Linear logic, monads, and the lambda calculus (DRAFT)

Nick Benton
University of Cambridge*
Nick.Benton@cl.cam.ac.uk

Philip Wadler
University of Glasgow†
wadler@dcs.glasgow.ac.uk

October 1995

Abstract

A category theorist models Girard's linear logic by comonads (with some additional structure), and Moggi's computational metalanguage by monads (with different additional structure). Both of these systems are are of interest, in part because of the existence of various translations of the simply-typed lambda calculus (or, equivalently, propositional intuitionistic logic) into each of them. Are the similarities between the two systems a mirage, or the reflection of a deep connection?

Recently, Benton described an adjoint model that subsumes both models of intuitionistic linear logic and a restricted class of models of the computational metalanguage. This shows that Girard's and Moggi's systems are indeed connected.

Here we strengthen that link by showing that the various translations of the lambda calculus proposed by Girard and Moggi are intimately related. In particular, the well-known 'Girard translation' corresponds to the trivial embedding of the lambda calculus into Moggi's metalanguage; an alternative translation due to Girard corresponds to Moggi's call-by-value translation; and Moggi's call-by-name translation corresponds to a new variant of the standard Girard translation.

Further, there are translations between the lambda calculi corresponding to these models. Corresponding to the adjoint model is a mixed linear and non-linear lambda calculus devised by Benton; corresponding to Moggi's model is his computational metalanguage; corresponding to Girard's work are a number of linear lambda calculi. The monadic and linear calculi map into the adjoint calculus, and this map preserves meaning.

*Computer Laboratory, Pembroke Street, Cambridge CB2 3QG. Research supported by the EU BRA 8130 LOMAPS.

†Department of Computing Science, University of Glasgow, Glasgow G12 8QQ, Scotland.

1 Introduction

Less then a decade ago, two upheavals struck in close succession and left the landscape of computational semantics forever changed: the linear logic of Girard [Gir87] and the computational metalanguage of Moggi [Mog89]. Any student of these formalisms will spot differences on the surface, but suspect there are deeper connections underneath. Have the different topographies been shaped by a single underlying pressure?

The intuitionistic fragment of Girard's formalism may be modeled in a symmetric monoidal closed category with a comonad ! and some addditional structure. An early model was sketched by Seely [See89], and a later model detailed by Benton, Bierman, de Paiva, and Hyland [BBdPH92], dubbed the 'Gang of Four'. The two models are similar, and their relation has been detailed by Bierman [Bie94]. Meanwhile, Moggi's formalism may be modelled in a cartesian closed category with a monad T and some different additional structure. The situations are tantalisingly close to dual: ! is almost, but not quite, entirely like T.

Following a suggestion of Plotkin, Benton has given an alternative presentation of models of intuitionistic linear logic, based on a cartesian closed category $\mathcal{C}$ and a symmetric monoidal closed category $\mathcal{L}$ connected by functors $F : \mathcal{C} \to \mathcal{L}$ and $G : \mathcal{L} \to \mathcal{C}$, with F left adjoint to G. Again there is additional structure, which is pleasingly easy to describe: the adjunction must be symmetric monoidal [Ben94]. As noted by Benton, taking $! = FG$ yields a model of linear logic in the style of the Gang of Four, while taking $T = GF$ yields a model of the computational lambda calculus. Further, every model of linear logic is yielded in this way; as is every model of computational lambda calculus which has certain equalisers and coequalisers and in which the monad is commutative.

There are calculi that correspond to the work of Moggi and Girard, which we here christen the *monad calculus*, λ_m, and the *linear calculus*, λ_l. There is an obvious candidate for the monad calculus, namely Moggi's computational lambda calculus. There are a plethora of candidates for the linear calculus, including proposals of Lafont [Laf88], Holmström [Hol88], Wadler [Wad90, Wad91], Abramsky [Abr93], Mackie [Mac94a], Lincoln and Mitchell [LM92], Troelstra [Tro92], Benton, Bierman, dePaiva, and Hyland [BBdPH92, Bie94], and della Rocca and Roversi [RR94]. This version of this paper uses the syntax of [BBdPH92].

Girard and Moggi proposed various embeddings of lambda calculus into their systems. Girard proposed two mappings of lambda calculus into linear logic, labeled $\circ$ and $*$ [Gir87]. It later turned out that the $\circ$ mapping corresponds to call-by-name lambda calculus, while the $*$ mapping corresponds to call-by-value [Mac94b, MOTW95]. Moggi also proposed two mappings, clearly labeled as call-by-value and call-by-name [Mog89]. The call-by-value translation was typed and the call-by-name untyped; but it is no surprise that typed translations for both exist, and they have been used by many authors, see [Pitts90, Crole92, Ben92, Wad92] for example. These translations also correspond closely to the call-by-value and call-by-name translations of lambda calculus into continuation passing style described by Plotkin [Plo75].

From our perspective, these may be viewed as translations of traditional simply-typed lambda calculus into the monadic and linear calculi. Indeed, there are three distinct source lambda calculi, with identical syntax but differing semantics. First there is the *direct* calculus, λ, corresponding to Church's traditional calculus. Second, there is the *lifted* calculus[1], $\lambda_\perp$, corresponding to Abramsky and Ong's lazy lambda calculus [Abr92, Ong88]. Third, there is the *call-by-value* calculus, λ_v, corresponding to Plotkin's calculus of that name [Plo75]. The lifted and call-by-value calculi both restrict the usual equational laws of the simply-typed lambda calculus so as to yield calculi which are correct for reasoning about observational equivalance in real functional programming languages.

Each of these three lambda calculi is given a semantics via two translations, into the monadic and linear calculi. The monad translation yields a semantics in $\mathcal{C}$, and the linear translation yields a semantics in $\mathcal{L}$, and these are related by the adjunction between the categories. From the direct call-by-name calculus, one has the $\circ$ translation of Girard, and a trivial embedding into Moggi's calculus. From the lifted call-by-name calculus, one has a modification of Girard's $\circ$ translation, and Moggi's call-by-name translation. From the call-by-value calculus, one has Girard's $*$ translation, and Moggi's call-by-value translation.

Everything is quite straightforward, so it is worthwhile to remind ourselves of what is being achieved: we are showing that translations which have been around for years, which seem to have similar purposes and yet have distinctly different appearances, are in fact equivalent. Thus for instance, Girard's $*$ translation published in 1987 corresponds exactly to Moggi's call-by-value translation published in 1989, and we can finally see that this is the case in 1995.

Corresponding to the adjoint model there is an elegant *adjoint calculus*, λ_a, proposed by Benton [Ben94]. As one would hope, there are embeddings of the monadic and linear lambda calculi into the adjoint calculus, and these maps preserve meaning. There does not appear to be any way to map the linear calculus into the monadic or vice versa without introducing some extra structure (such as fresh base types), so the adjoint calculus seems to be strictly more expressive than the other two.

This paper is divided into three parts. Part I reviews the adjoint model in Section 2, and the monadic and linear models and their corresponding calculi in Sections 3 and 4 respectively. Part II considers translations from the direct, call-by-name, and call-by-value lambda calculi, in Sections 6, 7, and 8, respectively. Part III reviews the adjoint calculus in Section 9, and considers translations from the monadic and linear calculi into this calculus in Section 10. Section 11 concludes.

[1] There is a slight problem of nomenclature here. The direct calculus is valid for reasoning about a call-by-name language with observation only at ground types, such as PCF [Plo77], so we could have called that 'call-by-name'. We could also have called $\lambda_\perp$ the call-by-name calculus, since it corresponds to a call-by-name language with convergence testing at higher types, or we could have followed Abramsky and called $\lambda_\perp$ the lazy calculus. Unfortunately the word 'lazy' is often also used to mean call-by-need, which refers to an optimisation in the implementation of languages with call-by-name semantics, so we prefer not to use it here.

Part I
Linear logic and monads

2 The adjoint model

This section reviews some basic definitions about monoidal categories, functors and natural transformations and then introduces the adjoint model from [Ben94].[2]

Definition 2.1 *A* monoidal category *is a category* $\mathcal{M}$ *equipped with a bifunctor* $\otimes : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$*, an object* I *of* $\mathcal{M}$*, and natural isomorphisms*

$$\alpha_{X,Y,Z} : (X \otimes Y) \otimes Z \to X \otimes (Y \otimes Z)$$

$$l_X : I \otimes X \to X$$

$$r_X : X \otimes I \to X$$

which satisfy the following pair of coherence diagrams:

$$\begin{tikzcd}
((W \otimes X) \otimes Y) \otimes Z \arrow[r, "\alpha \otimes 1"] \arrow[d, "\alpha"'] & (W \otimes (X \otimes Y)) \otimes Z \arrow[dd, "\alpha"] \\
(W \otimes X) \otimes (Y \otimes Z) \arrow[d, "\alpha"'] & \\
W \otimes (X \otimes (Y \otimes Z)) & W \otimes ((X \otimes Y) \otimes Z) \arrow[l, "1 \otimes \alpha"]
\end{tikzcd}$$

$$\begin{tikzcd}
(X \otimes I) \otimes Y \arrow[rr, "\alpha"] \arrow[dr, "r \otimes 1"'] & & X \otimes (I \otimes Y) \arrow[dl, "1 \otimes l"] \\
& X \otimes Y &
\end{tikzcd}$$

and for which $l_I = r_I$.

[2] Although we are careful to include all the necessary coherence conditions in the definitions, we will follow common (sloppy!) practice and merely assume the coherence results which are needed to ensure that our semantic interpretations are uniquely defined.

Definition 2.2 *A* symmetric monoidal category *(SMC) is a monoidal category* $(\mathcal{M}, \otimes, I, \alpha, l, r)$ *together with a natural transformation* $\sigma_{X,Y} : X \otimes Y \to Y \otimes X$ *satisfying the following three coherence conditions:*

$$\begin{tikzcd}
(X \otimes Y) \otimes Z \arrow[r, "\alpha"] \arrow[d, "\sigma \otimes 1"'] & X \otimes (Y \otimes Z) \arrow[r, "\sigma"] & (Y \otimes Z) \otimes X \arrow[d, "\alpha"] \\
(Y \otimes X) \otimes Z \arrow[r, "\alpha"'] & Y \otimes (X \otimes Z) \arrow[r, "1 \otimes \sigma"'] & Y \otimes (Z \otimes X)
\end{tikzcd}$$

$$\begin{tikzcd}
X \otimes Y \arrow[d, "\sigma"'] \arrow[dr, "1"] & \\
Y \otimes X \arrow[r, "\sigma"'] & X \otimes Y
\end{tikzcd}
\qquad
\begin{tikzcd}
I \otimes X \arrow[rr, "\sigma"] \arrow[dr, "l"'] & & X \otimes I \arrow[dl, "r"] \\
& X &
\end{tikzcd}$$

Note that every cartesian category (i.e. with finite products) is an SMC.

Definition 2.3 *A* symmetric monoidal closed category *(SMCC) is a symmetric monoidal category* $(\mathcal{M}, \otimes, I, \alpha, l, r, \sigma)$ *such for each* $B \in \mathcal{M}_0$ *the functor* $- \otimes B : \mathcal{M} \to \mathcal{M}$ *has a (specified) right adjoint. Thus there is for every* $A, C \in \mathcal{M}_0$ *an object* $(B \multimap C)$ *and a natural bijection*

$$\mathcal{M}(A \otimes B, C) \cong \mathcal{M}(A, B \multimap C)$$

Symmetric monoidal closed categories are also sometimes called *autonomous* categories.

Definition 2.4 *A* cartesian closed category *(CCC) is an SMCC for which the tensor product is cartesian.*

Whilst one might wish to consider functors between SMCs which preserve the structure on the nose or up to natural isomorphism, we shall take the class of functors between SMCs to be those preserving the structure up to a comparsion. We thus make the following definitions.

Definition 2.5 *Given monoidal categories* $(\mathcal{M}, \otimes, I, \alpha, l, r)$ *and* $(\mathcal{M}', \otimes', I', \alpha', l', r')$, *a* monoidal functor $F : \mathcal{M} \to \mathcal{M}'$ *is a functor from* $\mathcal{M}$ *to* $\mathcal{M}'$ *equipped with a map* $m_I : I' \to F(I)$ *in* $\mathcal{M}'$ *and a natural transformation* $m_{X,Y} : F(X) \otimes' F(Y) \to$

$F(X \otimes Y)$ *which satisfy the following coherence conditions:*

$$\begin{tikzcd}
(F(X) \otimes' F(Y)) \otimes' F(Z) \arrow[r, "\alpha'"] \arrow[d, "m \otimes' 1"'] & F(X) \otimes' (F(Y) \otimes' F(Z)) \arrow[d, "1 \otimes' m"] \\
F(X \otimes Y) \otimes' F(Z) \arrow[d, "m"'] & F(X) \otimes' F(Y \otimes Z) \arrow[d, "m"] \\
F((X \otimes Y) \otimes Z) \arrow[r, "F(\alpha)"'] & F(X \otimes (Y \otimes Z))
\end{tikzcd}$$

$$\begin{tikzcd}
I' \otimes' F(X) \arrow[r, "l'"] \arrow[d, "m \otimes' 1"'] & F(X) \\
F(I) \otimes' F(X) \arrow[r, "m"'] & F(I \otimes X) \arrow[u, "F(l)"']
\end{tikzcd}
\qquad
\begin{tikzcd}
F(X) \otimes' I' \arrow[r, "r'"] \arrow[d, "1 \otimes' m"'] & F(X) \\
F(X) \otimes' F(I) \arrow[r, "m"'] & F(X \otimes I) \arrow[u, "F(r)"']
\end{tikzcd}$$

Definition 2.6 *If $\mathcal{M}$ and $\mathcal{M}'$ above are* symmetric *monoidal, then F is a* symmetric monoidal functor *if it is monoidal and in addition satisfies the following coherence condition:*

$$\begin{tikzcd}
F(X) \otimes' F(Y) \arrow[r, "\sigma'"] \arrow[d, "m"'] & F(Y) \otimes' F(X) \arrow[d, "m"] \\
F(X \otimes Y) \arrow[r, "F(\sigma)"'] & F(Y \otimes X)
\end{tikzcd}$$

In the definition of a symmetric monoidal functor, one of the coherence diagrams for l and r is redundant, as it follows from the other and the diagram for σ. Note also that the identity functor is (symmetric) monoidal and that (symmetric) monoidal functors can be composed in an obvious way – if $(F, m) : \mathcal{M} \to \mathcal{M}'$ and $(G, n) : \mathcal{M}' \to \mathcal{M}''$ then their composite is given by the usual composition of functors together with the comparison natural transformation $p_{X,Y} : GFX \otimes'' GFY \to GF(X \otimes Y)$ where $p_{X,Y} = G(m_{X,Y}) \circ n_{FX,FY}$ (and similarly for the nullary version). It is then routine to check that (GF, p) is indeed a (symmetric) monoidal functor, and that (symmetric) monoidal categories and (symmetric) monoidal functors form a category.

Definition 2.7 *If (F, m) and (G, n) are monoidal functors from an MC $\mathcal{M}$ to an MC $\mathcal{M}'$, then a* monoidal natural transformation *from (F, m) to (G, n) is a natural* *transformation f_X from F to G which is compatible with the comparison maps in the sense that the following two diagrams commute:*

$$\begin{tikzcd} F(X) \otimes' F(Y) \arrow[r, "m"] \arrow[d, "f_X \otimes' f_Y"'] & F(X \otimes Y) \arrow[d, "f_{X \otimes Y}"] \\ G(X) \otimes' G(Y) \arrow[r, "n"'] & G(X \otimes Y) \end{tikzcd} \qquad \begin{tikzcd} F(I) \arrow[rr, "f_I"] & & G(I) \\ & I' \arrow[ul, "m"] \arrow[ur, "n"'] & \end{tikzcd}$$

Definition 2.8 *If $\mathcal{M}$ and $\mathcal{M}'$ are (symmetric) monoidal categories then a* (symmetric) monoidal adjunction *between them is an ordinary adjunction in which both of the functors are (symmetric) monoidal functors and both the unit and the counit of the adjunction are monoidal natural transformations (with respect to the natural monoidal structure on the two composite functors, as defined above).*

We are now in a position to make our central definition:

Definition 2.9 *An* adjoint model *consists of*

- *A cartesian closed category* $(\mathcal{C}, 1, \times, \to)$,
- *A symmetric monoidal closed category* $(\mathcal{L}, I, \otimes, \multimap)$,
- *A symmetric monoidal adjunction* $(F, G, \eta, \epsilon, m, n)$ *from* $\mathcal{C}$ *to* $\mathcal{L}$.

Let A, B, C and e, f, g range over objects and arrows in $\mathcal{L}$, and let X, Y, Z and s, t, u range over objects and arrows in $\mathcal{C}$. Expanding the above definition, we have arrows

$$\begin{aligned} m_I &: I \to F1, \\ m_{X,Y} &: FX \otimes FY \to F(X \times Y), \end{aligned}$$

$$\begin{aligned} n_1 &: 1 \to GI, \\ n_{A,B} &: GA \times GB \to G(A \otimes B). \end{aligned}$$

It turns out that m is an isomorphism, though n is not. Define

$$\begin{aligned} p_I &= F1 \xrightarrow{Fn_1} FGI \xrightarrow{\epsilon} I, \\ p_{X,Y} &= F(X \times Y) \xrightarrow{F(\eta\times\eta)} F(GFX \times GFY) \xrightarrow{n_{FX,FY}} FG(FX \otimes FY) \xrightarrow{\epsilon} FX \otimes FY. \end{aligned}$$

Then it is easy to show that m and p are inverses. Hence $F1 \simeq I$ and $F(X \times Y) \simeq FX \otimes FY$.

We shall also need the fact that in $\mathcal{C}$ there is an isomorphism $X \to GB \simeq G(FX \multimap B)$. This is derived from the following chain of isomorphisms between homsets, each of which is natural in Θ, X, and B,

$$\begin{array}{ll} & \mathrm{hom}_{\mathcal{C}}(\Theta,\ X \to GB) \\ \simeq & \qquad \{\text{ relation of } \times \text{ to } \to \} \\ & \mathrm{hom}_{\mathcal{C}}(\Theta \times X,\ GB) \\ \simeq & \qquad \{\ G \text{ is right adjoint to } F\ \} \\ & \mathrm{hom}_{\mathcal{L}}(F(\Theta \times X),\ B) \\ \simeq & \qquad \{\text{ relation of } \times \text{ to } \otimes \} \\ & \mathrm{hom}_{\mathcal{L}}(F\Theta \otimes FX,\ B) \\ \simeq & \qquad \{\text{ relation of } \otimes \text{ to } \multimap \} \\ & \mathrm{hom}_{\mathcal{L}}(F\Theta,\ FX \multimap B) \\ \simeq & \qquad \{\ F \text{ is left adjoint to } G\ \} \\ & \mathrm{hom}_{\mathcal{C}}(\Theta,\ G(FX \multimap B)). \end{array}$$

It follows that there is an isomorphism

$$X \to GB \simeq G(FX \multimap B).$$

The arrow from left to right is constructed by setting Θ to $X \to GB$, setting the arrow in the first hom set to the identity, and taking the corresponding arrow in the last hom set. The arrow from right to left is constructed, by setting Θ to $G(FX \multimap B)$, setting the arrow in the last hom set to the identity, and taking the corresponding arrow in the first hom set. The naturality conditions are adequate to guarantee that these arrows are inverses, and natural in X and B.

Example 2.10 Since in this paper we are interested in call-by-value, call-by-name and lifted lambda calculi, the most important examples of adjoint models for us are those arising from the operation of lifting on various categories of complete partial orders.

Let $\mathcal{L}$ be the category of pointed ωcpos (ω-cocomplete partial orders with a least element) and strict (bottom preserving) continuous maps. This is a symmetric monoidal closed category with tensor product given by the so-called smash product, the identity for the tensor by the two-point space and internal hom by the strict continuous function space. In fact, $\mathcal{L}$ also has binary products and coproducts, given by cartesian product and coalesced sum respectively. The one-point space is a biterminator.

Given this choice of $\mathcal{L}$, there are a couple of obvious choices for the CCC $\mathcal{C}$ which give an adjoint model. One is to take $\mathcal{C}$ to be the category of pointed ω-cpos and continuous (not necessarily strict) maps, G to be the inclusion functor and F to be the lifting functor $F : X \to X_\perp$. The monoidal structure m on F is given by the evident isomorphism $X_\perp \otimes Y_\perp \cong (X \times Y)_\perp$. In this case, $\mathcal{C}$ is (equivalent to) the Kleisli category of the lifting comonad on $\mathcal{L}$. Note that the cartesian closure of the Kleisli category follows from the fact that $\mathcal{L}$ has products. There are strong coproducts in $\mathcal{L}$ but only weak ones in $\mathcal{C}$.

An alternative choice of $\mathcal{C}$ is the category of (not necessarily pointed) ω-cpos (these are sometimes called *predomains*) and continuous maps, again with inclusion and lifting functors.

3 The monad model

This section introduces models of Moggi's computational metalanguage. The definition of strong monad makes sense in any monoidal category, but we will only need the special case where the tensor product is cartesian.

Definition 3.1 *If $\mathcal{C}$ is a cartesian category, and (T, η, μ) is a monad on $\mathcal{C}$, then T is a* strong monad *if there is a natural transformation τ (called the* tensorial strength*) with components*

$$\tau_{X,Y} : X \times TY \to T(X \times Y)$$

such that the following four diagrams commute:

$$\begin{tikzcd}
1 \times TX \arrow[r, "\tau"] \arrow[dr, "1"'] & T(1 \times X) \arrow[d, "T(1)"] \\
& TX
\end{tikzcd}
\qquad
\begin{tikzcd}
X \times Y \arrow[r, "1 \times \eta"] \arrow[dr, "\eta"'] & X \times TY \arrow[d, "\tau"] \\
& T(X \times Y)
\end{tikzcd}$$

$$\begin{tikzcd}
(X \times Y) \times TZ \arrow[rr, "\tau"] \arrow[d, "\alpha"'] & & T((X \times Y) \times Z) \arrow[d, "T(\alpha)"] \\
X \times (Y \times TZ) \arrow[r, "1 \times \tau"'] & X \times T(Y \times Z) \arrow[r, "\tau"'] & T(X \times (Y \times Z))
\end{tikzcd}$$

$$\begin{tikzcd}
X \times T^2Y \arrow[r, "\tau"] \arrow[d, "1 \times \mu"'] & T(X \times TY) \arrow[r, "T(\tau)"] & T^2(X \times Y) \arrow[d, "\mu"] \\
X \times TY \arrow[rr, "\tau"'] & & T(X \times Y)
\end{tikzcd}$$

Definition 3.2 *A* monad model *consists of*

- *A cartesian closed category $(\mathcal{C}, 1, \times, \to)$,*
- *A monad (T, η, μ) on $\mathcal{C}$,*
- *A tensorial strength $\tau_{X,Y} : X \times TY \to T(X \times Y)$ for T.*

Definition 3.3 *A monad (T, η, μ) over a cartesian category $\mathcal{C}$ with tensorial strength τ has a* twisted tensorial strength *τ' given by*

$$\tau'_{X,Y} \;=\; TX \times Y \simeq Y \times TX \xrightarrow{\tau_{Y,X}} T(Y \times X) \simeq T(X \times Y)$$

and two strengths Φ *and* Φ' *given by*

$$\Phi_{X,Y} \;=\; TX \times TY \xrightarrow{\tau'_{X,TY}} T(X \times TY) \xrightarrow{T\tau_{X,Y}} T^2(X \times Y) \xrightarrow{\mu} T(X \times Y),$$

$$\Phi'_{X,Y} \;=\; TX \times TY \xrightarrow{\tau_{TX,Y}} T(TX \times Y) \xrightarrow{T\tau'_{X,Y}} T^2(X \times Y) \xrightarrow{\mu} T(X \times Y).$$

The monad is commutative *if* $\Phi = \Phi'$.

Every adjoint model gives rise to a commutative monad model:

Proposition 3.0.1 *If* $(\mathcal{C}, \mathcal{L}, F, G, \eta, \epsilon)$ *is an adjoint model, then* $(\mathcal{C}, GF, \eta, G\epsilon F)$ *is a monad model with a commutative monad.*

Proof. See Benton [Ben94]. □

In the reverse direction, we have the following:

Proposition 3.0.2 *If* $(\mathcal{C}, T, \eta, \mu)$ *is a monad model with a commutative monad and in addition* $\mathcal{C}$ *has equalisers and coequalisers, then the Eilenberg-Moore category* $\mathcal{C}^T$ *is symmetric monoidal closed and* $(\mathcal{C}, \mathcal{C}^T, F, G, \eta, \epsilon)$ *is an adjoint model, where* G *is the forgetful functor sending the* T*-algebra* $(X, t : TX \to X)$ *to the underlying object* X*,* F *is the free functor sending* X *to the algebra* $(TX, \mu : T^2X \to TX)$ *and the component of* η *at the algebra* $(X, t : TX \to X)$ *is the algebra map* t*.*

Proof. See Keigher [Kei78]. □

The above result is not quite as strong as we should like – things would be nicer if we could construct an adjoint model from a commutative monad model without requiring equalisers and coequalisers, but we do not yet know if this is possible.

The monad calculus λ_m is presented in Figure 1. Corresponding to each judgement

$$x_1 : X_1, \ldots, x_m : X_m \vdash u : Y$$

in λ_m is an arrow

$$X_1 \times \cdots \times X_m \xrightarrow{u} Y$$

in $\mathcal{C}$.

4 The linear model

This section introduces 'Gang of Four' models of linear logic and the linear calculus λ_l. We are using the syntax of [BBdPH92] since it makes the type structure of the various translations clearer. Unfortunately, it also makes the terms associated with the translations rather large and unwieldy. The alternative would have been to use the syntax due to Wadler [Wad93a, Wad93b, MOTW95]. That calculus was suggested to Wadler by Girard, and is based on Girard's Logic of Unity, but without polarities [Gir93]. Plotkin independently proposed a similar calculus, which has been further developed by Barber [Bar95]. Using Wadler's syntax makes two of the translations considerably more concise, but hides some structures by making the use of ! on the left implicit rather than explicit.

Syntax

atomic types	K		
variables	x, y, z		
types	X, Y, Z	$::=$	$K \mid 1 \mid X \times Y \mid X \to Y \mid TX$
contexts	Θ	$::=$	$x_1 : X_1, \ldots, x_n : X_n$
terms	s, t, u	$::=$	$x \mid () \mid (t, u) \mid \mathrm{fst}\ s \mid \mathrm{snd}\ s \mid \lambda x{:}X.\,u \mid s\,t \mid [t] \mid \mathrm{let}\ x \leftarrow s\ \mathrm{in}\ u$

Types

$$\mathrm{Id}\ \frac{}{x_1 : X_1, \ldots, x_n : X_n \vdash x_i : X_i} \qquad \text{1-I}\ \frac{}{\Theta \vdash () : 1}$$

$$\times\text{-I}\ \frac{\Theta \vdash t : X \qquad \Theta \vdash u : Y}{\Theta \vdash (t, u) : X \times Y} \qquad \times\text{-E}\ \frac{\Theta \vdash s : X \times Y}{\Theta \vdash \mathrm{fst}\ s : X} \qquad \frac{\Theta \vdash s : X \times Y}{\Theta \vdash \mathrm{snd}\ s : Y}$$

$$\to\text{-I}\ \frac{\Theta, x : X \vdash u : Y}{\Theta \vdash \lambda x{:}X.\,u : X \to Y} \qquad \to\text{-E}\ \frac{\Theta \vdash s : X \to Y \qquad \Theta \vdash t : X}{\Theta \vdash s\,t : Y}$$

$$T\text{-I}\ \frac{\Theta \vdash t : X}{\Theta \vdash [t] : TX} \qquad T\text{-E}\ \frac{\Theta \vdash s : TX \qquad \Theta, x : X \vdash u : TY}{\Theta \vdash \mathrm{let}\ x \leftarrow s\ \mathrm{in}\ u : TY}$$

Categorical interpretation

$$\mathrm{Id}\ \frac{}{X_1 \times \cdots \times X_n \xrightarrow{\mathrm{sel}_i} X_i} \qquad \text{1-I}\ \frac{}{\Theta \xrightarrow{\langle\rangle} 1}$$

$$\times\text{-I}\ \frac{\Theta \xrightarrow{t} X \qquad \Theta \xrightarrow{u} Y}{\Theta \xrightarrow{\langle t, u\rangle} X \times Y} \qquad \times\text{-E}\ \frac{\Theta \xrightarrow{s} X \times Y}{\Theta \xrightarrow{s} X \times Y \xrightarrow{\mathrm{fst}} X} \qquad \frac{\Theta \xrightarrow{s} X \times Y}{\Theta \xrightarrow{s} X \times Y \xrightarrow{\mathrm{snd}} Y}$$

$$\to\text{-I}\ \frac{\Theta \times X \xrightarrow{u} Y}{\Theta \xrightarrow{\mathrm{cur}(u)} X \to Y} \qquad \to\text{-E}\ \frac{\Theta \xrightarrow{s} X \to Y \qquad \Theta \xrightarrow{t} X}{\Theta \xrightarrow{\langle s, t\rangle} (X \to Y) \times X \xrightarrow{\mathrm{app}} Y}$$

$$T\text{-I}\ \frac{\Theta \xrightarrow{t} X}{\Theta \xrightarrow{t} X \xrightarrow{\eta} TX} \qquad T\text{-E}\ \frac{\Theta \xrightarrow{s} TX \qquad \Theta \times X \xrightarrow{u} TY}{\Theta \xrightarrow{\langle \mathrm{id}, s\rangle} \Theta \times TX \xrightarrow{\tau} T(\Theta \times X) \xrightarrow{Tu} T^2 Y \xrightarrow{\mu} TY}$$

Figure 1: Monadic lambda calculus (λ_m)

Definition 4.1 *A* linear model *consists of*

- *A symmetric monoidal closed category* $(\mathcal{L}, I, \otimes, \multimap)$.
- *A symmetric monoidal comonad* $(!, \epsilon, \delta, q)$ *on* $\mathcal{L}$.
- *Monoidal natural transformations with components*

$$\begin{array}{rcl} \mathrm{disc}_A & : & !A \to I \\ \mathrm{dupl}_A & : & !A \to !A \otimes !A \end{array}$$

such that

1. *each* $(!A, \mathrm{disc}_A, \mathrm{dupl}_A)$ *is a commutative comonoid,*
2. disc_A *and* dupl_A *are coalgebra maps, and*
3. *all coalgebra maps between free coalgebras preserve the comonoid structure.*

Every adjoint model gives rise to a linear model and vice versa:

Proposition 4.0.3 *If* $(\mathcal{C}, \mathcal{L}, F, G, \eta, \epsilon)$ *is an adjoint model, then* $(\mathcal{L}, FG, \epsilon, F\eta G)$ *is a linear model.*

Proof. The monoidal struture $q_{A,B} : !A \otimes !B \to !(A \otimes B)$ is given by $q_{A,B} = F(n_{A,B}) \circ m_{GA,GB}$ and the components of the natural transformations disc_A and dupl_A are given by $p \circ F(\langle\,\rangle_{GA})$ and $p_{GA,GA} \circ F(\Delta_{GA})$ respectively, where $\langle\,\rangle_{GA}$ is the unique map from GA to the terminal object 1 in $\mathcal{C}$ and Δ_{GA} is the diagonal map from GA to $GA \times GA$ in $\mathcal{C}$. See [Ben94] for the details. □

Proposition 4.0.4 *Any linear model gives rise to an adjoint model, though it is not, in general, unique.*

Proof. Given a linear model $(\mathcal{L}, !, \epsilon, \delta)$, there are a couple of ways of constructing a CCC $\mathcal{C}$ as a full subcategory of the Eilenberg-Moore category $\mathcal{L}^!$ of coalgebras such that $(\mathcal{L}, \mathcal{C}, \ldots)$ is an adjoint model. Again, see [Ben94] for the details. □

A linear model may also be augmented with a product (&). This is essential for Seely's model, but not for 'Gang of Four' models. It is also essential for the direct translation of Section 6, but not for the call-by-name and call-by-value translations of Sections 7 and 8. The correspondence between adjoint and linear models is maintained in the presence of such a product.

The linear calculus λ_l is presented in Figure 2. Corresponding to each judgement

$$a_1 : A_1, \ldots, a_m : A_m \vdash e\colon B$$

is an arrow

$$A_1 \otimes \cdots \otimes A_m \xrightarrow{e} B$$

in $\mathcal{L}$.

Syntax

atomic types	L		
variables	a, b, c		
types	A, B, C	$::=$	$L \mid I \mid A \otimes B \mid 1 \mid A \mathbin{\&} B \mid A \multimap B \mid !A$
contexts	Γ, Δ	$::=$	$a_1 : A_1, \ldots, a_n : A_n$
terms	e, f, g, h	$::=$	$a \mid * \mid \text{let } * = e \text{ in } f \mid f \otimes g \mid \text{let } a \otimes b = e \text{ in } h \mid (e, f)$
			$\mid \text{fst } e \mid \text{snd } e \mid () \mid \lambda a{:}\, A.\, g \mid e\ f \mid \mathsf{derelict}\ e \mid \mathsf{discard}\ e\ \mathsf{in}\ f$
			$\mid \mathsf{copy}\ e\ \mathsf{as}\ a, b\ \mathsf{in}\ f \mid \mathsf{promote}\ e_1, \ldots, e_n\ \mathsf{for}\ a_1, \ldots, a_n\ \mathsf{in}\ f$

Types

$$\text{Id}\ \frac{}{a : A \vdash a : A} \qquad \text{I-I}\ \frac{}{\vdash *{:}\, I} \qquad \text{I-E}\ \frac{\Gamma \vdash e{:}\, I \quad \Delta \vdash f{:}\, A}{\Gamma, \Delta \vdash \text{let } * = e \text{ in } f{:}\, A}$$

$$\otimes\text{-I}\ \frac{\Gamma \vdash e{:}\, A \quad \Delta \vdash f{:}\, B}{\Gamma, \Delta \vdash e \otimes f{:}\, A \otimes B} \qquad \otimes\text{-E}\ \frac{\Gamma \vdash e{:}\, A \otimes B \quad \Delta, a{:}\, A, b{:}\, B \vdash f{:}\, C}{\Gamma, \Delta \vdash \text{let } a \otimes b = e \text{ in } f{:}\, C}$$

$$\&\text{-I}\ \frac{\Gamma \vdash e{:}\, A \quad \Gamma \vdash f{:}\, B}{\Gamma \vdash (e, f){:}\, A \mathbin{\&} B} \qquad \&\text{-E}\ \frac{\Gamma \vdash e{:}\, A \mathbin{\&} B}{\Gamma \vdash \text{fst } e{:}\, A} \qquad \frac{\Gamma \vdash e{:}\, A \mathbin{\&} B}{\Gamma \vdash \text{snd } e{:}\, B}$$

$$\text{1-I}\ \frac{}{\Gamma \vdash (){:}\, 1} \qquad \multimap\text{-I}\ \frac{\Gamma, a{:}\, A \vdash e{:}\, B}{\Gamma \vdash \lambda a{:}\, A.\, e{:}\, A \multimap B}$$

$$\multimap\text{-E}\ \frac{\Gamma \vdash e{:}\, A \multimap B \qquad \Delta \vdash f{:}\, A}{\Gamma,\ \Delta \vdash e\ f{:}\, B} \qquad \text{Der}\ \frac{\Gamma \vdash e{:}\, !A}{\Gamma \vdash \mathsf{derelict}\ e{:}\, A}$$

$$\text{Weak}\ \frac{\Gamma \vdash e{:}\, !A \qquad \Delta \vdash f{:}\, B}{\Gamma, \Delta \vdash \mathsf{discard}\ e\ \mathsf{in}\ f{:}\, B} \qquad \text{Contr}\ \frac{\Gamma \vdash e{:}\, !A \qquad \Delta, a{:}\, !A, b{:}\, !A \vdash f{:}\, B}{\Gamma, \Delta \vdash \mathsf{copy}\ e\ \mathsf{as}\ a, b\ \mathsf{in}\ f{:}\, B}$$

$$\text{Promote}\ \frac{\Delta_1 \vdash e_1{:}\, !A_1 \ \cdots\ \Delta_n \vdash e_n{:}\, !A_n \qquad a_1{:}\, !A_1, \ldots, a_n{:}\, !A_n \vdash f{:}\, B}{\Delta_1, \ldots, \Delta_n \vdash \mathsf{promote}\ e_1, \ldots, e_n\ \mathsf{for}\ a_1, \ldots, a_n\ \mathsf{in}\ f{:}\, !B}$$

Figure 2: Linear lambda calculus (λ_l)

Categorical interpretation

$$\text{Id}\ \frac{}{A \xrightarrow{1} A} \qquad \text{I-I}\ \frac{}{I \xrightarrow{1} I} \qquad \text{I-E}\ \frac{\Gamma \xrightarrow{e} I \qquad \Delta \xrightarrow{f} A}{\Gamma \otimes \Delta \xrightarrow{e \otimes f} I \otimes A \simeq A}$$

$$\otimes\text{-I}\ \frac{\Gamma \xrightarrow{e} A \qquad \Delta \xrightarrow{f} B}{\Gamma \otimes \Delta \xrightarrow{e \otimes f} A \otimes B} \qquad \otimes\text{-I}\ \frac{\Gamma \xrightarrow{e} A \otimes B \qquad \Delta \otimes A \otimes B \xrightarrow{f} C}{\Delta \otimes \Gamma \xrightarrow{1 \otimes f} \Delta \otimes A \otimes B \xrightarrow{f} C}$$

$$\&\text{-I}\ \frac{\Gamma \xrightarrow{e} A \qquad \Gamma \xrightarrow{f} B}{\Gamma \xrightarrow{\langle e,f \rangle} A \,\&\, B} \qquad \&\text{-E}\ \frac{\Gamma \xrightarrow{e} A \,\&\, B}{\Gamma \xrightarrow{e} A \,\&\, B \xrightarrow{\pi_1} A} \qquad \frac{\Gamma \xrightarrow{e} A \,\&\, B}{\Gamma \xrightarrow{e} A \,\&\, B \xrightarrow{\pi_2} B}$$

$$1\text{-I}\ \frac{}{\Gamma \xrightarrow{\langle\rangle} 1} \qquad \multimap\text{-I}\ \frac{\Gamma \otimes A \xrightarrow{e} B}{\Gamma \xrightarrow{\mathrm{cur}(e)} (A \multimap B)}$$

$$\multimap\text{-E}\ \frac{\Gamma \xrightarrow{e} A \multimap B \qquad \Delta \xrightarrow{f} A}{\Gamma \otimes \Delta \xrightarrow{e \otimes 1} (A \multimap B) \otimes A \xrightarrow{\mathrm{app}} B} \qquad \text{Der}\ \frac{\Gamma \xrightarrow{e} !A}{\Gamma \xrightarrow{e} !A \xrightarrow{\epsilon} A}$$

$$\text{Weak}\ \frac{\Gamma \xrightarrow{e} !A \qquad \Delta \xrightarrow{f} B}{\Gamma \otimes \Delta \xrightarrow{e \otimes f} !A \otimes B \xrightarrow{\mathrm{disc} \otimes 1} I \otimes B \simeq B}$$

$$\text{Cont}\ \frac{\Gamma \xrightarrow{e} !A \qquad \Delta \otimes !A \otimes !A \xrightarrow{f} B}{\Delta \otimes \Gamma \xrightarrow{1 \otimes e} \Delta \otimes !A \xrightarrow{1 \otimes \mathrm{dupl}} \Delta \otimes !A \otimes !A \xrightarrow{f} B}$$

$$\text{Prom}\ \frac{\Delta_1 \xrightarrow{e_1} !A_1 \ \cdots\ \Delta_n \xrightarrow{e_n} !A_n \qquad !A_1 \otimes \cdots \otimes !A_n \xrightarrow{f} B}{\Delta_1 \otimes \cdots \otimes \Delta_n \xrightarrow{e_1 \otimes \cdots \otimes e_n} !A_1 \otimes \cdots \otimes !A_n \xrightarrow{\delta \otimes \cdots \otimes \delta} !!A_1 \otimes \cdots \otimes !!A_n \xrightarrow{q} !(!A_1 \otimes \cdots \otimes !A_n) \xrightarrow{!f} !B}$$

Figure 3: Linear lambda calculus (λ_l), categorical interpretation

atomic types	K		
variables	x, y, z		
types	X, Y, Z	$::=$	$K \mid 1 \mid X \times Y \mid X \to Y$
contexts	Θ	$::=$	$x_1 : X_1, \ldots, x_n : X_n$
terms	s, t, u	$::=$	$x \mid () \mid (t, u) \mid \mathrm{fst}\, s \mid \mathrm{snd}\, s \mid \lambda x{:}\, X.\, u \mid s\, t$

Figure 4: Syntax of lambda calculus ($\lambda, \lambda_\perp, \lambda_v$)

Part II
The lambda calculus

We now turn to our three source lambda calculi and look at the translations of them into the linear and monadic calculi.

The lifted and call-by-value calculi were both developed in response to the mismatch between the standard equational theory of the direct lambda calculus and the theory of observational equivalence of functional programming languages: the usual β and η laws are both unsound for reasoning about observational equivalence in a call-by-value programming language, whilst the usual η law is unsound for a call-by-name language which allows the observation of termination at higher types.

One of the motivations behind Moggi's development of his computational metalanguage was to give a cleaner treatment of these issues. The monadic calculus lifts the distinction between computations and values which is already present in, for example, the modified $\beta\eta$ laws of Plotkin's λ_v to the level of types. This yields a calculus which is both very general (the one metalanguage can be used to model programming languages with different choices of evaluation order and observables and can also be used to describe a wide range of notions of computation other than potential non-termination) and mathematically much more natural (for example, like the ordinary simply typed lambda calculus and the linear lambda calculus, it arises via the Curry-Howard correspondence from an intuitionistic logic [BBdP95]).

The syntax of our three lambda calculi is given in Figure 4. It is a subset of the syntax of the monad calculus given in Figure 1, and the type rules are identical to those in the monad calculus.

The equational theory of the direct calculus λ is given by rules for deriving equations in context of the form $\Theta \vdash s = t : X$. These comprise some standard rules expressing the symmetry, reflexivity, transitivity and substitutivity (congruence) of the equality relation, together with the β and η laws shown in Figure 5.

The lifted calculus $\lambda_\perp$ has an equational theory which just adds the β rules from Figure 5 to the usual rules for equality.[3] The call-by-value calculus λ_v has an equational theory which restricts the rules of the direct calculus by insisting that

[3] $\lambda_\perp$ does satisfy some restricted η rules, such as $\lambda x.\, s = \lambda y.\, \lambda x.\, s\, y$, but these are derivable from the β laws.

Equations in context

$$\beta_{\to}\ \frac{\Theta, x\colon X \vdash s : Y \qquad \Theta \vdash t : X}{\Theta \vdash (\lambda x\colon X.\, s)\; t = s[t/x] : Y} \qquad \eta_{\to}\ \frac{\Theta \vdash s\colon X \to Y}{\Theta \vdash s = (\lambda x\colon X.\, s\; x) : X \to Y}(x \notin \Theta)$$

$$\beta_{\times}\ \frac{\Theta \vdash s\colon X \qquad \Theta \vdash t\colon Y}{\Theta \vdash \mathrm{fst}\ (s,t) = s : X} \qquad \frac{\Theta \vdash s\colon X \qquad \Theta \vdash t\colon Y}{\Theta \vdash \mathrm{snd}\ (s,t) = t : Y}$$

$$\eta_{1}\ \frac{\Theta \vdash e : 1}{\Theta \vdash e = () : 1} \qquad \eta_{\times}\ \frac{\Theta \vdash s : X \times Y}{\Theta \vdash s = (\mathrm{fst}\ s, \mathrm{snd}\ s) : X \times Y}$$

Figure 5: $\beta\eta$ rules of direct calculus (λ) and β rules of $\lambda_{\perp}$

Values

$$v, v' \ ::= \ x \mid () \mid (v, v') \mid \lambda x\colon X.\, s$$

Equations in context

$$\beta_{v\to}\ \frac{\Theta, x\colon X \vdash s : Y \qquad \Theta \vdash v : X}{\Theta \vdash (\lambda x\colon X.\, s)\; v = s[v/x] : Y} \qquad \eta_{v\to}\ \frac{\Theta \vdash v\colon X \to Y}{\Theta \vdash v = (\lambda x\colon X.\, v\; x) : X \to Y}(x \notin \Theta)$$

$$\beta_{v\times}\ \frac{\Theta \vdash s\colon X \qquad \Theta \vdash v\colon Y}{\Theta \vdash \mathrm{fst}\ (s,v) = s : X} \qquad \frac{\Theta \vdash v\colon X \qquad \Theta \vdash t\colon Y}{\Theta \vdash \mathrm{snd}\ (v,t) = t : Y}$$

$$\eta_{v1}\ \frac{\Theta \vdash v : 1}{\Theta \vdash v = () : 1} \qquad \eta_{v\times}\ \frac{\Theta \vdash v : X \times Y}{\Theta \vdash v = (\mathrm{fst}\ v, \mathrm{snd}\ v) : X \times Y}$$

Figure 6: Equational theory of the call-by-value lambda calculus λ_v

certain terms be *values*. The definition of values and the restricted rules are shown in Figure 6

Each of the translations which we shall present in the following sections will preserve the equational theory of the relevant version of the lambda calculus, though it may not reflect it. These results are rather messy to state formally since we have not given a syntactic equational theory for the linear or monadic calculi (these can be found in [BBdPH92] and [Mog89] respectively) and we have not given a categorical semantics for the lifted or call-by-value calculi, but, taking the call-by-value to monadic translation as an example (see Section 8), we can mix syntax and semantics and assert that if $\Theta \vdash s = t : X$ is provable in λ_v then the categorical semantics of the translations of s and t are equal, i.e.

$$[\![\Theta^{\star} \vdash s^{\star} : TX^{\star}]\!] = [\![\Theta^{\star} \vdash t^{\star} : TX^{\star}]\!]$$

We will not give all the six results of the above form explicitly, since we are more interested in the translations than in the various source lambda calculi themselves; but it is as well to note that there is a good reason why the various translations have the names they do. In fact, some authors have proved stronger results which directly relate *reductions* (rather than just equations) in the source calculi to those

Translation $|\text{-}|_m : \lambda_m \to \lambda, \lambda_\perp, \lambda_v$

$$
\begin{array}{rcl}
|K|_m & = & K \\
|1|_m & = & 1 \\
|X \times Y|_m & = & |X|_m \times |Y|_m \\
|X \to Y|_m & = & |X|_m \to |Y|_m \\
|TX|_m & = & |X|_m
\end{array}
$$

$$
\begin{array}{rcl}
|x|_m & = & x \\
|()|_m & = & () \\
|(s,t)|_m & = & (|s|_m, |t|_m) \\
|\text{fst } e|_m & = & \text{fst } |e|_m \\
|\text{snd } e|_m & = & \text{snd } |e|_m \\
|\lambda x\colon X.\, s|_m & = & \lambda x\colon |X|_m.\, |s|_m \\
|s\ t|_m & = & |s|_m\ |t|_m \\
|[s]|_m & = & |s|_m \\
|\text{let } x \leftarrow s \text{ in } t|_m & = & |t|_m[|s|_m/x]
\end{array}
$$

Figure 7: The monadic erasure translation

in their linear and monadic translations. See [Pitts90, MOTW95], for example.

5 Erasures

Before studying in detail the translations of each of the source calculi into the linear and monadic calculi, we need to define two translations of in the other direction which are the same for each of our source calculi. From now on, we assume that each base type K of the lambda calculus λ corresponds to a base type K of the monad calculus and a base type L_K of the linear calculus.

The *monadic erasure* translation $|\text{-}|_m$ maps types and terms of the monadic calculus λ_m into types and terms of the direct calculus λ (which are, of course, also the types and terms of $\lambda_\perp$ and λ_v). The definition of the monadic erasure is shown in Figure 7

The *linear erasure* translation $|\text{-}|_l$ maps types and terms of the linear calculus λ_l into types and terms of *lambda*, and is shown in Figure 8.

6 Direct translations

Since lambda calculus is a subset of the monad calculus, one possible translation is the trivial embedding which leaves types and terms unchanged. For consistency with everything which follows, we give this trivial translation a name, $\bullet : \lambda \to \lambda_m$, and write it out in Figure 9.

Translation $|\text{-}|_l : \lambda_l \to \lambda, \lambda_\perp, \lambda_v$

$$\begin{array}{rcl}
|L_K|_l & = & K \\
|I|_l & = & 1 \\
|A \otimes B|_l & = & |A|_l \times |B|_l \\
|1|_l & = & 1 \\
|A \mathbin{\&} B|_l & = & |A|_l \times |B|_l \\
|A \multimap B|_l & = & |A|_l \to |B|_l \\
|!A|_l & = & |A|_l
\end{array}$$

$$\begin{array}{rcl}
|a|_l & = & a \\
|*|_l & = & () \\
|\text{let } * = e \text{ in } f|_l & = & |f|_l \\
|e \otimes f|_l & = & (|e|_l, |f|_l) \\
|\text{let } a \otimes b = e \text{ in } f|_l & = & |f|_l[\text{fst } |e|_l/a, \text{snd } |e|_l/b] \\
|()|_l & = & () \\
|(e, f)|_l & = & (|e|_l, |f|_l) \\
|\text{fst } e|_l & = & \text{fst } |e|_l \\
|\text{snd } e|_l & = & \text{snd } |e|_l \\
|\lambda a\colon A.\, f|_l & = & \lambda a\colon |A|_l.\, |f|_l \\
|d\, e|_l & = & |d|_l\, |e|_l \\
|\textsf{derelict } e|_l & = & |e|_l \\
|\textsf{discard } e \textsf{ in } f|_l & = & |f|_l \\
|\textsf{copy } e \textsf{ as } a, b \textsf{ in } f|_l & = & |f|_l[|e|_l/a, |e|_l/b] \\
|\textsf{promote } \vec{e_i} \textsf{ for } \vec{a_i} \textsf{ in } f|_l & = & |f|_l[\overrightarrow{|e_i|_l}/\vec{x_i}]
\end{array}$$

Figure 8: The linear erasure translation

Translation $\bullet : \lambda \to \lambda_m$

$$
\begin{array}{rcl}
K^\bullet & = & K \\
1^\bullet & = & 1 \\
(X \times Y)^\bullet & = & X^\bullet \times Y^\bullet \\
(X \to Y)^\bullet & = & X^\bullet \to Y^\bullet \\[1ex]
(\Theta \vdash t : X)^\bullet & = & \Theta^\bullet \vdash t^\bullet : X^\bullet \\[1ex]
(\Theta, x{:}\, X \vdash x{:}\, X)^\bullet & = & \Theta^\bullet, x{:}\, X^\bullet \vdash x{:}\, X^\bullet \\
(\Theta \vdash ()\colon 1)^\bullet & = & \Theta^\bullet \vdash ()\colon 1 \\
(\Theta \vdash (s,t)\colon X \times Y)^\bullet & = & \Theta^\bullet \vdash (s^\bullet, t^\bullet)\colon X^\bullet \times Y^\bullet \\
(\Theta \vdash \mathrm{fst}\, s\colon X)^\bullet & = & \Theta^\bullet \vdash \mathrm{fst}\, s^\bullet\colon X^\bullet \\
(\Theta \vdash \mathrm{snd}\, s\colon Y)^\bullet & = & \Theta^\bullet \vdash \mathrm{snd}\, s^\bullet\colon Y^\bullet \\
(\Theta \vdash \lambda x{:}\, X.\, s\colon X \to Y)^\bullet & = & \Theta^\bullet \vdash \lambda x{:}\, X^\bullet.\, s^\bullet\colon X^\bullet \to Y^\bullet \\
(\Theta \vdash s\, t\colon Y)^\bullet & = & \Theta^\bullet \vdash s^\bullet\, t^\bullet\colon Y^\bullet
\end{array}
$$

Translation $\circ : \lambda \to \lambda_l$

$$
\begin{array}{rcl}
K^\circ & = & L_K \\
1^\circ & = & 1 \\
(X \times Y)^\circ & = & X^\circ \,\&\, Y^\circ \\
(X \to Y)^\circ & = & !X^\circ \multimap Y^\circ \\[1ex]
(\Theta \vdash t : X)^\circ & = & !\Theta^\circ \vdash t^\circ : X^\circ \\[1ex]
(\vec{x_i}{:}\, \Theta, x{:}\, X \vdash x{:}\, X)^\circ & = & \vec{x_i}{:}\, !\Theta^\circ, x{:}\, !X^\circ \vdash \mathsf{discard}\ \vec{x_i}\ \mathsf{in}\ \mathsf{derelict}\ x{:}\, X^\circ \\
(\Theta \vdash ()\colon 1)^\circ & = & !\Theta^\circ \vdash ()\colon 1 \\
(\Theta \vdash (s,t)\colon X \times Y)^\circ & = & !\Theta^\circ \vdash (s^\circ, t^\circ)\colon X^\circ \,\&\, Y^\circ \\
(\Theta \vdash \mathrm{fst}\, s\colon X)^\circ & = & !\Theta^\circ \vdash \mathrm{fst}\, s^\circ\colon X^\circ \\
(\Theta \vdash \mathrm{snd}\, s\colon Y)^\circ & = & !\Theta^\circ \vdash \mathrm{snd}\, s^\circ\colon Y^\circ \\
(\Theta \vdash \lambda x{:}\, X.\, s\colon X \to Y)^\circ & = & !\Theta^\circ \vdash \lambda x{:}\, X^\circ.\, s^\circ\colon !X^\circ \multimap Y^\circ \\
(\vec{x_i}{:}\, \Theta \vdash s\, t\colon Y)^\circ & = & \vec{x_i}{:}\, !\Theta^\circ \vdash \mathsf{copy}\ \vec{x_i}\ \mathsf{as}\ \vec{a_i}, \vec{b_i}\ \mathsf{in}\ s^\circ[\vec{a_i}/\vec{x_i}]\ \mathsf{promote}\ \vec{b_i}\ \mathsf{for}\ \vec{x_i}\ \mathsf{in}\ t^\circ\colon Y^\circ
\end{array}
$$

Figure 9: Direct translations

Throughout, we assume that translations extend pointwise to contexts. So if Θ is $x_1 : X_1, \ldots, x_n : X_n$, then $\Theta^\bullet$ is $x_1 : X_1^\bullet, \ldots, x_n : X_n^\bullet$.

Each of the translations that we consider both preserves and reflects typing judgements:

Proposition 6.0.5

1. *If* $\Theta \vdash t : X$ *in* λ *then* $\Theta^\bullet \vdash t^\bullet : X^\bullet$ *in* λ_m.

2. *If* $\Theta^\bullet \vdash s : X^\bullet$ *in* λ_m *then* $\Theta \vdash |s|_m : X$ *in* λ.

3. *The composite* $|\text{-}^\bullet|_m$ *is the identity on* λ *typing judgements.*

Proof. Trivial, as the translation is essentially the identity. □

The corresponding translation into the linear calculus is $\circ : \lambda \to \lambda_l$, also presented in Figure 9, where, for exampe, $\mathsf{discard}\ \vec{e_i}\ \mathsf{in}\ f$ is shorthand for

$$\mathsf{discard}\ e_1\ \mathsf{in}\ \mathsf{discard}\ e_2\ \mathsf{in}\ \ldots\ \mathsf{discard}\ e_n\ \mathsf{in}\ f$$

This translation is essentially the same as Girard's original $\circ$ translation of intuitionistic logic into linear logic. Each variable is taken as having an 'of course' type, since it may be accessed an arbitrary number of times. Hence '!' is added to the left of the turntstile, and to the left of a function arrow. The product $\times$ of lambda calculus is interpreted by the product & of linear logic.

Again, the translation preserves and reflects typing judgements:

Proposition 6.0.6

1. *If* $\Theta \vdash t : X$ *in* λ *then* $!\Theta^\circ \vdash t^\circ : X^\circ$ *in* λ_l.

2. *If* $!\Theta^\circ \vdash e : X^\circ$ *in* λ_l *then* $\Theta \vdash |e|_l : X$ *in* λ.

3. *The composite* $|\text{-}^\circ|_l$ *is the identity on* λ *typing judgements.*

Proof. The proof is straightforward, if not completely trivial. The proof for each translation is similar, so it's worth sketching this one in a little detail.

For the first part, induct on the structure of lambda term t, and hence on the structure of the derivation of $\Theta \vdash t : X$.

For the second part, show that $|X^\circ|_l = X$ for every lambda type X, and that if $\Gamma \vdash e : A$ in λ_l then $|\Gamma|_l \vdash |e|_l : |A|_l$ in λ; by induction on X and e respectively. It follows that $!\Gamma^\circ \vdash e : X^\circ$ implies $|!\Gamma^\circ|_l \vdash |e|_l : |X^\circ|_l$ which is $\Gamma \vdash |e|_l : X$.

The third part then follows once one has shown $|t^\circ|_l = t$ for every lambda term t by induction on t. □

The two translations yield equivalent semantics. This is hardly surprising, as it was the motivation behind Girard's formulation, and has been a touchstone of the categorical models of Seely [See89] and Benton, Bierman, de Paiva, and Hyland [BBdPH92]. We spell out the equivalence here, for comparison with the later development.

The equivalence has two parts: the two translations of types are isomorphic, and the two translations of terms are related by this isomorphism. This level of detail is skipped over in most presentations, though it can be found in, for example, [Brau94].

First, the two translations of types are related. Each type X in λ maps to a type X° in λ_l, corresponding to an object in $\mathcal{L}$, and to a type $X^\bullet$ in λ_m, corresponding to an object in $\mathcal{C}$. How should these be related?

The answer is that $GX^\circ \simeq X^\bullet$ for each lambda calculus type X, assuming that $GK^\circ \simeq K^\bullet$ for each base type K. For the unit type, this follows because G has a left adjoint and so $G1 \simeq 1$ as G preserves limits. For the compound types, a simple inductive calculation gives:

$$
\begin{array}{cl}
 & G(X \times Y)^\circ \\
= & \qquad \{ \text{ def'n } \circ \text{ } \} \\
 & G(X^\circ \mathbin{\&} Y^\circ) \\
\simeq & \qquad \{ \text{ } G \text{ preserves limits } \} \\
 & GX^\circ \times GY^\circ \\
\simeq & \qquad \{ \text{ inductive hypothesis } \} \\
 & X^\bullet \times Y^\bullet \\
= & \qquad \{ \text{ def'n } \bullet \text{ } \} \\
 & (X \times Y)^\bullet,
\end{array}
\qquad\qquad
\begin{array}{cl}
 & G(X \to Y)^\circ \\
= & \qquad \{ \text{ def'n } \circ \text{ } \} \\
 & G(FGX^\circ \multimap Y^\circ) \\
\simeq & \qquad \{ \text{ relation of } \to \text{ to } \multimap \text{ } \} \\
 & GX^\circ \to GY^\circ \\
\simeq & \qquad \{ \text{ inductive hypothesis } \} \\
 & X^\bullet \to Y^\bullet \\
= & \qquad \{ \text{ def'n } \bullet \text{ } \} \\
 & (X \to Y)^\bullet.
\end{array}
$$

Second, the two translations of terms are, up to this isomorphism, mates across the adjunction in the categorical model. Each typed term $\Theta \vdash t : X$ in λ maps to a term $!\Theta^\circ \vdash t^\circ : X^\circ$ in λ_l, corresponding to an arrow in $\mathcal{L}$, and to a term $\Theta^\bullet \vdash t^\bullet : X^\bullet$ in λ_m, corresponding to an arrow in $\mathcal{C}$.

Now the heart of the matter: to relate these two arrows, write φ for the top-to-bottom composite of the following chain of natural isomorphisms between homsets (assuming Θ is $X_1, \ldots, X_n$):

$$
\begin{array}{cl}
 & \hom_{\mathcal{C}}(X_1^\bullet \times \cdots \times X_n^\bullet,\, X^\bullet) \\
\simeq & \qquad \{ \text{ relation of } X^\bullet \text{ to } X^\circ \text{ } \} \\
 & \hom_{\mathcal{C}}(GX_1^\circ \times \cdots \times GX_n^\circ,\, GX^\circ) \\
\simeq & \qquad \{ \text{ adjunction } \} \\
 & \hom_{\mathcal{L}}(F(GX_1^\circ \times \cdots \times GX_n^\circ),\, X^\circ) \\
\simeq & \qquad \{ \text{ } F \text{ is strong } \} \\
 & \hom_{\mathcal{L}}(FGX_1^\circ \otimes \cdots \otimes FGX_n^\circ,\, X^\circ)
\end{array}
$$

and note that the interpretations of $t^\bullet$ and t° live in the homsets at the top and bottom of the chain, respectively. It is a relatively simple induction on t to show that

$$\varphi([\![\Theta^\bullet \vdash t^\bullet : X^\bullet]\!]) \qquad = \qquad [\![!\Theta^\circ \vdash t^\circ : X^\circ]\!]$$

or, of course, equivalently

$$[\![\Theta^\bullet \vdash t^\bullet : X^\bullet]\!] \qquad = \qquad \varphi^{-1}([\![!\Theta^\circ \vdash t^\circ : X^\circ]\!])$$

which is the relation we were seeking.

The preceding discussion may also be expressed as follows.

Proposition 6.0.7 *(Direct translations are related.) Assume for each base type K that there is an isomorphism $i_K : GK^\circ \to K^\bullet$.*

- *For each type X in λ, the translation X° in λ_l, when transposed by functor G from $\mathcal{L}$ to $\mathcal{C}$, is isomorphic to the translation $X^\bullet$ in λ_m; that is, there is an isomorphism*
$$i_X : GX^\circ \to X^\bullet.$$
- *For each typed term $\Theta \vdash t : X$ in λ, the translation $!\Theta^\circ \vdash t^\circ : X^\circ$ in λ_l, when transposed across the adjunction $\langle F, G, \eta, \epsilon\rangle$ from $\mathcal{L}$ to $\mathcal{C}$, relates by isomorphism i to the translation $\Theta^\bullet \vdash t^\bullet : X^\bullet$ in λ_m,*

$$FG\Theta^\circ \xrightarrow{t^\circ} X^\circ$$

$$\begin{tikzcd}
G\Theta^\circ \arrow[r, "\eta\,;\,Gt^\circ"] \arrow[d, "i_\Theta"'] & GX^\circ \arrow[d, "i_X"] \\
\Theta^\bullet \arrow[r, "t^\bullet"'] & X^\bullet.
\end{tikzcd}$$

Proof. For types, see above. For terms, induct over term structure. □

Having examined this pair of translations in some detail, the remaining pairs of translations will prove straightforward.

7 Lifted translations

Our second lambda calculus has a semantics given by Moggi's call-by-name translation, which is implicit in Moggi's original work [Mog89], and has been made explicit by, for example, Wadler [Wad92]. This semantics is analogous to Reynold's original call-by-name continuation passing style transformation, as described by Plotkin [Plo75]; and to Abramksy's lazy lambda calculus [Abr92]. The corresponding linear translation appears to be new, and is presented here for the first time.

The lifted call-by-name monad translation, $\dagger : \lambda_\perp \to \lambda_m$, is presented in Figure 10. This translation is satisfyingly symmetric: each layer of type structure introduces a monad constructor. Under call-by-name, each variable and each term corresponds to a computation, so the same translation applies to both the left and right of the turnstile, and the left and right of the function arrow.

The translation preserves and reflects typing judgements:

Proposition 7.0.8

1. *If $\Theta \vdash t : X$ in $\lambda_\perp$ then $\Theta^\dagger \vdash t^\dagger : X^\dagger$ in λ_m.*
2. *If $\Theta^\dagger \vdash s : X^\dagger$ in λ_m then $\Theta \vdash |s|_m : X$ in $\lambda_\perp$.*

Translation $\dagger : \lambda_\perp \to \lambda_m$

$$
\begin{array}{rcl}
K^\dagger & = & TK \\
1^\dagger & = & T1 \\
(X \times Y)^\dagger & = & T(X^\dagger \times Y^\dagger) \\
(X \to Y)^\dagger & = & T(X^\dagger \to Y^\dagger) \\[1ex]
(\Theta \vdash t : X)^\dagger & = & \Theta^\dagger \vdash t^\dagger : X^\dagger \\[1ex]
(\Theta, x\colon X \vdash x\colon X)^\dagger & = & \Theta^\dagger, x\colon X^\dagger \vdash x\colon X^\dagger \\
(\Theta \vdash ()\colon 1)^\dagger & = & \Theta^\dagger \vdash [()]\colon T1 \\
(\Theta \vdash (s,t)\colon X \times Y)^\dagger & = & \Theta^\dagger \vdash [(s^\dagger, t^\dagger)]\colon T(X^\dagger \times Y^\dagger) \\
(\Theta \vdash \mathrm{fst}\ s\colon X)^\dagger & = & \Theta^\dagger \vdash \mathrm{let}\ z \leftarrow s^\dagger\ \mathrm{in}\ \mathrm{fst}\ z\colon X^\dagger \\
(\Theta \vdash \mathrm{snd}\ s\colon Y)^\dagger & = & \Theta^\dagger \vdash \mathrm{let}\ z \leftarrow s^\dagger\ \mathrm{in}\ \mathrm{snd}\ z\colon Y^\dagger \\
(\Theta \vdash \lambda x\colon X.\, u\colon X \to Y)^\dagger & = & \Theta^\dagger \vdash [(\lambda x\colon X^\dagger.\, u^\dagger)]\colon T(X^\dagger \to Y^\dagger) \\
(\Theta \vdash s\, t\colon Y)^\dagger & = & \Theta^\dagger \vdash \mathrm{let}\ z \leftarrow s^\dagger\ \mathrm{in}\ z\, t^\dagger\colon Y^\dagger
\end{array}
$$

Translation $\ddagger : \lambda_\perp \to \lambda_l$

$$
\begin{array}{rcl}
K^\ddagger & = & L_K \\
1^\ddagger & = & I \\
(X \times Y)^\ddagger & = & !X^\ddagger \otimes !Y^\ddagger \\
(X \to Y)^\ddagger & = & !(!X^\ddagger \multimap Y^\ddagger) \\[1ex]
(\Theta \vdash t : X)^\ddagger & = & !\Theta^\ddagger \vdash t^\ddagger : X^\ddagger \\[1ex]
(\vec{x_i}\colon \Theta, x\colon X \vdash x\colon X)^\ddagger & = & \vec{x_i}\colon !\Theta^\ddagger, x\colon !X^\ddagger \vdash \mathsf{discard}\ \vec{x_i}\ \mathsf{in}\ \mathsf{derelict}\ x\colon X^\ddagger \\
(\vec{x_i}\colon \Theta \vdash ()\colon 1)^\ddagger & = & \vec{x_i}\colon !\Theta^\ddagger \vdash \mathsf{discard}\ \vec{x_i}\ \mathsf{in}\ *\colon I \\
(\vec{x_i}\colon \Theta \vdash (s,t)\colon X \times Y)^\ddagger & = & \vec{x_i}\colon !\Theta^\ddagger \vdash \mathsf{copy}\ \vec{x_i}\ \mathsf{as}\ \vec{a_i}, \vec{b_i}\ \mathsf{in}\ (\mathsf{promote}\ \vec{a_i}\ \mathsf{for}\ \vec{x_i}\ \mathsf{in}\ s^\ddagger) \otimes \\
 & & \qquad (\mathsf{promote}\ \vec{a_i}\ \mathsf{for}\ \vec{x_i}\ \mathsf{in}\ s^\ddagger)\ \colon !X^\ddagger \otimes !Y^\ddagger \\
(\Theta \vdash \mathrm{fst}\ s\colon X)^\ddagger & = & !\Theta^\ddagger \vdash \mathrm{let}\ a \otimes b = s^\ddagger\ \mathrm{in}\ \mathsf{discard}\ b\ \mathsf{in}\ \mathsf{derelict}\ a\colon X^\ddagger \\
(\Theta \vdash \mathrm{snd}\ s\colon Y)^\ddagger & = & !\Theta^\ddagger \vdash \mathrm{let}\ a \otimes b = s^\ddagger\ \mathrm{in}\ \mathsf{discard}\ a\ \mathsf{in}\ \mathsf{derelict}\ b\colon Y^\ddagger \\
(\vec{x_i}\colon \Theta \vdash \lambda x\colon X.\, s\colon X \to Y)^\ddagger & = & \vec{x_i}\colon !\Theta^\ddagger \vdash \mathsf{promote}\ \vec{x_i}\ \mathsf{for}\ \vec{a_i}\ \mathsf{in}\ (\lambda x\colon !X^\ddagger.\, e)[\vec{a_i}/\vec{x_i}]\colon !(!X^\ddagger \multimap Y^\ddagger) \\
(\vec{x_i}\colon \Theta \vdash s\, t\colon Y)^\ddagger & = & \vec{x_i}\colon !\Theta^\ddagger \vdash \mathsf{copy}\ \vec{x_i}\ \mathsf{as}\ \vec{a_i}, \vec{b_i}\ \mathsf{in}\ (\mathsf{derelict}\ s^\ddagger[\vec{a_i}/\vec{x_i}]) \\
 & & \qquad (\mathsf{promote}\ \vec{b_i}\ \mathsf{for}\ \vec{x_i}\ \mathsf{in}\ t^\ddagger) : Y^\ddagger
\end{array}
$$

Figure 10: Lifted call-by-name translations

3. *The composite* $|\text{-}^{\dagger}|_m$ *is the identity on* $\lambda_\perp$ *typing judgements.*

Proof. The proof is similar to that of Proposition 6.0.6. $\square$

The corresponding translation into the linear calculus is $\ddagger : \lambda_\perp \to \lambda_l$, also presented in Figure 10. This translation is similar to Girard's original $\circ$ translation of the preceding section. Again, '!' is added to the left of the turnstile, and to the left of a function arrow. The translation for base types is L_K, as in the previous translation. The translation for product types is $!X^\ddagger \otimes !Y^\ddagger$, and this is isomorphic to $!(X^\ddagger \mathbin{\&} Y^\ddagger)$, which has an extra '!' at the outside compared to the previous translation. Similarly, the translation of function types has an extra '!' at the outside compared to the previous translation.

As always, the translation preserves and reflects typing judgements:

Proposition 7.0.9

1. *If* $\Theta \vdash t : X$ *in* $\lambda_\perp$ *then* $!\Theta^\ddagger \vdash t^\ddagger : X^\circ$ *in* λ_l.
2. *If* $!\Theta^\ddagger \vdash e : X^\ddagger$ *in* λ_l *then* $\Theta \vdash |e|_l : X$ *in* $\lambda_\perp$.
3. *The composite* $|\text{-}^\ddagger|_l$ *is the identity on* $\lambda_\perp$ *typing judgements.*

$\square$

Comparing the monad translation with the linear translation, the monad translation is the same on the left and the right of a function arrow, while the linear translation adds an extra '!' to the left of a function arrow. Comparing the new translation with the old translation, the 'T's added to the monad translation are balanced by the '!'s added to the linear translation. Again, the two translations are equivalent, and this equivalence has two parts.

First, we again have that the translations yield related types, assuming that the translations of base types are related. That is, $GX^\ddagger \simeq X^\dagger$ for each lambda calculus type X, assuming that $GK^\ddagger \simeq K^\dagger$ for each base type K. For the unit type this means $GI \simeq GF1$, which follows from the fact that $I \simeq F1$. For compound types, the result follows from a small modification to the previous calculation:

$$
\begin{array}{cl}
 & G(X \times Y)^\ddagger \\
= & \quad \{ \text{ def'n } \ddagger \ \} \\
 & G(FGX^\ddagger \otimes FGY^\ddagger) \\
\simeq & \quad \{ \text{ relation of } \otimes \text{ to } \times \ \} \\
 & GF(GX^\ddagger \times GY^\ddagger) \\
\simeq & \quad \{ \text{ inductive hypothesis } \} \\
 & GF(X^\dagger \times Y^\dagger) \\
= & \quad \{ \text{ def'n } \dagger \ \} \\
 & (X \times Y)^\dagger,
\end{array}
\qquad
\begin{array}{cl}
 & G(X \to Y)^\ddagger \\
= & \quad \{ \text{ def'n } \ddagger \ \} \\
 & GFG(FGX^\ddagger \multimap Y^\ddagger) \\
\simeq & \quad \{ \text{ relation of } \to \text{ to } \multimap \ \} \\
 & GF(GX^\ddagger \to GY^\ddagger) \\
\simeq & \quad \{ \text{ inductive hypothesis } \} \\
 & GF(X^\dagger \to Y^\dagger) \\
= & \quad \{ \text{ def'n } \dagger \ \} \\
 & (X \to Y)^\dagger.
\end{array}
$$

Second, translations of terms are mates. If we write ϕ for the top-to-bottom composite of the following chain of natural isomorphisms between homsets:

$$\begin{array}{ll}
 & \hom_{\mathcal{C}}(X_1^\dagger \times \cdots \times X_n^\dagger,\, X^\dagger) \\
\simeq & \qquad \{\text{ relation of } X^\ddagger \text{ to } X^\dagger \ \} \\
 & \hom_{\mathcal{C}}(GX_1^\ddagger \times \cdots \times GX_n^\ddagger,\, GX^\ddagger) \\
\simeq & \qquad \{\text{ adjunction } \} \\
 & \hom_{\mathcal{L}}(F(GX_1^\ddagger \times \cdots \times GX_n^\ddagger),\, X^\ddagger) \\
\simeq & \qquad \{\ F \text{ is strong } \} \\
 & \hom_{\mathcal{L}}(FGX_1^\ddagger \otimes \cdots \otimes FGX_n^\ddagger,\, X^\ddagger)
\end{array}$$

then the result we want is that

$$\phi([\![\Theta^\dagger \vdash t^\dagger : X^\dagger]\!]) \qquad = \qquad [\![!\Theta^\ddagger \vdash t^\ddagger : X^\ddagger]\!]$$

which follows by induction on t.

Again, we may spell this out as in the following proposition, which is a straightforward modification of the previous result.

Proposition 7.0.10 *(Lifted call-by-name translations are related.) Assume for each base type K that there is an isomorphism $j_K : GK^\ddagger \to K^\dagger$.*

- *For each type X in $\lambda_\perp$, the translation $X^\ddagger$ in λ_l, when transposed by functor G from $\mathcal{L}$ to $\mathcal{C}$, is isomorphic to the translation $X^\dagger$ in λ_m; that is, there is an isomorphism*
$$j_X : GX^\ddagger \to X^\dagger.$$
- *For each typed term $\Theta \vdash t : X$ in $\lambda_\perp$, the translation $!\Theta^\ddagger \vdash t^\ddagger : X^\ddagger$ in λ_l, when transposed across the adjunction $\langle F, G, \eta, \epsilon\rangle$ from $\mathcal{L}$ to $\mathcal{C}$, relates by isomorphism j to the translation $\Theta^\dagger \vdash t^\dagger : X^\dagger$ in λ_m,*

$$FG\Theta^\ddagger \xrightarrow{t^\ddagger} X^\ddagger$$

$$\begin{tikzcd}
G\Theta^\ddagger \arrow[r, "\eta\,;\,Gt^\ddagger"] \arrow[d, "j_\Theta"'] & GX^\ddagger \arrow[d, "j_X"] \\
\Theta^\dagger \arrow[r, "t^\dagger"'] & X^\dagger.
\end{tikzcd}$$

Proof. For types, see above. For terms, induct over term structure. □

8 Call-by-value translations

Our third lambda calculus is Plotkin's call-by-value lambda calculus [Plo75]. The monad translation appears in Moggi's original work on monads [Mog89], and the linear translation appears in Girard's orginal work on linear logic [Gir87]. But

Translation $\star : \lambda_v \to \lambda_m$

$$
\begin{array}{rcl}
K^\star & = & K \\
1^\star & = & 1 \\
(X \times Y)^\star & = & X^\star \times Y^\star \\
(X \to Y)^\star & = & X^\star \to TY^\star \\
\\
(\Theta \vdash t : X)^\star & = & \Theta^\star \vdash t^\star : TX^\star \\
\\
(\Theta, x{:}\, X \vdash x{:}\, X)^\star & = & \Theta^\star, x{:}\, X^\star \vdash [x]{:}\, TX^\star \\
(\Theta \vdash ()\colon 1)^\star & = & \Theta^\star \vdash [()]{:}\, T1 \\
(\Theta \vdash (s,t){:}\, X \times Y)^\star & = & \Theta^\star \vdash \text{let } x \leftarrow s^\star \text{ in let } y \leftarrow t^\star \text{ in } [(x,y)]{:}\, T(X^\star \times Y^\star) \\
(\Theta \vdash \text{fst } s{:}\, X)^\star & = & \Theta^\star \vdash \text{let } z \leftarrow s^\star \text{ in } [\text{fst } z]{:}\, TX^\star \\
(\Theta \vdash \text{snd } s{:}\, Y)^\star & = & \Theta^\star \vdash \text{let } z \leftarrow s^\star \text{ in } [\text{snd } z]{:}\, TY^\star \\
(\Theta \vdash \lambda x{:}\, X.\, s{:}\, X \to Y)^\star & = & \Theta^\star \vdash [(\lambda x{:}\, X^\star.\, s^\star)]{:}\, T(X^\star \to TY^\star) \\
(\Theta \vdash s\, t{:}\, Y)^\star & = & \Theta^\star \vdash \text{let } z \leftarrow s^\star \text{ in let } x \leftarrow t^\star \text{ in } z\, x{:}\, TY^\star
\end{array}
$$

Translation $* : \lambda_v \to \lambda_l$

$$
\begin{array}{rcl}
K^* & = & !L_K \\
1^* & = & I \\
(X \times Y)^* & = & X^* \otimes Y^* \\
(X \to Y)^* & = & !(X^* \multimap Y^*) \\
\\
(\Theta \vdash t : X)^* & = & \Theta^* \vdash t^* : X^* \\
\\
(\vec{x_i}{:}\, \vec{X_i}, x{:}\, X \vdash x{:}\, X)^* & = & \vec{x_i}{:}\, \vec{X}_i^*, x{:}\, X^* \vdash \mathit{discard}_{\vec{X_i}}\ \vec{x_i}\ \mathit{in}\ x{:}\, X^* \\
(\vec{x_i}{:}\, \vec{X_i} \vdash ()\colon 1)^* & = & \vec{x_i}{:}\, \vec{X}_i^* \vdash \mathit{discard}_{\vec{X_i}}\ \vec{x_i}\ \mathit{in}\ *{:}\, I \\
(\vec{x_i}{:}\, \vec{X_i} \vdash (s,t){:}\, X \times Y)^* & = & \vec{x_i}{:}\, \vec{X}_i^* \vdash \mathit{copy}_{\vec{X_i}}\ \vec{x_i}\ \mathit{as}\ \vec{a_i}, \vec{b_i}\ \mathit{in}\ s^*[\vec{a_i}/\vec{x_i}] \otimes t^*[\vec{b_i}/\vec{x_i}]{:}\, X^* \otimes Y^* \\
(\vec{x_i}{:}\, \vec{X_i} \vdash \text{fst } s{:}\, X)^* & = & \vec{x_i}{:}\, \vec{X}_i^* \vdash \text{let } a \otimes b = s^* \text{ in } \mathit{discard}_Y\ b\ \mathit{in}\ a{:}\, X^* \\
(\vec{x_i}{:}\, \vec{X_i} \vdash \text{snd } s{:}\, Y)^* & = & \vec{x_i}{:}\, \vec{X}_i^* \vdash \text{let } a \otimes b = s^* \text{ in } \mathit{discard}_X\ a\ \mathit{in}\ b{:}\, Y^* \\
(\vec{x_i}{:}\, \vec{X_i} \vdash (\lambda x{:}\, X.\, s){:}\, X \to Y)^* & = & \vec{x_i}{:}\, \vec{X}_i^* \vdash \mathit{PROM}\ (\text{-}; \vec{x_i}{:}\, \vec{X}_i^* \rhd \lambda x{:}\, X^*.\, s^*){:}\, !(X^* \multimap Y^*) \\
(\vec{x_i}{:}\, \vec{X_i} \vdash s\, t{:}\, Y)^* & = & \vec{x_i}{:}\, \vec{X}_i^* \vdash \mathit{copy}_{\vec{X_i}}\ \vec{x_i}\ \mathit{as}\ \vec{a_i}, \vec{b_i}\ \mathit{in}\ (\mathsf{derelict}\ s^*[\vec{a_i}/\vec{x_i}])\ t^*{:}\, Y^*
\end{array}
$$

Figure 11: Call-by-value translations

while the monad translation is arguably the centrepiece of Moggi's work, Girard commented that his translation "was not of much interest". The correspondence of Girard's translation to call-by-value is discussed by Mackie [Mac94b] and Maraist *et al.* [MOTW95].

The call-by-value monad translation, $\star : \lambda_v \to \lambda_m$, is presented in Figure 11. Under call-by-value, each variable corresponds to a value while the term corresponds to a computation. Hence, a monad constructor T is inserted to the right of a turnstile and the right of a function arrow, but nowhere else.

The translation preserves and reflects typing judgements.

Proposition 8.0.11

1. *If* $\Theta \vdash t : X$ *in* λ_v *then* $\Theta^\star \vdash t^\star : TX^\star$ *in* λ_m.
2. *If* $\Theta^\star \vdash s : TX^\star$ *in* λ_m *then* $\Theta \vdash |s|_m : X$ *in* λ_v.
3. *The composite* $|\text{-}^\star|_m$ *is the identity on* λ_v *typing judgements.*

$\square$

The corresponding translation into the linear calculus is $* : \lambda_v \to \lambda_l$, also presented in Figure 11. Here, '!' is added uniformly, just as in the call-by-name translation into the monad calculus.

The $*$ translation leads to rather complicated linear terms whichever variant of the linear term calculus one chooses to use. This is because the types of variables in translated contexts are not all ! types and yet we must still be able to copy and discard them. This is possible, since the types of free variables are all ultimately constructed out of !ed base types, but the details are fairly tricky. We have made use of a number of abbreviations for λ_l terms in Figure 11, which we now explain.

The abbreviations *discard* and *copy* are indexed by types of λ_v and are defined inductively as follows:

$$
\begin{array}{rcl}
\mathit{discard}_K\ e\ \mathit{in}\ f & = & \mathsf{discard}\ e\ \mathsf{in}\ f \\
\mathit{discard}_1\ e\ \mathit{in}\ f & = & \mathrm{let}\ * = e\ \mathrm{in}\ f \\
\mathit{discard}_{X\times Y}\ e\ \mathit{in}\ f & = & \mathrm{let}\ a \otimes b = e\ \mathrm{in}\ \mathit{discard}_X\ a\ \mathit{in}\ \mathit{discard}_Y\ b\ \mathit{in}\ f \\
\mathit{discard}_{X\to Y}\ e\ \mathit{in}\ f & = & \mathsf{discard}\ e\ \mathsf{in}\ f \\
\\
\mathit{copy}_1\ e\ \mathit{as}\ a, b\ \mathit{in}\ f & = & \mathrm{let}\ * = e\ \mathrm{in}\ f[*/a, */b] \\
\mathit{copy}_K\ e\ \mathit{as}\ a, b\ \mathit{in}\ f & = & \mathsf{copy}\ e\ \mathsf{as}\ a, b\ \mathsf{in}\ f \\
\mathit{copy}_{X\times Y}\ e\ \mathit{as}\ a, b\ \mathit{in}\ f & = & \mathrm{let}\ c \otimes d = e\ \mathrm{in}\ \mathit{copy}_X\ c\ \mathit{as}\ c_1, c_2\ \mathit{in} \\
 & & \qquad \mathit{copy}_Y\ d\ \mathit{as}\ d_1, d_2\ \mathit{in}\ f[c_1 \otimes d_1/a, c_2 \otimes d_2/b] \\
\mathit{copy}_{X\to Y}\ e\ \mathit{as}\ a, b\ \mathit{in}\ f & = & \mathsf{copy}\ e\ \mathsf{as}\ a, b\ \mathsf{in}\ f
\end{array}
$$

The abbreviation *PROM* is necessary to cope with our desire to promote the translations of λ_v functions when the free variables of the translations are not all

!ed. *PROM* is defined inductively as follows:

$$\begin{array}{rcl}
PROM\ (\Delta;\Gamma, a{:}\, I\ \rhd h) & = & \text{let } * = a \text{ in } PROM\ (\Delta;\Gamma\ \rhd h[*/a]) \\
PROM\ (\Delta;\Gamma, a{:}\, A\otimes B\ \rhd h) & = & \text{let } c\otimes d = a \text{ in } PROM\ (\Delta;\Gamma, c{:}\, A, d{:}\, B\ \rhd h[c\otimes d/a]) \\
PROM\ (\Delta;\Gamma, a{:}\, !A\ \rhd h) & = & PROM\ (\Delta, a{:}\, !A;\Gamma\ \rhd h) \\
PROM\ (\vec{a_i}{:}\, !\vec{A}_i; \text{-}\ \rhd h) & = & \mathsf{promote}\ \vec{a_i}\ \mathsf{for}\ \vec{z_i}\ \mathsf{in}\ t[\vec{z_i}/\vec{a_i}]
\end{array}$$

Again, the translation preserves and reflects typing judgements.

Proposition 8.0.12

1. *If* $\Theta \vdash t : X$ *in* λ *then* $\Theta^* \vdash t^* : X^*$ *in* λ_l.
2. *If* $\Theta^* \vdash e : X^*$ *in* λ_l *then* $\Theta \vdash |e|_l : X$ *in* λ_v.
3. *The composite* $|\text{-}^*|_l$ *is the identity on* λ_v *typing judgements.*

□

And again, the two translations yield equivalent semantics. Whereas before the isomorphism was achieved by using G to move the linear type from $\mathcal{L}$ to $\mathcal{C}$, here the isomorphism is achieved by using F to move the monad type from $\mathcal{C}$ to $\mathcal{L}$. So now we have that if $K^* \simeq FK^\star$ for each base type K, then $X^* \simeq FX^\star$ for each lambda calculus type X. For the unit type this holds because $I \simeq F1$, and for compound types it follows by another simple inductive calculation:

$$\begin{array}{cl}
 & (X\times Y)^* \\
= & \quad \{ \text{ def'n } * \ \} \\
 & X^*\otimes Y^* \\
\simeq & \quad \{ \text{ inductive hypothesis } \} \\
 & FX^\star\otimes FY^\star \\
\simeq & \quad \{ \text{ relation of } \otimes \text{ to } \times \ \} \\
 & F(X^\star\times Y^\star) \\
= & \quad \{ \text{ def'n } \star \ \} \\
 & F(X\times Y)^\star,
\end{array}
\qquad
\begin{array}{cl}
 & (X\to Y)^* \\
= & \quad \{ \text{ def'n } * \ \} \\
 & FG(X^*\multimap Y^*) \\
\simeq & \quad \{ \text{ inductive hypothesis } \} \\
 & FG(FX^\star\multimap FY^\star) \\
\simeq & \quad \{ \text{ relatation of } \multimap \text{ to } \to \ \} \\
 & F(X^\star\to GFY^\star) \\
= & \quad \{ \text{ def'n } \star \ \} \\
 & F(X\to Y)^\star.
\end{array}$$

Again, translations of terms are mates across the adjunction. Let ψ denote the top-to-bottom composite of the following chain of natural bijections between homsets:

$$\begin{array}{cl}
 & \hom_{\mathcal{C}}(X_1^\star\times\cdots\times X_n^\star,\ GFX^\star) \\
\simeq & \quad \{ \text{ adjunction } \} \\
 & \hom_{\mathcal{L}}(F(X_1^\star\times\cdots\times X_n^\star),\ FX^\star) \\
\simeq & \quad \{ \ F \text{ is strong } \} \\
 & \hom_{\mathcal{L}}(FX_1^\star\otimes\cdots\otimes FX_n^\star,\ FX^\star) \\
\simeq & \quad \{ \text{ relation of } \star \text{ to } * \ \} \\
 & \hom_{\mathcal{L}}(X_1^*\otimes\cdots\otimes X_n^*,\ X^*)
\end{array}$$

and then the relation between the two translations is that

$$\psi([\![\Theta^\star\vdash t^\star : TX^\star]\!]) \qquad = \qquad [\![\Theta^*\vdash t^* : X^*]\!]$$

Again, we can state this result as follows:

Proposition 8.0.13 *(Call-by-value translations are related.) Assume for each base type K that there is an isomorphism $k_K : FK^\star \to K^*$.*

- *For each type X in λ, the translation $X^\star$ in λ_m, when transposed by functor F from $\mathcal{C}$ to $\mathcal{L}$, is isomorphic to the translation X^* in λ_l; that is, there is an isomorphism*
$$k_X : FX^\star \to X^*.$$
- *For each typed term $\Theta \vdash t : X$ in λ, the translation $\Theta^\star \vdash t^\star : TX^\star$ in λ_m, when transposed across the adjunction $\langle F, G, \eta, \epsilon \rangle$ from $\mathcal{C}$ to $\mathcal{L}$, relates by isomorphism k to the translation $\Theta^* \vdash t^* : X^*$ in λ_l,*

$$\Theta^\star \xrightarrow{t^\star} GFX^\star$$

$$\begin{tikzcd}
F\Theta^\star \arrow[r, "Ft^\star\,;\,\epsilon"] \arrow[d, "k_\Theta"'] & FX^\star \arrow[d, "k_X"] \\
\Theta^* \arrow[r, "t^*"'] & X^*.
\end{tikzcd}$$

Proof. For types, see above. For terms, induct over term structure. □

Part III
The adjoint calculus

9 The adjoint calculus

The adjoint calculus λ_a is presented in Figures 12 and 13. We have changed Benton's syntax in one particular, writing $G^{-1}e$ instead of $\mathsf{derelict}\ e$. Corresponding to each judgement

$$x_1 : X_1,\ \ldots,\ x_m : X_m \vdash_{\mathcal{C}} u : Y$$

is an arrow

$$X_1 \times \cdots \times X_n \xrightarrow{u}_{\mathcal{C}} Y$$

in $\mathcal{C}$ and corresponding to each judgment

$$x_1 : X_1,\ \ldots,\ x_m : X_m;\ b_1 : B_1,\ \ldots,\ b_n : B_n \vdash_{\mathcal{L}} h : C$$

is an arrow

$$F(X_1 \times \cdots \times X_m) \otimes B_1 \otimes \cdots \otimes B_n \xrightarrow{h}_{\mathcal{L}} C.$$

in $\mathcal{L}$. (We subscript the arrows to indicate the category.) Our interpretation is identical to Benton's, except we gain a little compactness by writing $F(X_1 \times \cdots \times X_m)$ where he writes the isomorphic $FX_1 \otimes \cdots \otimes FX_m$.

Syntax

atomic types (intuitionistic)	K		
atomic types (linear)	L		
variables (intuitionistic)	x, y, z		
variables (linear)	a, b, c		
types (intuitionistic)	X, Y, Z	::=	$K \mid 1_{\mathcal{C}} \mid A \times B \mid A \to B \mid GA$
types (linear)	A, B, C	::=	$L \mid I \mid 1_{\mathcal{L}} \mid A \,\&\, B \mid A \otimes B \mid A \multimap B \mid FX$
contexts (intuitionistic)	Θ	::=	$x_1 : X_1, \ldots, x_n : X_n$
contexts (linear)	Γ, Δ	::=	$a_1 : A_1, \ldots, a_n : A_n$
terms (intuitionistic)	s, t, u	::=	$x \mid ()_{\mathcal{C}} \mid (s, t) \mid \mathrm{fst}\ s \mid \mathrm{snd}\ s \mid \lambda x{:}\, X.\, s \mid s\, t \mid Ge$
terms (linear)	e, f, g, h	::=	$a \mid * \mid \mathrm{let}\ * = e\ \mathrm{in}\ f \mid e \otimes f \mid \mathrm{let}\ a \otimes b = e\ \mathrm{in}\ f$
			$\mid ()_{\mathcal{L}} \mid (e, f) \mid \mathrm{fst}\ e \mid \mathrm{snd}\ e$
			$\mid \lambda a{:}\, A.\, e \mid e\ f \mid Ft \mid \mathrm{let}\ Fx = e\ \mathrm{in}\ g \mid G^{-1}s$

Types

$$\mathrm{Id}_{\mathcal{C}}\ \frac{}{x_1 : X_1, \ldots, x_n : X_n \vdash_{\mathcal{C}} x_i : X_i} \qquad \mathrm{Id}_{\mathcal{L}}\ \frac{}{\Theta;\ a : A \vdash_{\mathcal{L}} a : A}$$

$$1_{\mathcal{L}}\text{-I}\ \frac{}{\Theta; \Gamma \vdash_{\mathcal{L}} ()_{\mathcal{L}} {:} 1_{\mathcal{L}}} \qquad 1_{\mathcal{C}}\text{-I}\ \frac{}{\Theta \vdash_{\mathcal{C}} ()_{\mathcal{C}} {:} 1_{\mathcal{C}}}$$

$$\times\text{-I}\ \frac{\Theta \vdash_{\mathcal{C}} t : X \qquad \Theta \vdash_{\mathcal{C}} u : Y}{\Theta \vdash_{\mathcal{C}} (t, u) : X \times Y} \qquad \times\text{-E}\ \frac{\Theta \vdash_{\mathcal{C}} s : X \times Y}{\Theta \vdash_{\mathcal{C}} \mathrm{fst}\ s : X} \qquad \frac{\Theta \vdash_{\mathcal{C}} s : X \times Y}{\Theta \vdash_{\mathcal{C}} \mathrm{snd}\ s : Y}$$

$$\&\text{-I}\ \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} f : A \quad \Theta;\ \Gamma \vdash_{\mathcal{L}} g : B}{\Theta;\ \Gamma \vdash_{\mathcal{L}} (f, g) : A \,\&\, B} \qquad \&\text{-E}\ \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e : A \,\&\, B}{\Theta;\ \Gamma \vdash_{\mathcal{L}} \mathrm{fst}\ e : A} \qquad \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e : A \,\&\, B}{\Theta;\ \Gamma \vdash_{\mathcal{L}} \mathrm{snd}\ e : B}$$

$$I\text{-I}\ \frac{}{\Theta;\ \text{-} \vdash_{\mathcal{L}} * {:} I} \qquad I\text{-E}\ \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e {:} I \qquad \Theta;\ \Delta \vdash_{\mathcal{L}} f {:} A}{\Theta;\ \Gamma, \Delta \vdash_{\mathcal{L}} \mathrm{let}\ * = e\ \mathrm{in}\ f {:} A}$$

$$\otimes\text{-I}\ \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} f : A \quad \Theta;\ \Delta \vdash_{\mathcal{L}} g : B}{\Theta;\ \Gamma, \Delta \vdash_{\mathcal{L}} f \otimes g : A \otimes B} \qquad \otimes\text{-E}\ \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e : A \otimes B \quad \Theta;\ a : A, b : B, \Delta \vdash_{\mathcal{L}} h : C}{\Theta;\ \Gamma, \Delta \vdash_{\mathcal{L}} \mathrm{let}\ a \otimes b = e\ \mathrm{in}\ h : C}$$

$$\to\text{-I}\ \frac{\Theta, x : X \vdash_{\mathcal{C}} u : Y}{\Theta \vdash_{\mathcal{C}} \lambda x{:}\, X.\, u : X \to Y} \qquad \to\text{-E}\ \frac{\Theta \vdash_{\mathcal{C}} s : X \to Y \qquad \Theta \vdash_{\mathcal{C}} t : X}{\Theta \vdash_{\mathcal{C}} s\, t : Y}$$

$$\multimap\text{-I}\ \frac{\Theta;\ \Gamma, a : A \vdash_{\mathcal{L}} g : B}{\Theta;\ \Gamma \vdash_{\mathcal{L}} \lambda a{:}\, A.\, g : A \multimap B} \qquad \multimap\text{-E}\ \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e : A \multimap B \qquad \Theta;\ \Delta \vdash_{\mathcal{L}} f : A}{\Theta;\ \Gamma, \Delta \vdash_{\mathcal{L}} e\ f : B}$$

$$G\text{-I}\ \frac{\Theta;\ \text{-} \vdash_{\mathcal{L}} f : A}{\Theta \vdash_{\mathcal{C}} Gf : GA} \qquad G\text{-E}\ \frac{\Theta \vdash_{\mathcal{C}} s : GA}{\Theta;\ \text{-} \vdash_{\mathcal{L}} G^{-1}s : A}$$

$$F\text{-I}\ \frac{\Theta \vdash_{\mathcal{C}} t : X}{\Theta;\ \text{-} \vdash_{\mathcal{L}} Ft : FX} \qquad F\text{-E}\ \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e : FX \qquad \Theta, x : X;\ \Delta \vdash_{\mathcal{L}} g : B}{\Theta;\ \Gamma, \Delta \vdash_{\mathcal{L}} \mathrm{let}\ Fx = e\ \mathrm{in}\ g : B}$$

Figure 12: Adjoint lambda calculus (λ_a)

$$\mathrm{Id}_{\mathcal{C}}\ \frac{}{X_1 \times \cdots \times X_n \xrightarrow{\mathrm{sel}_i}_{\mathcal{C}} X_i} \qquad \mathrm{Id}_{\mathcal{L}}\ \frac{}{F\Theta \otimes A \xrightarrow{\mathrm{disc}\otimes\mathrm{id}}_{\mathcal{L}} I \otimes A \simeq A}$$

$$\times\text{-I}\ \frac{\Theta \xrightarrow{t}_{\mathcal{C}} X \qquad \Theta \xrightarrow{u}_{\mathcal{C}} Y}{\Theta \xrightarrow{\langle t, u\rangle}_{\mathcal{C}} X \times Y} \qquad \times\text{-E}\ \frac{\Theta \xrightarrow{s}_{\mathcal{C}} X \times Y}{\Theta \xrightarrow{s}_{\mathcal{C}} X \times Y \xrightarrow{\mathrm{fst}}_{\mathcal{C}} X} \qquad \frac{\Theta \xrightarrow{s}_{\mathcal{C}} X \times Y}{\Theta \xrightarrow{s}_{\mathcal{C}} X \times Y \xrightarrow{\mathrm{snd}} Y}$$

$$\&\text{-I}\ \frac{F\Theta \otimes \Gamma \xrightarrow{f}_{\mathcal{L}} A \qquad F\Theta \otimes \Gamma \xrightarrow{g}_{\mathcal{L}} B}{F\Theta \otimes \Gamma \xrightarrow{\langle f, g\rangle}_{\mathcal{L}} A \,\&\, B} \qquad ()_{\mathcal{L}}\text{-I}\ \frac{}{F\Theta \otimes \Gamma \xrightarrow{\langle\rangle}_{\mathcal{L}} 1_{\mathcal{L}}} \qquad ()_{\mathcal{C}}\text{-I}\ \frac{}{\Theta \xrightarrow{\langle\rangle}_{\mathcal{C}} ()_{\mathcal{C}}}$$

$$\&\text{-E}\ \frac{F\Theta \otimes \Gamma \xrightarrow{e}_{\mathcal{L}} A \,\&\, B}{F\Theta \otimes \Gamma \xrightarrow{e}_{\mathcal{L}} A \,\&\, B \xrightarrow{\mathrm{fst}}_{\mathcal{L}} A} \qquad \frac{F\Theta \otimes \Gamma \xrightarrow{e}_{\mathcal{L}} A \,\&\, B}{F\Theta \otimes \Gamma \xrightarrow{e}_{\mathcal{L}} A \,\&\, B \xrightarrow{\mathrm{snd}}_{\mathcal{L}} B}$$

$$\otimes\text{-I}\ \frac{F\Theta \otimes \Gamma \xrightarrow{f}_{\mathcal{L}} A \qquad F\Theta \otimes \Delta \xrightarrow{g}_{\mathcal{L}} B}{F\Theta \otimes \Gamma \otimes \Delta \xrightarrow{\mathrm{dupl}\otimes\mathrm{id}\otimes\mathrm{id}}_{\mathcal{L}} F\Theta \otimes F\Theta \otimes \Gamma \otimes \Delta \simeq F\Theta \otimes \Gamma \otimes F\Theta \otimes \Delta \xrightarrow{f \otimes g}_{\mathcal{L}} A \otimes B}$$

$$\otimes\text{-E}\ \frac{F\Theta \otimes \Gamma \xrightarrow{e}_{\mathcal{L}} A \otimes B \quad F\Theta \otimes A \otimes B \otimes \Delta \xrightarrow{h}_{\mathcal{L}} C}{F\Theta \otimes \Gamma \otimes \Delta \xrightarrow{\mathrm{dupl}\otimes\mathrm{id}\otimes\mathrm{id}}_{\mathcal{L}} F\Theta \otimes F\Theta \otimes \Gamma \otimes \Delta \xrightarrow{\mathrm{id}\otimes e\otimes\mathrm{id}}_{\mathcal{L}} F\Theta \otimes A \otimes B \otimes \Delta \xrightarrow{h}_{\mathcal{L}} C}$$

$$\to\text{-I}\ \frac{\Theta \times X \xrightarrow{u}_{\mathcal{C}} Y}{\Theta \xrightarrow{\mathrm{cur}(u)}_{\mathcal{C}} X \to Y} \qquad \to\text{-E}\ \frac{\Theta \xrightarrow{s}_{\mathcal{C}} X \to Y \qquad \Theta \xrightarrow{t}_{\mathcal{C}} X}{\Theta \xrightarrow{\langle s, t\rangle}_{\mathcal{C}} (X \to Y) \times X \xrightarrow{\mathrm{app}}_{\mathcal{C}} Y}$$

$$I\text{-E}\ \frac{F\Theta \otimes \Gamma \xrightarrow{e}_{\mathcal{L}} I \qquad F\Theta \otimes \Delta \xrightarrow{f}_{\mathcal{L}} A}{F\Theta \otimes \Gamma \otimes \Delta \xrightarrow{\mathrm{dupl}\otimes\mathrm{id}}_{\mathcal{L}} F\Theta \otimes F\Theta \otimes \Gamma \otimes \Delta \xrightarrow{\mathrm{id}\otimes e\otimes\mathrm{id}}_{\mathcal{L}} F\Theta \otimes I \otimes \Delta \simeq F\Theta \otimes \Delta \xrightarrow{f}_{\mathcal{L}} A}$$

$$I\text{-I}\ \frac{}{F\Theta \xrightarrow{\mathrm{disc}}_{\mathcal{L}} I} \qquad \multimap\text{-I}\ \frac{F\Theta \otimes \Gamma \otimes A \xrightarrow{g}_{\mathcal{C}} B}{F\Theta \otimes \Gamma \xrightarrow{\mathrm{cur}_{\mathcal{L}}(g)}_{\mathcal{C}} A \multimap B}$$

$$\multimap\text{-E}\ \frac{F\Theta \otimes \Gamma \xrightarrow{e}_{\mathcal{C}} A \multimap B \qquad F\Theta \otimes \Delta \xrightarrow{f}_{\mathcal{C}} A}{F\Theta \otimes \Gamma \otimes \Delta \xrightarrow{\mathrm{dupl}\otimes\mathrm{id}\otimes\mathrm{id}}_{\mathcal{L}} F\Theta \otimes F\Theta \otimes \Gamma \otimes \Delta \simeq F\Theta \otimes \Gamma \otimes F\Theta \otimes \Delta \xrightarrow{e \otimes f}_{\mathcal{L}} (A \multimap B) \otimes A \xrightarrow{\mathrm{app}_{\mathcal{L}}}_{\mathcal{L}} B}$$

$$G\text{-I}\ \frac{F\Theta \xrightarrow{f}_{\mathcal{L}} A}{\Theta \xrightarrow{\eta}_{\mathcal{C}} GF\Theta \xrightarrow{Gf}_{\mathcal{C}} GA} \qquad G\text{-E}\ \frac{\Theta \xrightarrow{s}_{\mathcal{C}} GA}{F\Theta \xrightarrow{Fs}_{\mathcal{L}} FGA \xrightarrow{\epsilon}_{\mathcal{L}} A} \qquad F\text{-I}\ \frac{\Theta \xrightarrow{t}_{\mathcal{C}} X}{F\Theta \xrightarrow{Ft}_{\mathcal{L}} FX}$$

$$F\text{-E}\ \frac{F\Theta \otimes \Gamma \xrightarrow{e}_{\mathcal{L}} FX \qquad F(\Theta \times X) \otimes \Delta \xrightarrow{g}_{\mathcal{L}} B}{F\Theta \otimes \Gamma \otimes \Delta \xrightarrow{\mathrm{dupl}\otimes\mathrm{id}\otimes\mathrm{id}}_{\mathcal{L}} F\Theta \otimes F\Theta \otimes \Gamma \otimes \Delta \xrightarrow{\mathrm{id}\otimes e\otimes\mathrm{id}}_{\mathcal{L}} F\Theta \otimes FX \otimes \Delta \xrightarrow{m\otimes\mathrm{id}}_{\mathcal{L}} F(\Theta \times X) \otimes \Delta \xrightarrow{g}_{\mathcal{L}} B}$$

Auxiliary definitions

$$FX \xrightarrow{\mathrm{disc}}_{\mathcal{L}} I \quad = \quad FX \xrightarrow{F\langle\rangle}_{\mathcal{L}} F1 \xrightarrow{m^{-1}}_{\mathcal{L}} I$$

$$FX \xrightarrow{\mathrm{dupl}}_{\mathcal{L}} FX \otimes FX \quad = \quad FX \xrightarrow{F\langle\mathrm{id},\mathrm{id}\rangle}_{\mathcal{L}} F(X \times X) \xrightarrow{m^{-1}}_{\mathcal{L}} FX \otimes FX$$

Figure 13: Adjoint lambda calculus (λ_a), categorical interpretation

Translation of λ_m into λ_a.

$$
\begin{array}{rcl}
K^\flat & = & K \\
1^\flat & = & 1_{\mathcal{C}} \\
(X \times Y)^\flat & = & X^\flat \times Y^\flat \\
(X \to Y)^\flat & = & X^\flat \to Y^\flat \\
(TX)^\flat & = & GFX^\flat \\[1ex]
(\Theta \vdash u\colon Y)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} u^\flat\colon Y^\flat \\[1ex]
(\Theta, x\colon X \vdash x\colon X)^\flat & = & \Theta^\flat, x\colon X^\flat \vdash_{\mathcal{C}} x\colon X^\flat \\
(\Theta \vdash ()\colon 1)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} ()_{\mathcal{C}}\colon 1_{\mathcal{C}} \\
(\Theta \vdash (t, u)\colon X \times Y)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} (t^\flat, u^\flat)\colon X^\flat \times Y^\flat \\
(\Theta \vdash \mathrm{fst}\; s\colon X)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} \mathrm{fst}\; s^\flat\colon X^\flat \\
(\Theta \vdash \mathrm{snd}\; s\colon Y)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} \mathrm{snd}\; s^\flat\colon Y^\flat \\
(\Theta \vdash \lambda x\colon X.\, u\colon X \to Y)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} \lambda x\colon X^\flat.\, u^\flat\colon X^\flat \to Y^\flat \\
(\Theta \vdash s\, t\colon Y)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} s^\flat\, t^\flat : Y^\flat \\
(\Theta \vdash [t]\colon TX)^\flat & = & \Theta \vdash_{\mathcal{C}} GFt^\flat\colon GFX^\flat \\
(\Theta \vdash \mathrm{let}\; x \leftarrow s \;\mathrm{in}\; u\colon TY)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} G(\mathrm{let}\; Fx = G^{-1}s^\flat \;\mathrm{in}\; G^{-1}u^\flat)\colon GFY^\flat
\end{array}
$$

Figure 14: Translation of monadic into adjoint calculus

10 Translations into the adjoint calculus

Translations from the monad and linear calculi into the adjoint calculus are given in Figures 14 and 15. These translations preserve semantics.

Proposition 10.0.14 *A judgement $\Theta \vdash t : X$ of λ_m and its translation $\Theta^\flat \vdash t^\flat : X^\flat$ into λ_a correspond to identical arrows in $\mathcal{C}$.*

Proposition 10.0.15 *A judgement $\Gamma \vdash e : A$ of λ_l and its translation $\Gamma^\sharp \vdash e^\sharp : A^\sharp$ into λ_a correspond to identical arrows in $\mathcal{L}$.*

Proof. Straightforward induction. □

11 Conclusions and Further Work

We have shown how three different translations of the lambda calculus into the computational lambda calculus correspond exactly to three translations into intuitionistic linear logic.

One might be tempted to see this work as showing that ILL and the computational lambda calculus are really the same thing, but seen from different sides of the adjunction. The real situation is, however, rather more complicated. Although we have shown that the relationship is close, ILL only corresponds to models of the

Translation of λ_l into λ_a.

$$
\begin{array}{rcl}
L^\sharp & = & L \\
I^\sharp & = & I \\
(A \otimes B)^\sharp & = & A^\sharp \otimes B^\sharp \\
1^\sharp & = & 1_{\mathcal{L}} \\
(A \mathbin{\&} B)^\sharp & = & A^\sharp \mathbin{\&} B^\sharp \\
(A \multimap B)^\sharp & = & A^\sharp \multimap B^\sharp \\
(!A)^\sharp & = & FGA^\sharp \\
\\
(\Gamma \vdash e\colon A)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} e^\sharp\colon A^\sharp \\
\\
(a\colon A \vdash a\colon A)^\sharp & = & \text{-};\ a\colon A^\sharp \vdash_{\mathcal{L}} a\colon A^\sharp \\
(\ \vdash *\colon I)^\sharp & = & \text{-};\ \text{-} \vdash_{\mathcal{L}} *\colon I \\
(\Gamma \vdash \text{let } * = e \text{ in } f\colon A)^\sharp & = & \text{-};\ \Gamma \vdash_{\mathcal{L}} \text{let } * = e^\sharp \text{ in } f^\sharp\colon A^\sharp \\
(\Gamma \vdash e \otimes f\colon A \otimes B)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} e^\sharp \otimes f^\sharp\colon A^\sharp \otimes B^\sharp \\
(\Gamma \vdash \text{let } a \otimes b = e \text{ in } f\colon C)^\sharp & = & \text{-};\ \Gamma \vdash_{\mathcal{L}} \text{let } a \otimes b = e^\sharp \text{ in } f^l a\colon C^\sharp \\
(\Gamma \vdash ()\colon 1)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} ()_{\mathcal{L}}\colon 1_{\mathcal{L}} \\
(\Gamma \vdash (e, f)\colon A \mathbin{\&} B)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} (e\sharp, f\sharp)\colon A\sharp \mathbin{\&} B^\sharp \\
(\Gamma \vdash \text{fst } e\colon A)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} \text{fst } e^\sharp\colon A^\sharp \\
(\Gamma \vdash \text{snd } e\colon B)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} \text{snd } e^\sharp\colon B^\sharp \\
(\Gamma \vdash \lambda a\colon A.\, e\colon A \multimap B)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} \lambda a\colon A^\sharp.\, e^\sharp : A^\sharp \multimap B^\sharp \\
(\Gamma \vdash e\, f\colon B)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} e^\sharp\ f^\sharp : B^\sharp \\
(\Gamma \vdash \mathsf{promote}\ \vec{e_i}\ \mathsf{for}\ \vec{a_i}\ \mathsf{in}\ f\colon !B)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} \text{let } \vec{Fy_i} = \vec{e_i^\sharp} \text{ in } FGf^\sharp[\vec{Fy_i}/\vec{a_i}] : FGB^\sharp \\
(\Gamma \vdash \mathsf{derelict}\ e\colon A)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} \text{let } Fx = e^\sharp \text{ in } G^{-1}x\colon A^\sharp \\
(\Gamma \vdash \mathsf{discard}\ e\ \mathsf{in}\ f\colon B)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} \text{let } Fx = e^\sharp \text{ in } f^\sharp\colon B^\sharp \\
(\Gamma \vdash \mathsf{copy}\ e\ \mathsf{as}\ a, b\ \mathsf{in}\ f\colon B)^\sharp & = & \text{-};\ \Gamma^\sharp \vdash_{\mathcal{L}} \text{let } Fx = e^\sharp \text{ in } f^\sharp[Fx/a, Fx/b]\colon B^\sharp
\end{array}
$$

Figure 15: Translation of linear into adjoint calculus

computational lambda calculus in which the monad is commutative, and even then we have to add mild extra conditions to the monad models in order to obtain a model of ILL. A natural question is whether we can make the correspondence even closer. One way to do this would be to relax the extra conditions in Proposition 3.0.2, but we do not yet know of any natural way to do this. Another apparently plausible hope is that by removing the symmetry requirements from the adjoint model we might obtain a closer correspondence between the computational lambda calculus and a *non-commutative* variant of ILL. We have not investigated this possibility in any detail, but it does not seem very likely to work.

A more promising approach is to make use of recent work by Power and Robinson on [PR95]. They define a new notion of premonoidal category and use it to define a slight generalisation of Moggi's computational monads. This may well be an even better fit with ILL.

We are currently working on extending the results of the current paper to include recursion, making use of Crole and Pitts's notion of a fixpoint object (which is based on an inital T-algebra) [Crole92]. There is a natural linear version of this notion (based on an initial !-algebra) and in any adjoint model, linear fixpoint objects in L are closely related to fixpoint objects in C. We conjecture that this will allow us to extend the equivalence of the lifted and call-by-value translations to source languages including recursion. Note that this approach differs somewhat from that of [Brau94] in that we are using fixpoint objects, rather than fixpoint operations, and that for us, the construction of fixpoints in C is intimately related to the monad T, whereas Brauner assumes all fixpoints in the coKleisli category (which corresponds to our C). We do not wish to do this because it would rule out one of our most natural domain-theoretic models, namely that in which C is the category of predomains.

References

[Abr92] S. Abramsky, The lazy lambda calculus. In D. Turner, editor, *Research topics in functional programming*, Addison-Wesley, 1992.

[Abr93] S. Abramsky, Computational interpretations of linear logic. *Theoretical Computer Science* 111:3–57, 1993.

[Bar95] Andrew Barber, Dual Intuitionistic Linear Logic. Draft paper, University of Edinburgh, April 1995.

[BBdPH92] N. Benton, G. Bierman, V. de Paiva, and M. Hyland, Term assignment for intuitionistic linear logic. Technical report 262, Computer Laboratory, University of Cambridge, August 1992.

[BBdP95] P. N. Benton, G. M. Bierman and V. C. V. de Paiva, Computational types from a logical perspective I. Technical report 365, Computer Laboratory, University of Cambridge, May 1995.

[Ben92] P. N. Benton. Strictness analysis of lazy functional programs. PhD Thesis, Computer Laboratory, University of Cambridge, December 1992. Available as Technical report 309, August 1993.

[Ben94] P. N. Benton, A mixed linear and non-linear logic: proofs, terms, and models. In *Proceedings of Computer Science Logic 1994, Kazimierz, Poland*, Springer-Verlag LNCS 933, 1995. Full version available as Technical report 352, Computer Laboratory, University of Cambridge, October 1994.

[Bie94] G. Bierman, What is a categorical model of intuitionistic linear logic? Technical report 333, Computer Laboratory, University of Cambridge, April 1994. To appear in *Proceedings of the Conference on Typed Lambda Calculus and Applications*, Springer Verlag LNCS, April 1995.

[Brau94] T. Braüner. The Girard translation extended with recursion. In *Proceedings of Computer Science Logic 1994, Kazimierz, Poland*, Springer-Verlag LNCS 933, 1995.

[Crole92] R. L. Crole. Programming metalogics with a fixpoint type. PhD Thesis, Computer Laboratory, University of Cambridge, January 1992. Available as Technical report 247, February 1992.

[Gir87] J.-Y. Girard, Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir93] J.-Y. Girard, On the unity of logic. *Annals of Pure and Applied Logic*, 59:201–217, 1993.

[Hol88] S. Holmström, A linear functional language. Draft paper, Chalmers University of Technology, 1988.

[Kei78] W. F. Keigher. Symmetric monoidal closed categories generated by commutative adjoint monads. *Cahiers de Topolgie et Geometrie Differentielle* XIX-3 1978.

[Laf88] Y. Lafont, The linear abstract machine. *Theoretical Computer Science*, 59:157–180, 1988.

[LM92] P. Lincoln and J. Mitchell, Operational aspects of linear lambda calculus. In *7'th Symposium on Logic in Computer Science*, IEEE Press, Santa Cruz, California, June 1992.

[Mac94a] I. Mackie, Lilac: a functional programming language based on linear logic. *Journal of Functional Programming*, 4(4):395–434, October 1994.

[Mac94b] I. Mackie, The Geometry of Implementation. Doctoral Thesis, Imperial College London, 1994.

[Mog89] E. Moggi, Computational lambda-calculus and monads. In *4'th Symposium on Logic in Computer Science*, IEEE Press, Asilomar, California, June 1989.

[MOTW95] J. Maraist, M. Odersky, D. N. Turner, and P. Wadler, Call-by-name, call-by-value, call-by-need, and the linear lambda calculus. Technical report, Fakultät für Informatik, Universität Karlsruhe and Department of Computing Science, University of Glasgow, March 1995.

[Ong88] C.-H. L. Ong. The lazy lambda calculus: an investigation into the foundations of functional programming. PhD Thesis, Imperial College of Science and Technology, May 1988.

[Pitts90] A. M. Pitts. Notes on the call-by-value and call-by-name translation of the simply typed lambda calculus into the computational lambda calculus. Manuscript, October 1990.

[Plo75] G. D. Plotkin, Call-by-name, call-by-value, and the lambda calculus. *Theoretical Computer Science* 1:125–159, 1975.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[PR95] J. Power and E. Robinson. *Premonoidal categories and notions of computation.* Draft. 1995.

[RR94] S. R. della Rocca and L. Roversi, Lambda calculus and intuitionistic linear logic. Manuscript, July 1994. (Available from L. Roversi, University of Pisa, `rover@di.unipi.it`.)

[See89] R. A. G. Seely, Linear logic, $*$-autonomous categories, and cofree coalgebras. In *Categories in Computer Science and Logic*, June 1989. AMS Contemporary Mathematics 92.

[Tro92] A. S. Troelstra, *Lectures on Linear Logic*. CSLI Lecture Notes, 1992.

[Wad90] P. Wadler, Linear types can change the world! In M. Broy and C. Jones, editors, *Programming Concepts and Methods*, Sea of Galilee, Israel, North Holland, April 1990.

[Wad91] P. Wadler, Is there a use for linear logic? In *Partial Evaluation and Semantics-Based Program Manipulation (PEPM)*, New Haven, Connecticut, ACM Press, June 1991.

[Wad92] P. Wadler, Comprehending monads. *Mathematical Structures in Computer Science*, 2:461–493, 1992. (Special issue of selected papers from 6'th Conference on Lisp and Functional Programming.)

[Wad93a] P. Wadler, A syntax for linear logic. *Ninth International Conference on the Mathematical Foundations of Programming Semantics*, New Orleans, Louisiana, April 1993. Springer Verlag, LNCS 802.

[Wad93b] P. Wadler, A taste of linear logic. *Mathematical Foundations of Computer Science*, Gdansk, Poland, August 1993. Springer Verlag, LNCS 711.