'arithmètique d'ordre supérieur, Thèse d'Etat, Université Paris VII, 1972.

[Gir89] J.Y.Girard. Towards a Geometry of Interaction.*Contemporary Mathematics* **92**, 1989, 69–108.

[Hug97] D.Hughes. Games and Definability for system F, *LICS'97* Conf. Proc., 1997.

[HRR90] J.Hyland, E.Robinson, G.Rosolini. Algebraic types in PER models, *MFPS* Conf. Proc., M.Main *et al.* eds, LNCS **442**, 1990, 333–350.

[HO96] M.Hyland, L.Ong. On full abstraction for PCF, *Information and Computation*, 1996, to appear.

[HY97] K.Honda, N.Yoshida. Game-theoretic analysis of call-by-value computation, *ICAlP'97* Conf. Proc., LNCS, 1997, 225–236.

[KNO99] A.Ker, H.Nickau, L.Ong. More Universal Game Models of Untyped λ-Calculus: The Böhm Tree Strikes Back, *CSL'99* Conf. Proc., LNCS, 1999.

[JSV96] A.Joyal, R.Street, D.Verity. Traced monoidal categories, *Math. Proc. Comb. Phil. Soc.* **119**, 1996, 447–468.

[Lai97] J.Laird. Full abstraction for functional languages with control, *LICS* Conf. Proc., 58–64, 1997.

[LMS92] G.Longo, K.Milsted, S.Soloviev. The Genericity Theorem and the Notion of Parametricity in the Polymorphic λ-calculus, *LICS'92* Conf. Proc., 1992.

[McC96] G.McCusker. Games and full abstraction for FPC, *LICS'96* Conf. Proc., 1996.

[Nic94] H.Nickau. Hereditarily sequential functionals, Proc. of the Symposium *Logical Foundations for Computer Science*, LNCS **813**, 1994.

[Plo93] G.Plotkin. Linear Parametricy, Notes, 1993.

[PA93] G.Plotkin, M.Abadi. A Logic for Parametric Polymorphism, *TLCA'93* Conf. Proc., LNCS, 1993.

[Sta88] Statman. λ-definable functionals and $\beta\eta$-conversion, *Arch. Math. Logik* **23**, 1983, 21–26.