Semantics of Linear Continuation-Passing in Call-by-Name

Masahito Hasegawa[1,2]

[1] Research Institute for Mathematical Sciences, Kyoto University
hassei@kurims.kyoto-u.ac.jp
[2] PRESTO, Japan Science and Technology Agency

Abstract. We propose a semantic framework for modelling the linear usage of continuations in typed call-by-name programming languages. On the semantic side, we introduce a construction for *categories of linear continuations*, which gives rise to cartesian closed categories with "linear classical disjunctions" from models of intuitionistic linear logic with sums. On the syntactic side, we give a simply typed call-by-name $\lambda\mu$-calculus in which the use of names (continuation variables) is restricted to be linear. Its semantic interpretation into a category of linear continuations then amounts to the call-by-name continuation-passing style (CPS) transformation into a linear lambda calculus with sum types. We show that our calculus is sound for this CPS semantics, hence for models given by the categories of linear continuations.

1 Introduction

1.1 Linearly Used Continuations

Recent work on *linearly used continuations* by Berdine, O'Hearn, Reddy and Thielecke [7,8] points out the advantage of looking at the linear usage of *continuations* in programming languages. They observe:

> ... in the many forms of control, continuations are used *linearly*. This is true for a wide range of effects, including procedure call and return, exceptions, `goto` statements, and coroutines.

They then propose linear type systems (based on a version of intuitionistic linear logic [13,2,3]) for capturing the linear usage of continuations, where the linear types are used for typing the target codes of continuation-passing style (CPS) transforms, rather than the source (ML or Scheme, for example) programs. Several "good" examples are shown to typecheck, while examples which duplicate continuations do not. An instance of such situations is found in a recent work on axiomatizing delimited continuations [19] where the linear usage of metacontinuations is crucial.

Motivated by Berdine et al.'s work, in a previous paper [14] we have developped a semantic framework for linearly used continuations (and more generally linearly used effects) in typed call-by-value (CBV) programming languages in

Y. Kameyama and P.J. Stuckey (Eds.): FLOPS 2004, LNCS 2998, pp. 229–243, 2004.

terms of models of linear type theories. In particular, the CBV CPS transformation is naturally derived as an instance of general monadic transformation into the linear lambda calculus in this framework, and we have shown that the CPS transformation enjoys good properties, most notably the full completeness ("no-junk property"). Further results including a fully abstract game semantics have been given by Laird [20]. Thus the semantic analysis on linear CPS in the CBV setting has been shown fruitful and successful to some extent.

The present paper proposes an analogous approach for linearly used continuations in *call-by-name* setting. Thus we first seek for the semantic construction which gives a model capturing the linearity of the usage of continuations from a model of linear type theory, and then extract the call-by-name CPS transformation into the linear lambda calculus from the construction. In this way we provide sound models of the call-by-name $\lambda\mu$-calculus [23] in which the use of names (continuation variables) is restricted to be linear. Proof theoretically, this restriction prohibits us to write programs (proofs) of many of "classical" types, because the disjunction type is used only linearly. We still have the excluded middle $\neg A \vee A$ (because it is isomorphic to $A \Rightarrow A$ in this world), but not the double-negation elimination $\neg\neg A \Rightarrow A$ (equivalently $\neg\neg\neg A \vee A$) in general. This means that the typing for linearly used continuations is placed somewhere between the intuitionistic and classical ones [1].

1.2 Semantic Construction: Categories of Linear Continuations

The central semantic construction in this work, though rather simple and possibly folklore among specialists, is that of *categories of linear continuations*, which can be considered as a generalization of two well-known constructions of cartesian closed categories:

1. The semantic counterpart of the *(call-by-name) double-negation translation from classical logic to intuitionistic logic*: we construct a cartesian closed category from a cartesian closed category with sums as the opposite of the Kleisli category of the "continuation monad" $((-) \to R) \to R$, also known as the *category of continuations* [16,26].
2. The semantic counterpart of the *Girard translation from intuitionistic logic to linear logic*: we construct a cartesian closed category from a model of linear logic as the co-Kleisli category of the comonad $!(-) = ((-) \to \bot) \multimap \bot$ (where we assume the presence of products) — equivalently as the opposite of the Kleisli category of the monad $?(-) = ((-) \multimap \bot) \to \bot$ (where we need sums).

The view of regarding modalities ! and ? as expressing "linearly used continuations" and "linearly defined continuations" has been emphasized in our previous work [15] (and also implicit in Filinski's work [12]), and it helps us to understand these two situations as instances of a single setting. Starting from a model of linear logic with sums, we construct a cartesian closed category as the opposite of the Kleisli category of the ?-like monad $T(-) = ((-) \multimap R) \to R$.

One technically interesting point is that monads of this form are in general *not* strong — they only have "strength with respect to !" [10]. Thus they seem less useful in the call-by-value setting (because a monad needs to be strong for interpreting a reasonable "notion of computation" in the sense of Moggi [22]). This also implies that the induced operators on objects for the "linear classical disjunction" does not form a premonoidal structure [24] — the object function $A \vee (-)$ does not extend to a functor.

1.3 Organization of This Paper

This article is organized as follows. In Sect. 2 we recall the semantics and syntax of the linear lambda calculus which serves as the target of our CPS transformation. Sect. 3 introduces the construction of categories of linear continuations. In Sect. 4 we consider the $\lambda\mu$-calculus with linear controls, and spell out the CPS transformation derived from the semantic construction of the last section. Sect. 5 concludes the paper. Appendices summarize the linear lambda calculus DILL and the notion of !-strong monads.

2 Preliminaries

2.1 Categorical Models of Linear Logic

We describe models of linear logic in terms of symmetric monoidal closed categories with additional structure – suitable comonad for modelling the modality "of course" !, and finite products/coproducts for modelling additives, and a dualising object (hence $*$-autonomous structure [4,5,25]) for modelling the duality of classical linear logic. For reference, we shall give a compact description of the comonads to be used below, due to Hyland and Schalk (which is equivalent to Bierman's detailed definition [9], and also to the formulation based on symmetric monoidal adjunctions [6,3]).

Definition 1 (linear exponential comonad [17]). *A symmetric monoidal comonad* $! = (!, \varepsilon, \delta, m_{A,B}, m_I)$ *on a symmetric monoidal category* $\mathcal{C}$ *is called a* linear exponential comonad *when the category of its coalgebras is a category of commutative comonoids – that is:*

- *there are specified monoidal natural transformations* $e_A : !A \to I$ *and* $d_A : !A \to !A \otimes !A$ *which form a commutative comonoid* $(!A, e_A, d_A)$ *in* $\mathcal{C}$ *and also are coalgebra morphisms from* $(!A, \delta_A)$ *to* (I, m_I) *and* $(!A \otimes !A, m_{!A,!A} \circ (\delta_A \otimes \delta_A))$ *respectively, and*
- *any coalgebra morphism from* $(!A, \delta_A)$ *to* $(!B, \delta_B)$ *is also a comonoid morphism from* $(!A, e_A, d_A)$ *to* $(!B, e_B, d_B)$.

2.2 Dual Intuitionistic Linear Logic

The target calculus we will make use of is the multiplicative exponential fragment of intuitionistic linear logic, formulated as a linear lambda calculus summarized

in Appendix. Our presentation is based on a dual-context type system for intuitionistic linear logic (called DILL) due to Barber and Plotkin [2,3]. In this formulation of the linear lambda calculus, a typing judgement takes the form $\Gamma\ ;\ \Delta \vdash M : \tau$ in which Γ represents an intuitionistic (or additive) context whereas Δ is a linear (multiplicative) context. It has been shown that symmetric monoidal closed categories with linear exponential comonad provide a sound and complete class of categorical models of DILL [3].

In the sequel, it turns out to be convenient to introduce syntax sugars for "intuitionistic" or "non-linear" function type

$$\begin{array}{rcl} \tau_1 \to \tau_2 & \equiv & !\tau_1 \multimap \tau_2 \\ \boldsymbol{\lambda} x^{\tau}.M & \equiv & \lambda y^{!\tau}.\mathsf{let}\ !x^{\tau}\ \mathsf{be}\ y\ \mathsf{in}\ M \\ M^{\tau_1 \to \tau_2} \mathbin{@} N^{\tau_1} & \equiv & M\,(!N) \end{array}$$

which enjoy the following typing derivations.

$$\frac{\Gamma, x:\tau_1\ ;\ \Delta \vdash M : \tau_2}{\Gamma\ ;\ \Delta \vdash \boldsymbol{\lambda} x^{\tau_1}.M : \tau_1 \to \tau_2} \qquad \frac{\Gamma\ ;\ \Delta \vdash M : \tau_1 \to \tau_2 \quad \Gamma\ ;\ \emptyset \vdash N : \tau_1}{\Gamma\ ;\ \Delta \vdash M \mathbin{@} N : \tau_2}$$

As one expects, the usual $\beta\eta$-equalities $(\boldsymbol{\lambda} x.M) \mathbin{@} N = M[N/x]$ and $\boldsymbol{\lambda} x.M \mathbin{@} x = M$ (with x not free in M) are easily provable from the axioms of DILL. (In fact it is possible to have them as primitives rather than derived constructs [7,8,15].)

In addition to the constructs of DILL, we need to deal with (additive) sum types. Here we employ the fairly standard syntax:

$$\frac{\Gamma\ ;\ \Delta \vdash M : 0}{\Gamma\ ;\ \Delta \vdash \mathsf{abort}_{\sigma}\, M : \sigma}\ (0\,\mathrm{E})$$

$$\frac{\Gamma\ ;\ \Delta \vdash M : \sigma}{\Gamma\ ;\ \Delta \vdash \mathsf{inl}_{\sigma,\tau}\, M : \sigma \oplus \tau}\ (\oplus\,\mathrm{I}_L) \qquad \frac{\Gamma\ ;\ \Delta \vdash N : \tau}{\Gamma\ ;\ \Delta \vdash \mathsf{inr}_{\sigma,\tau}\, N : \sigma \oplus \tau}\ (\oplus\,\mathrm{I}_R)$$

$$\frac{\Gamma\ ;\ \Delta_1 \vdash L : \sigma \oplus \tau \quad \Gamma\ ;\ \Delta_2, x:\sigma \vdash M : \theta \quad \Gamma\ ;\ \Delta_2, y:\tau \vdash N : \theta}{\Gamma\ ;\ \Delta_1 \sharp \Delta_2 \vdash \mathsf{case}\ L\ \mathsf{of}\ \mathsf{inl}\, x^{\sigma} \mapsto M \parallel \mathsf{inr}\, y^{\tau} \mapsto N : \theta}\ (\oplus\,\mathrm{E})$$

3 Categories of Linear Continuations

Let $\mathcal{C}$ be a symmetric monoidal closed category with a linear exponential comonad ! and finite coproducts (we write 0 for an initial object and $\oplus$ for binary coproducts). Fix an object R, and define a category $R^{\mathcal{C}}$ as follows: $R^{\mathcal{C}}$'s objects are the same as those of $\mathcal{C}$, and arrows are given by

$$R^{\mathcal{C}}(A, B) \ = \ \mathcal{C}(!(A \multimap R), B \multimap R).$$

The identities and compositions in $R^{\mathcal{C}}$ are inherited from the co-Kleisli category $\mathcal{C}_!$ of the comonad ! (so, up to equivalence, $R^{\mathcal{C}}$ can be considered as the full subcategory of $\mathcal{C}_!$ whose objects are of the form $A \multimap R$).

As easily seen, $R^{\mathcal{C}}(A,B) \simeq \mathcal{C}(B, !(A \multimap R) \multimap R)$, thus $R^{\mathcal{C}}$ is isomorphic to the opposite of the Kleisli category of the monad $TX = !(X \multimap R) \multimap R$ on $\mathcal{C}$. Note that this monad is *not* necessarily strong – but it is strong with respect to ! (i.e., has a restricted form of strength $!A \otimes TX \to T(!A \otimes X)$) – see Appendix for the definition of !-strong monads. This notion is introduced by Blute et al. [10] for axiomatising the exponential "why not" ?. A !-strong monad may not be strong, though it induces a strong monad on the co-Kleisli category $\mathcal{C}_!$ [14].

Proposition 1. *The monad $TX = !(X \multimap R) \multimap R$ on $\mathcal{C}$ is !-strong.*

In terms of DILL, the !-strength is represented as follows.

$$a : A \ ; \ m : (X \multimap R) \to R \vdash \boldsymbol{\lambda} k^{(!A \otimes X) \multimap R}.m \, @ \, (\lambda x^X.k \, (!a \otimes x)) \\ : ((!A \otimes X) \multimap R) \to R$$

We shall note the non-linear use of the variable $a : A$.

Note that the exponential $?(-) = !((-) \multimap \bot) \multimap \bot$ is a particular instance of this — see Example 2 below. Another typical example of !-strong (but not necessarily strong) monads is the exception monad $X \mapsto X \oplus E$.

3.1 Cartesian Closure

A category of linear continuations has sufficient structures for modelling the simply typed lambda calculus [21]:

Proposition 2. *$R^{\mathcal{C}}$ is a cartesian closed category.*

Proof: Define

$$\top \ = \ 0 \quad A \wedge B \ = \ A \oplus B \quad A \Rightarrow B \ = \ !(A \multimap R) \otimes B$$

We shall see that $\top$ is a terminal object, $A \wedge B$ is a binary product of A and B, and $A \Rightarrow B$ is an exponential of B by A in $R^{\mathcal{C}}$.

$$\begin{aligned} R^{\mathcal{C}}(A, \top) &= \mathcal{C}(!(A \multimap R), 0 \multimap R) \\ &\simeq \mathcal{C}(0, !(A \multimap R) \multimap R) \\ &\simeq 1 \end{aligned}$$

$$\begin{aligned} R^{\mathcal{C}}(A, B \wedge C) &= \mathcal{C}(!(A \multimap R), (B \oplus C) \multimap R) \\ &\simeq \mathcal{C}(B \oplus C, !(A \multimap R) \multimap R) \\ &\simeq \mathcal{C}(B, !(A \multimap R) \multimap R) \times \mathcal{C}(C, !(A \multimap R) \multimap R) \\ &\simeq \mathcal{C}(!(A \multimap R), B \multimap R) \times \mathcal{C}(!(A \multimap R), C \multimap R) \\ &= R^{\mathcal{C}}(A, B) \times R^{\mathcal{C}}(A, C) \end{aligned}$$

$$\begin{aligned} R^{\mathcal{C}}(A \wedge B, C) &= \mathcal{C}(!((A \oplus B) \multimap R), C \multimap R) \\ &\simeq \mathcal{C}(!(A \multimap R) \otimes !(B \multimap R), C \multimap R) \\ &\simeq \mathcal{C}(!(A \multimap R), (!(B \multimap R) \otimes C) \multimap R) \\ &= R^{\mathcal{C}}(A, B \Rightarrow C) \end{aligned}$$

Here we use an isomorphism $!((A \oplus B) \multimap R) \simeq !(A \multimap R) \otimes !(B \multimap R)$ which can be thought as an instance of the "Seely isomorphism" $!(X \& Y) \simeq !X \otimes !Y$ [25,9] as we have a product diagram $A \multimap R \xleftarrow{\mathsf{inl} \multimap R} (A \oplus B) \multimap R \xrightarrow{\mathsf{inr} \multimap R} B \multimap R$. Since a linear exponential comonad ! arises from a symmetric monoidal adjunction from a category with finite products [9,6,3], it follows that ! sends a product of X and Y (if exists) to $!X \otimes !Y$ up to coherent isomorphism. □

3.2 Disjunctions

It is natural to define the (linear) disjunctions on $R^{\mathcal{C}}$ as

$$\bot \;=\; I \qquad A \vee B \;=\; A \otimes B$$

which satisfy the isomorphisms one would expect for "classical" disjunctions:

Proposition 3. *The following isomorphisms exist:*

$$\begin{array}{rcl} A \Rightarrow B & \simeq & (A \Rightarrow \bot) \vee B \\ A \Rightarrow (B \vee C) & \simeq & (A \Rightarrow B) \vee C \end{array}$$

However, they are *not* even premonoidal (because the monad T is *not* strong). The functor $A \otimes (-)$ on $\mathcal{C}$ does not give rise to a functor on $R^{\mathcal{C}}$.

We also note that these linear disjunctions do not give weak coproducts in general. For instance $\bot$ is not a weak initial object:

$$R^{\mathcal{C}}(\bot, X) \;=\; \mathcal{C}(!(I \multimap R), X \multimap R) \;\simeq\; \mathcal{C}(X, !R \multimap R) \;=\; \mathcal{C}(X, R \to R)$$

Hence we can define the canonical map from $\bot$ to only objects of the form $!X$.

3.3 Examples

As mentioned in the introduction, the categories of linear continuations subsume two well-known constructions of cartesian closed categories: one for the call-by-name double-negation translation from classical logic to intuitionistic logic, and the other for (the dual of) the Girard translation from intuitionistic logic to linear logic. For the former, it suffices to simply trivialize the linearity. For the latter, we let the response object R be the dualising object (linear falsity type) $\perp\!\!\!\perp$.[3]

Example 1 (Categories of continuations). Let $\mathcal{C}$ be a cartesian closed category with finite coproducts and an object R. By taking the identity comonad as the linear exponential comonad, we have a sufficient structure for constructing a category $R^{\mathcal{C}}$ of (linear) continuations. Its objects are the same as $\mathcal{C}$, with $R^{\mathcal{C}}(A, B) = (R^A, R^B)$, together with the terminal object 0, binary product $A \oplus B$ and exponential $R^A \times B$. This is exactly the category of continuations [16,26]. Note that, in this case, the monad T is the standard continuation monad and is strong, hence the classical disjunction is premonoidal.

[3] This should not be confused with the classical falsity $\bot$ introduced in the last section. In this paper we use $\bot$ and $\perp\!\!\!\perp$ for the classical falsity and linear falsity (dualising object) respectively.

Example 2 (Girard translation). Suppose that $\mathcal{C}$ is $*$-autonomous [4], thus has a dualising object $\bot$ and can model classical linear logic [25,5]. We note that its opposite $\mathcal{C}^{\mathrm{op}}$ is also $*$-autonomous, with a linear exponential comonad $?X = !(X \multimap \bot) \multimap \bot$. By letting R be $\bot$, we have

$$\bot^{\mathcal{C}^{\mathrm{op}}}(A,B) \simeq \mathcal{C}^{\mathrm{op}}(?(A \multimap \bot), B \multimap \bot) \simeq \mathcal{C}^{\mathrm{op}}(B, !A) = \mathcal{C}(!A, B) = \mathcal{C}_!(A,B).$$

Thus the derivation of the cartesian closure of $\bot^{\mathcal{C}^{\mathrm{op}}}$ is exactly the well-known "decomposition" $A \Rightarrow B = !A \multimap B$ (given a symmetric monoidal closed category $\mathcal{C}$ with a linear exponential comonad ! and finite products, the co-Kleisli category $\mathcal{C}_!$ is cartesian closed). Sec.4.5 gives a syntactic interpretation of this observation.

Of course, there are many categories of linear continuations which do not fall into these two extreme cases. For instance:

Example 3. Let $\mathcal{C}$ be the category of ω-cpo's (with bottom) and strict continuous functions, ! be the lifting, and R be any object. The category $R^{\mathcal{C}}$ has the same objects as $\mathcal{C}$, but its morphism from A to B is a (possibly non-strict) continuous function between the strict-function spaces $A \multimap R$ and $B \multimap R$.

3.4 Discussion: Towards Direct-Style Models

Ideally, we would like to find a direct axiomatization of the categories of linear continuations as cartesian closed categories with extra structure — as neatly demonstrated in the non-linear case by Selinger as control categories and its structural theorem with respect to categories of continuations [26]. The main difficulty in our linear case is that the linear classical disjunction is no longer premonoidal, and we do not know how to axiomatize them. So there seems no obvious way to adopt Selinger's work to define a notion of "linear control categories".

But there still are some hope: we can consider the category $R^{\mathcal{C}^{\mathsf{lin}}}$ defined by $R^{\mathcal{C}^{\mathsf{lin}}}(A,B) = \mathcal{C}(A \multimap R, B \multimap R)$, which can be regarded as a lluf subcategory of linear maps in $R^{\mathcal{C}}$ (provided the counit is epi). The disjunctions do form a symmetric premonoidal structure on $R^{\mathcal{C}^{\mathsf{lin}}}$. Moreover, the category $R^{\mathcal{C}^{\mathsf{lin}}}$ has finite products and the inclusion from $R^{\mathcal{C}^{\mathsf{lin}}}$ to $R^{\mathcal{C}}$ preserves them.

So it might be natural to formulate the structure directly as a cartesian closed category together with a lluf subcategory with finite products (preserved by the inclusion) and distributive symmetric premonoidal products (but not with codiagonals as required for control categories), satisfying certain coherence conditions — e.g. on the isomorphism $A \Rightarrow (B \vee C) \simeq (A \Rightarrow B) \vee C$.

4 The $\lambda\mu$-Calculus with Linear Controls

We formulate the calculus for expressing "linearly used continuations" as a constrained $\lambda\mu$-calculus where names (continuation variables) are used and bound just linearly. Here we make use of the syntax for the simply typed $\lambda\mu$-calculus with disjunctions [26], together with a typing system which represents this linearity constraint on names.

4.1 The Calculus

Types and Terms

$$\begin{array}{rcl} \sigma & ::= & b \mid \sigma \Rightarrow \sigma \mid \top \mid \sigma \wedge \sigma \mid \bot \mid \sigma \vee \sigma \\ M & ::= & x \mid \lambda x^{\sigma}.M \mid M\,M \mid * \mid \langle M, M \rangle \mid \pi_1\,M \mid \pi_2\,M \mid \\ & & [\alpha]M \mid \mu\alpha^{\sigma}.M \mid [\alpha,\alpha]M \mid \mu(\alpha^{\sigma},\alpha^{\sigma}).M \end{array}$$

where b ranges over base types.

Typing

$$\frac{}{\Gamma \vdash x : \sigma \mid \emptyset}\ x : \sigma \in \Gamma$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau \mid \Delta}{\Gamma \vdash \lambda x^{\sigma}.M : \sigma \Rightarrow \tau \mid \Delta} \qquad \frac{\Gamma \vdash M : \sigma \Rightarrow \tau \mid \Delta \quad \Gamma \vdash N : \sigma \mid \emptyset}{\Gamma \vdash M\,N : \tau \mid \Delta}$$

$$\frac{}{\Gamma \vdash * : \top \mid \emptyset} \qquad \frac{\Gamma \vdash M : \sigma \mid \Delta_1 \quad \Gamma \vdash N : \tau \mid \Delta_2}{\Gamma \vdash \langle M, N \rangle : \sigma \wedge \tau \mid \Delta_1 \sharp \Delta_2}$$

$$\frac{\Gamma \vdash M : \sigma \wedge \tau \mid \emptyset}{\Gamma \vdash \pi_1\,M : \sigma \mid \emptyset} \qquad \frac{\Gamma \vdash M : \sigma \wedge \tau \mid \emptyset}{\Gamma \vdash \pi_2\,M : \tau \mid \emptyset}$$

$$\frac{\Gamma \vdash M : \sigma \mid \Delta}{\Gamma \vdash [\alpha]M : \bot \mid \{\alpha : \sigma\} \sharp \Delta} \qquad \frac{\Gamma \vdash M : \bot \mid \alpha : \sigma, \Delta}{\Gamma \vdash \mu\alpha^{\sigma}.M : \sigma \mid \Delta}$$

$$\frac{\Gamma \vdash M : \sigma \vee \tau \mid \Delta}{\Gamma \vdash [\alpha,\beta]M : \bot \mid \{\alpha : \sigma, \beta : \tau\} \sharp \Delta} \qquad \frac{\Gamma \vdash M : \bot \mid \alpha : \sigma, \beta : \tau, \Delta}{\Gamma \vdash \mu(\alpha^{\sigma},\beta^{\tau}).M : \sigma \vee \tau \mid \Delta}$$

where $\Delta_1 \sharp \Delta_2$ represents one of possible merges of Δ_1 and Δ_2 as finite lists. We assume that, when we introduce $\Delta_1 \sharp \Delta_2$, there is no name occurring both in Δ_1 and in Δ_2. We write $\emptyset$ for the empty context. Note that names cannot be free in the argument of a function application.

Example 4. The reader is invited to verify that

$$\mu(\kappa^{\sigma \Rightarrow \bot}, \alpha^{\sigma}).[\kappa](\lambda x^{\sigma}.[\alpha]x) : (\sigma \Rightarrow \bot) \vee \sigma$$

typechecks, while

$$\lambda m^{(\sigma \Rightarrow \bot) \Rightarrow \bot}.\mu\alpha^{\sigma}.m\,(\lambda x^{\sigma}.[\alpha]x) : ((\sigma \Rightarrow \bot) \Rightarrow \bot) \Rightarrow \sigma$$

does not — the name α occurs in the argument of a function m which may duplicate or discard α.

Example 5. The disjunction $(-) \vee \tau$ fails to be functorial: given a term $M : \sigma \Rightarrow \sigma'$, one might expect to have

$$M \vee \tau \ = \ \lambda z^{\sigma \vee \tau}.\mu(\alpha'^{\sigma'}, \beta^{\tau}).[\alpha'](M\,(\mu\alpha^{\sigma}.[\alpha,\beta]z)) \ : \ \sigma \vee \tau \Rightarrow \sigma' \vee \tau$$

which is illegal because M may not use β linearly.

4.2 The Call-by-Name CPS Interpretation

The cartesian closed structure of the category of linear continuations motivates the following interpretation of the arrow type

$$[\![\sigma \Rightarrow \tau]\!] \;\simeq\; !([\![\sigma]\!] \multimap R) \otimes [\![\tau]\!]$$

which leads us to interpret a typing judgement as follows.

$$\begin{array}{l}[\![x_1 : \sigma_1, \ldots, x_m : \sigma_m \vdash M : \sigma \mid \alpha_1 : \tau_1, \ldots, \alpha_n : \tau_n]\!] : \\ \quad !([\![\sigma_1]\!] \multimap R) \otimes \ldots \otimes !([\![\sigma_m]\!] \multimap R) \otimes [\![\tau_1]\!] \otimes \ldots \otimes [\![\tau_n]\!] \longrightarrow [\![\sigma]\!] \multimap R\end{array}$$

Rather than describing the translation in terms of categorical combinators, below we shall give it as a CPS transform into DILL with sums. For types we have

$$\begin{array}{ccc} b^\circ = b & \top^\circ = 0 & \bot^\circ = I \\ (\sigma \Rightarrow \tau)^\circ = !(\sigma^\circ \multimap R) \otimes \tau^\circ & (\sigma \wedge \tau)^\circ = \sigma^\circ \oplus \tau^\circ & (\sigma \vee \tau)^\circ = \sigma^\circ \otimes \tau^\circ \end{array}$$

and for terms

$$\begin{array}{rcl} x^\circ & = & x \\ (\lambda x^\sigma.M^\tau)^\circ & = & \lambda(!x^{\sigma^\circ \multimap R} \otimes k^{\tau^\circ}).M^\circ\, k \\ (M^{\sigma \Rightarrow \tau}\, N^\sigma)^\circ & = & \lambda k^{\tau^\circ}.M^\circ\, (!N^\circ \otimes k) \\ *^\circ & = & \lambda k^0.\mathsf{abort}_R\, k \\ \langle M^\sigma, N^\tau \rangle^\circ & = & \lambda k^{\sigma^\circ \oplus \tau^\circ}.\mathsf{case}\ k\ \mathsf{of}\ \mathsf{inl}(x) \mapsto M^\circ\, x \parallel \mathsf{inr}(y) \mapsto N^\circ\, y \\ (\pi_1\, M^{\sigma \wedge \tau})^\circ & = & \lambda k^{\sigma^\circ}.M^\circ\, (\mathsf{inl}\, k) \\ (\pi_2\, M^{\sigma \wedge \tau})^\circ & = & \lambda k^{\tau^\circ}.M^\circ\, (\mathsf{inr}\, k) \\ ([\alpha]M)^\circ & = & \lambda *^I .M^\circ\, \alpha \\ (\mu\alpha^\sigma.M)^\circ & = & \lambda \alpha^{\sigma^\circ}.M^\circ\, * \\ ([\alpha, \beta]M)^\circ & = & \lambda *^I .M^\circ\, (\alpha \otimes \beta) \\ (\mu(\alpha^\sigma, \beta^\tau).M)^\circ & = & \lambda(\alpha^{\sigma^\circ} \otimes \beta^{\tau^\circ}).M^\circ\, * \end{array}$$

Also we use some "pattern matching binding", e.g. $\lambda(x^\sigma \otimes y^\tau).M$ as a shorthand for $\lambda z^{\sigma \otimes \tau}.\mathsf{let}\ x^\sigma \otimes y^\tau\ \mathsf{be}\ z\ \mathsf{in}\ M$, and $\lambda *^I .M$ for $\lambda z^I.\mathsf{let}\ *\ \mathsf{be}\ z\ \mathsf{in}\ M$. A typing judgement

$$x_1 : \sigma_1, \ldots, x_m : \sigma_m \vdash M : \sigma \mid \alpha_1 : \tau_1, \ldots, \alpha_n : \tau_n$$

is sent to

$$x_1 : \sigma_1^\circ \multimap R, \ldots, x_m : \sigma_m^\circ \multimap R\ ;\ \alpha_1 : \tau_1^\circ, \ldots, \alpha_n : \tau_n^\circ \vdash M^\circ : \sigma^\circ \multimap R.$$

Note that, if we ignore all the linearity information, this is precisely the same as the call-by-name CPS transformation of Selinger [26].

Example 6. The well-typed term

$$\mu(\kappa^{\sigma \Rightarrow \bot}, \alpha^\sigma).[\kappa](\lambda x^\sigma.[\alpha]x) : (\sigma \Rightarrow \bot) \vee \sigma$$

is sent to

$$\lambda((!x^{\sigma^\circ \multimap R} \otimes *^I) \otimes \alpha^{\sigma^\circ}).x\, \alpha : ((!(\sigma^\circ \multimap R) \otimes I) \otimes \sigma^\circ) \multimap R$$

which essentially agrees with the transform of the identity function

$$(\lambda x^\sigma.x)^\circ = \lambda(!x^{\sigma^\circ \multimap R} \otimes k^{\sigma^\circ}).x\, k : (!(\sigma^\circ \multimap R) \otimes \sigma^\circ) \multimap R.$$

4.3 Axioms and Soundness

It is routine to see that this CPS transform validates the $\beta\eta$-equalities of the simply typed lambda calculus with products (this is a consequence of the cartesian closedness of the categories of linear continuations). In fact all axioms for the call-by-name $\lambda\mu$-calculus (as given by Selinger [26]) are valid, except the non-linear axiom $(\beta_\perp) : [\alpha^\perp]M = M$ which is replaced by its linear variant $\mu\alpha^\perp.C[[\alpha]M] = C[M]$ below.

Axioms

$$
\begin{array}{lll}
(\beta_\Rightarrow) & (\lambda x.M)\, N & = M[N/x] \\
(\eta_\Rightarrow) & \lambda x.M\, x & = M[N/x] \quad (x \notin FV(M)) \\
(\beta_\wedge) & \pi_i \langle M_1, M_2 \rangle & = M_i \\
(\eta_\wedge) & \langle \pi_1\, M, \pi_2\, M \rangle & = M \\
(\eta_\top) & * & = M \\
 & & \\
(\beta_\mu) & [\alpha'](\mu\alpha.M) & = M[\alpha'/\alpha] \\
(\eta_\mu) & \mu\alpha.[\alpha]M & = M \\
(\beta_\vee) & [\alpha', \beta'](\mu(\alpha, \beta).M) & = M[\alpha'/\alpha, \beta'/\beta] \\
(\eta_\vee) & \mu(\alpha, \beta).[\alpha, \beta]M & = M \\
(\beta_\perp) & \mu\alpha^\perp.C[[\alpha]M] & = C[M] \\
(\zeta_\Rightarrow) & (\mu\alpha^{\sigma\Rightarrow\tau}.C[[\alpha]M])\, N & = \mu\beta^\tau.C[[\beta](M\, N)] \\
(\zeta_\wedge) & \pi_i\, (\mu\alpha^{\sigma_1\wedge\sigma_2}.C[[\alpha]M]) & = \mu\beta^{\sigma_i}.C[[\beta](\pi_i\, M)] \\
(\zeta_\vee) & [\alpha, \beta](\mu\delta.C[[\delta]M]) & = C[[\alpha, \beta]M]
\end{array}
$$

Theorem 1. *The CPS translation is sound: $\Gamma \vdash M = N : \sigma \mid \Delta$ implies $\Gamma^\circ \multimap R\ ;\ \Delta^\circ \vdash M^\circ = N^\circ : \sigma^\circ \multimap R$.*

Corollary 1. *The interpretation of the calculus into any symmetric monoidal closed category with linear exponential comonad and sums is sound: $\Gamma \vdash M = N : \sigma \mid \Delta$ implies $[\![\Gamma \vdash M : \sigma \mid \Delta]\!] = [\![\Gamma \vdash N : \sigma \mid \Delta]\!]$.*

4.4 Discussion: Completeness

We conjecture that the equational theory of the calculus given above is not just sound but also complete for the CPS semantics, hence also the models given by categories of linear continuations. (This should be the case, as the usual (non-linear) $\lambda\mu$-calculus is likely to be a conservative extension of our calculus and its term model gives rise to a category of continuations [16,26], hence a complete model.)

The main problem for showing the completeness, however, is that the types and terms are not sufficiently rich to give rise to a category of linear continuations as the term model; it is not very clear how to derive a base category and the response object R, as well as the linear exponential comonad !. This also suggests that the calculus may not be sufficient for explaining the nature of linear continuation-passing — there seem some room for further improvement.

4.5 Girard Translation as a CPS Transformation

In Example 2, we have noted that the standard construction of cartesian closed categories from models of linear logic can be regarded as an instance of our construction of categories of linear continuations. This means that Girard translation from intuitionistic logic into the (classical) linear logic can also be derived as an instance of our CPS transformation and extends to our $\lambda\mu$-calculus — in this reading, it really is a simply typed lambda calculus enriched with par's. The derived translation (obtained by taking the opposite of the term model of the linear lambda calculus and then letting R be $\bot\!\!\!\bot$) is straightforwardly described as follows, using the linear lambda calculus DCLL [15] as the target. For types: $b^\circ = b$, $\top^\circ = \top$, $(\sigma \wedge \tau)^\circ = \sigma^\circ \,\&\, \tau^\circ$, $(\sigma \Rightarrow \tau)^\circ = \sigma^\circ \to \tau^\circ$, $\bot^\circ = \bot\!\!\!\bot$ and $(\sigma \vee \tau)^\circ = \sigma^\circ \parr \tau^\circ = (\sigma^\circ \multimap \bot\!\!\!\bot) \multimap (\tau^\circ \multimap \bot\!\!\!\bot) \multimap \bot\!\!\!\bot$. For terms, we have $x^\circ = x$, $(\lambda x.M)^\circ = \boldsymbol{\lambda} x.M^\circ$, $(M\,N)^\circ = M^\circ \mathbin{@} N^\circ$, $*^\circ = \langle\,\rangle$, $\langle M, N\rangle^\circ = \langle M^\circ, N^\circ\rangle$, $(\pi_1\,M)^\circ = \mathsf{fst}\,M^\circ$, $(\pi_2\,M)^\circ = \mathsf{snd}\,M^\circ$, $([\alpha]M)^\circ = \alpha\,M^\circ$, $(\mu\alpha.M)^\circ = \mathsf{C}\,(\lambda\alpha.M^\circ)$, $([\alpha,\beta]M)^\circ = M^\circ\,\alpha\,\beta$ and $(\mu(\alpha,\beta).M)^\circ = \lambda\alpha.\lambda\beta.M^\circ$, where the combinator $\mathsf{C}_\sigma : ((\sigma \multimap \bot\!\!\!\bot) \multimap \bot\!\!\!\bot) \multimap \sigma$ expresses the isomorphism from $(\sigma \multimap \bot\!\!\!\bot) \multimap \bot\!\!\!\bot$ to σ. A judgement $\Gamma \vdash M : \sigma \mid \Delta$ is sent to $\Gamma^\circ\ ;\ \Delta^\circ \multimap \bot\!\!\!\bot \vdash M^\circ : \sigma^\circ$. The soundness of this translation is just an instance of Corollary 1.

5 Concluding Remarks

In this paper we proposed a semantic approach for linearly used continuations in call-by-name setting, and developped a relevant semantic construction and also a syntactic calculus for such linear controls, together with its CPS transformation.

However, we must say that this work is still premature — at least not as successful as the case of the call-by-value setting for now — and there remain many important issues to be addressed. Among them, we already mentioned two major problems (which are related each other): (1) the lack of direct-style models, and (2) the completeness problem. This situation is really frustrating, compared with the call-by-value setting for which we have satisfactory answers for them [14]. The non-premonoidal disjunction in particular rises serious obstacles for a direct/complete axiomatization.

Another issue which we wish to understand is how the *Filinski duality* [11,26,27] between call-by-name and call-by-value languages with first-class continuations can be related to our approach on linear controls. To sketch the overall picture, below we shall list up the interpretations of the function type $A \Rightarrow B$ in the possible combinations of linearity and non-linearity:

	call-by-name	call-by-value
non-linear lang. with non-linear control	$(A \to R) \times B$	$(B \to R) \to (A \to R)$
non-linear lang. with linear control	$!(A \multimap R) \otimes B$	$(B \to R) \multimap (A \to R)$
linear lang. with non-linear control	$(A \to R) \otimes !B$	$(B \multimap R) \to (A \multimap R)$
linear lang. with linear control	$(A \multimap R) \otimes B$	$(B \multimap R) \multimap (A \multimap R)$

The top row was studied in detail by Selinger [26]. The bottom row (purely linear setting) is more or less trivial. We are most interested in the second row, but also

in the third, as there seem to exist certain dualities between the call-by-name non-linear language with linear control and the call-by-value linear language with non-linear control

$$(!(A \multimap R) \otimes B) \multimap R \ \simeq \ (A \multimap R) \to (B \multimap R),$$

and also between the call-by-value non-linear language with linear control and the call-by-name linear language with non-linear control

$$((A \to R) \otimes !B) \to R \ \simeq \ (A \to R) \multimap (B \to R).$$

The practical interest on the third row may be rather limited, but we still hope that this duality-based view provides some good insights on the nature of linear controls in call-by-name and call-by-value settings, potentially with some tie-up with other computational features such as recursion and iteration [18].

For more practical side, we have not yet demonstrated the usefulness of this approach for reasoning about call-by-name programs with first-class (linear) controls. Perhaps this also reflects our lack of experience with call-by-name programming languages with first-class control primitives whose practical advantage is, we believe, yet to be understood.

Acknowledgements I thank Oliver Danvy for asking me how linear CPS for call-by-name can be semantically understood, and anonymous reviewers for helpful comments.

References

1. Ariola, Z.M. and Herbelin, H (2003) Minimal classical logic and control operators. In *Proc. 30th International Colloquium on Automata, Languages and Programming (ICALP2003)*, Springer Lecture Notes in Comput. Sci. **2719**, pp. 871–885.
2. Barber, A. (1997) *Linear Type Theories, Semantics and Action Calculi*. PhD Thesis ECS-LFCS-97-371, Univ. Edinburgh.
3. Barber, A. and Plotkin, G. (1997) Dual intuitionistic linear logic. Manuscript. An earlier version available as Technical Report ECS-LFCS-96-347, Univ. Edinburgh.
4. Barr, M. (1979) $*$*-Autonomous Categories*. Springer Lecture Notes in Math. **752**.
5. Barr, M. (1991) $*$-autonomous categories and linear logic. *Math. Struct. Comp. Sci.* **1**, 159–178.
6. Benton, N. (1995) A mixed linear non-linear logic: proofs, terms and models (extended abstract). In *8th International Workshop on Computer Science Logic (CSL'94), Selected Papers*, Springer Lecture Notes in Comput. Sci. **933**, pp. 121–135.
7. Berdine, J., O'Hearn, P.W., Reddy, U.S. and Thielecke, H. (2001) Linearly used continuations. In *Proc. 3rd ACM SIGPLAN Workshop on Continuations (CW'01)*, Technical Report No. 545, Computer Science Dept., Indiana Univ., pp. 47–54.
8. Berdine, J., O'Hearn, P.W., Reddy, U.S. and Thielecke, H. (2002) Linear continuation-passing. *Higher-Order and Symbolic Computation* **15**(2/3), 181–203.
9. Bierman, G.M. (1995) What is a categorical model of intuitionistic linear logic? In *Proc. 2nd International Conference on Typed Lambda Calculi and Applications (TLCA'95)*, Springer Lecture Notes in Comput. Sci. **902**, pp. 78–93.

10. Blute, R., Cockett, J.R.B. and Seely, R.A.G. (1996) ! and ? - storage as tensorial strength. *Mathematical Structures in Computer Science* **6**(4), 313–351.
11. Filinski, A. (1989) Declarative continuations: an investigation of duality in programming language semantics. In *Proc. Category Theory and Computer Science*, Springer Lecture Notes in Comput. Sci. **389**, pp. 224–249.
12. Filinski, A. (1992) Linear continuations. In *Proc. 19th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'92)*, pp. 27–38.
13. Girard, J.-Y. (1987) Linear logic. *Theoret. Comp. Sci.* **50**, 1–102.
14. Hasegawa, M. (2002) Linearly used effects: monadic and CPS transformations into the linear lambda calculus. In *Proc. 6th International Symposium on Functional and Logic Programming (FLOPS2002)*, Springer Lecture Notes in Comput. Sci. **2441**, pp. 167–182.
15. Hasegawa, M. (2002) Classical linear logic of implications. In *Proc. 11th Annual Conference of the European Association for Computer Science Logic (CSL'02)*, Springer Lecture Notes in Comput. Sci. **2471**, pp. 458–472.
16. Hofmann, M. and Streicher, T. (1997) Continuation models are universal for $\lambda\mu$-calculus. In *Proc. 12th Annual IEEE Symposium on Logic in Computer Science (LICS'97)*, pp. 387–397.
17. Hyland, M. and Schalk, A. (2003) Glueing and orthogonality for models of linear logic. *Theoret. Comp. Sci.* **294**(1/2), 183–231.
18. Kakutani, Y. (2002) Duality between call-by-name recursion and call-by-value iteration. In *Proc. 11th Annual Conference of the European Association for Computer Science Logic (CSL'02)*, Springer Lecture Notes in Comput. Sci. **2471**, pp. 506–521.
19. Kameyama, Y. and Hasegawa, M. (2003) A sound and complete axiomatization of delimited continuations. In *Proc. 8th ACM SIGPLAN International Conference on Functional Programming (ICFP 2003)*, pp. 177–188.
20. Laird, J. (2003) A game semantics of linearly used continuations. In *Proc. 6th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS2003)*, Springer Lecture Notes in Comput. Sci. **2620**, pp. 313–327.
21. Lambek, J. and Scott, P.J. (1986) *Introduction to Higher Order Categorical Logic*. Cambridge Studies in Advanced Mathematics 7, Cambridge University Press.
22. Moggi, E. (1991) Notions of computation and monads. *Inform. and Comput.* **93**(1), 55–92.
23. Parigot, M. (1992) $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proc. International Conference on Logic Programming and Automated Reasoning (LPAR'92)*, Springer Lecture Notes in Comput. Sci. **624**, pp. 190–201.
24. Power, A.J. and Robinson, E.P. (1997) Premonoidal categories and notions of computation. *Math. Structures Comput. Sci.* **7**(5), 453–468.
25. Seely, R.A.G. (1989) Linear logic, $*$-autonomous categories and cofree coalgebras. In *Categories in Computer Science*, AMS Contemp. Math. **92**, pp. 371–389.
26. Selinger, P. (2001) Control categories and duality: on the categorical semantics of the lambda-mu calculus. *Math. Structures Comput. Sci.* **11**(2), 207–260.
27. Wadler, P. (2003) Call-by-value is dual to call-by-name. In *Proc. 8th ACM SIGPLAN International Conference on Functional Programming (ICFP 2003)*, pp. 189–202.

A Dual Intuitionistic Linear Logic

Types and Terms

$$\begin{array}{rcl} \sigma & ::= & b \mid I \mid \sigma \otimes \sigma \mid \sigma \multimap \sigma \mid !\sigma \\ M & ::= & x \mid * \mid \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ M \mid M \otimes M \mid \mathsf{let}\ x^\sigma \otimes x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ M \mid \\ & & \lambda x^\sigma.M \mid MM \mid !M \mid \mathsf{let}\ !x^\sigma\ \mathsf{be}\ M\ \mathsf{in}\ M \end{array}$$

Typing

$$\frac{}{\Gamma_1, x:\sigma, \Gamma_2\ ;\ \emptyset \vdash x:\sigma}\,(\text{Int-Ax}) \qquad \frac{}{\Gamma\ ;\ x:\sigma \vdash x:\sigma}\,(\text{Lin-Ax})$$

$$\frac{}{\Gamma\ ;\ \emptyset \vdash *:I}\,(I\,\mathrm{I}) \qquad \frac{\Gamma\ ;\ \Delta_1 \vdash M:I \quad \Gamma\ ;\ \Delta_2 \vdash N:\sigma}{\Gamma\ ;\ \Delta_1 \sharp \Delta_2 \vdash \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ N:\sigma}\,(I\,\mathrm{E})$$

$$\frac{\Gamma\ ;\ \Delta_1 \vdash M:\sigma_1 \quad \Gamma\ ;\ \Delta_2 \vdash N:\sigma_2}{\Gamma\ ;\ \Delta_1 \sharp \Delta_2 \vdash M \otimes N:\sigma_1 \otimes \sigma_2}\,(\otimes\,\mathrm{I}) \qquad \frac{\begin{array}{c}\Gamma\ ;\ \Delta_1 \vdash M:\sigma_1 \otimes \sigma_2 \\ \Gamma\ ;\ \Delta_2, x:\sigma_1, y:\sigma_2 \vdash N:\tau\end{array}}{\Gamma\ ;\ \Delta_1 \sharp \Delta_2 \vdash \mathsf{let}\ x^{\sigma_1} \otimes y^{\sigma_2}\ \mathsf{be}\ M\ \mathsf{in}\ N:\tau}\,(\otimes\,\mathrm{E})$$

$$\frac{\Gamma\ ;\ \Delta, x:\sigma_1 \vdash M:\sigma_2}{\Gamma\ ;\ \Delta \vdash \lambda x^{\sigma_1}.M:\sigma_1 \multimap \sigma_2}\,(\multimap\,\mathrm{I}) \qquad \frac{\Gamma\ ;\ \Delta_1 \vdash M:\sigma_1 \multimap \sigma_2 \quad \Gamma\ ;\ \Delta_2 \vdash N:\sigma_1}{\Gamma\ ;\ \Delta_1 \sharp \Delta_2 \vdash M\,N:\sigma_2}\,(\multimap\,\mathrm{E})$$

$$\frac{\Gamma\ ;\ \emptyset \vdash M:\sigma}{\Gamma\ ;\ \emptyset \vdash !M:!\sigma}\,(!\,\mathrm{I}) \qquad \frac{\Gamma\ ;\ \Delta_1 \vdash M:!\sigma \quad \Gamma, x:\sigma\ ;\ \Delta_2 \vdash N:\tau}{\Gamma\ ;\ \Delta_1 \sharp \Delta_2 \vdash \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ N:\tau}\,(!\,\mathrm{E})$$

where $\Delta_1 \sharp \Delta_2$ represents one of possible merges of Δ_1 and Δ_2 as finite lists.

Axioms

$$\begin{array}{rr} \mathsf{let}\ *\ \mathsf{be}\ *\ \mathsf{in}\ M = M & \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ * = M \\ \mathsf{let}\ x \otimes y\ \mathsf{be}\ M \otimes N\ \mathsf{in}\ L = L[M/x, N/y] & \mathsf{let}\ x \otimes y\ \mathsf{be}\ M\ \mathsf{in}\ x \otimes y = M \\ (\lambda x.M)\,N = M[N/x] & \lambda x.M\,x = M \\ \mathsf{let}\ !x\ \mathsf{be}\ !M\ \mathsf{in}\ N = N[M/x] & \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ !x = M \end{array}$$

$$\begin{array}{c} C[\mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ N] = \mathsf{let}\ *\ \mathsf{be}\ M\ \mathsf{in}\ C[N] \\ C[\mathsf{let}\ x \otimes y\ \mathsf{be}\ M\ \mathsf{in}\ N] = \mathsf{let}\ x \otimes y\ \mathsf{be}\ M\ \mathsf{in}\ C[N] \\ C[\mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ N] = \mathsf{let}\ !x\ \mathsf{be}\ M\ \mathsf{in}\ C[N] \end{array}$$

where $C[-]$ is a linear context (no ! binds $[-]$).

Semantics A typing judgement

$$x_1:\sigma_1, \ldots, x_m:\sigma_m\ ;\ y_1:\tau_1, \ldots, y_n:\tau_n \vdash M:\sigma$$

is inductively interpreted as a morphism $[\![x_1:\sigma_1, \ldots\ ;\ y_1:\tau_1, \ldots \vdash M:\sigma]\!]$ from $![\![\sigma_1]\!] \otimes \ldots \otimes ![\![\sigma_m]\!] \otimes [\![\tau_1]\!] \otimes \ldots \otimes [\![\tau_n]\!]$ to $[\![\sigma]\!]$ in a symmetric monoidal closed category with a linear exponential comonad !.

Proposition 4 (categorical completeness). *The equational theory of DILL is sound and complete for categorical models given by symmetric monoidal closed categories with linear exponential comonads: $\Gamma\ ;\ \Delta \vdash M = N:\sigma$ is provable if and only if $[\![\Gamma\ ;\ \Delta \vdash M:\sigma]\!] = [\![\Gamma\ ;\ \Delta \vdash N:\sigma]\!]$ holds for every such models.*

B !-Strong Monads

Let $\mathcal{D}$ be a symmetric monoidal category with a linear exponential comonad !. A monad (T, η, μ) on $\mathcal{D}$ is !-*strong* (or: *strong with respect to* ! [10]) if it is equipped with a natural transformation called !-*strength* (*strength with respect to* !)

$$\theta_{A,X} : !A \otimes TX \longrightarrow T(!A \otimes X)$$

subject to the following axioms.