# Linear Logic, Monads and the Lambda Calculus

Nick Benton*
University of Cambridge
Computer Laboratory
New Museums Site
Pembroke Street
Cambridge CB2 3QG, UK
Nick.Benton@cl.cam.ac.uk

Philip Wadler
University of Glasgow
Department of Computing Science
Lilybank Gardens
Glasgow G12 8QQ, UK
wadler@dcs.glasgow.ac.uk

## Abstract

*Models of intuitionistic linear logic also provide models of Moggi's computational metalanguage. We use the adjoint presentation of these models and the associated adjoint calculus to show that three translations, due mainly to Moggi, of the lambda calculus into the computational metalanguage (direct, call-by-name and call-by-value) correspond exactly to three translations, due mainly to Girard, of intuitionistic logic into intuitionistic linear logic. We also consider extending these results to languages with recursion.*

## 1. Introduction

Two of the most significant developments in semantics during the last decade are Girard's linear logic [10] and Moggi's computational metalanguage [14]. Any student of these formalisms will suspect that there are significant connections between the two, despite their apparent differences. The intuitionistic fragment of linear logic (ILL) may be modelled in a *linear model* – a symmetric monoidal closed category with a comonad ! which satisfies some extra conditions relating it to the monoidal structure [6]. Moggi's computational metalanguage may be modelled in a *monad model* – a cartesian closed category with a monad $T$ satisfying some different conditions relating it to the cartesian structure. The situations are tantalisingly close to dual: ! is almost, but not quite, entirely like $T$.

Benton has given an alternative presentation of models of ILL (*adjoint models*), 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 [4]. Taking $! = FG$ makes $\mathcal{L}$ a linear model, while taking $T = GF$ makes $\mathcal{C}$ a monad model. Further, every linear model arises in this way, as do all the members of a significant class of monad models.

Each of the three models corresponds to both a logic and an associated term calculus. The logic associated with monad models and the computational metalanguage is an intuitionistic modal logic, dubbed CL-logic in [5]. Associated with linear models and ILL are several proposals for linear term calculi, such as those of [6], [18], [11]. Here we choose to work with the calculus of [6]. Corresponding to adjoint models are the LNL term calculus (here referred to as the adjoint calculus) and LNL logic of [4].

Girard proposed two mappings of intuitionistic logic (or, equivalently, the simply typed lambda calculus) into ILL (or, equivalently, the linear term calculus), labelled $\circ$ and $*$ [10]. The connection between the $\circ$ translation and call-by-name (CBN) lambda calculus, and between the $*$ translation and call-by-value (CBV) lambda calculus has previously been noted in [12, 13]. Moggi also proposed two translations, clearly labelled as call-by-value and call-by-name [14], of the lambda calculus into his computational metalanguage; Wadler discusses these further in [17].

In this paper we start with three source calculi with identical syntax but different semantics: the *direct* calculus $\lambda$ is the usual simply typed lambda calculus, the *lifted* calculus $\lambda_\perp$ is a typed version of Abramsky and Ong's lazy lambda calculus [1], and the *call-by-value* calculus $\lambda_v$ is typed version of Plotkin's calculus of that name [16]. Each of these calculi is given two interpretations in an adjoint model via two translations – one into the monadic calculus, yielding a semantics in $\mathcal{C}$,

*Supported by EU BRA 8130 LOMAPS.

and one into the linear calculus, yielding a semantics in $\mathcal{L}$. For $\lambda$ we use a trivial mapping into the monadic calculus and Girard's $\circ$ translation into the linear calculus. For $\lambda_\perp$, we use Moggi's CBN translation and a slight variant of Girard's $\circ$ translation. For $\lambda_v$ we use Moggi's CBV translation and Girard's $*$ translation. The central result of the paper is that for each of the three calculi, the two interpretations are essentially the same, in that they are mates across the adjunction between $\mathcal{C}$ and $\mathcal{L}$. Thus, for instance, Girard's $*$ translation, first published in 1987, corresponds exactly to Moggi's call-by-value translation, first published in 1989, and we can finally see that this is the case in 1996.

## 2. Categorical models

In this section we recall the central semantic definitions from [4], [14] and [6] and some results which we shall need later.

**Definition 2.1** *An* adjoint model *is specified by*

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

We will use $A, B, C$ and $e, f, g$ range over objects and arrows in $\mathcal{L}$, and $X, Y, Z$ and $s, t, u$ to range over objects and arrows in $\mathcal{C}$.

Particularly important examples of adjoint models arise from the operation of lifting on categories of complete partial orders. Let $\mathcal{L}$ be the category of pointed $\omega$-cpos (domains) and strict continuous maps. This is an SMCC with tensor product given by smash product and internal hom by the strict continuous function space. We can take $\mathcal{C}$ to be either the category of $\omega$-cpos and continuous maps (predomains) or the category of pointed $\omega$-cpos and continuous maps. In both cases we obtain an adjoint model by taking $G$ to be the inclusion functor and $F$ to be the lifting functor $F\colon X \mapsto X_\perp$. The monoidal structure on $F$ is given by the isomorphism $X_\perp \otimes Y_\perp \cong (X \times Y)_\perp$.

Later, we will exploit the following isomorphisms:

**Lemma 2.2** *In any adjoint model,*

1. $F(X \times Y) \cong FX \otimes FY$ *and*
2. $X \to GB \;\cong\; G(FX \multimap B)$.

□

**Definition 2.3** *A* monad model *is specified by*

1. *a cartesian closed category* $(\mathcal{C}, 1, \times, \to)$, *and*
2. *a strong monad* $(T, \eta, \mu, \tau)$ *on* $\mathcal{C}$.

The relation between adjoint models and monad models is given by the following two propositions:

**Proposition 2.4** *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.* □

**Proposition 2.5** *If* $(\mathcal{C}, T, \eta, \mu)$ *is a monad model with a commutative monad and in addition* $\mathcal{C}$ *has equalisers and* $\mathcal{C}^T$*, the Eilenberg-Moore category, has coequalisers of reflexive pairs, then* $\mathcal{C}^T$ *is symmetric monoidal closed and* $(\mathcal{C}, \mathcal{C}^T, F, G, \eta, \epsilon)$ *is an adjoint model, where* $G$ *and* $F$ *are the forgetful and free functors, respectively.* □

**Definition 2.6** *A* linear model *is specified by*

1. *a symmetric monoidal closed category* $(\mathcal{L}, I, \otimes, \multimap)$,
2. *a symmetric monoidal comonad* $(!, \epsilon, \delta, q)$ *on* $\mathcal{L}$, *and*
3. *monoidal natural transformations with components* $\mathrm{disc}_A\colon !A \to I$ *and* $\mathrm{dupl}_A\colon !A \to !A \otimes !A$ *such that*
   (a) *each* $(!A, \mathrm{disc}_A, \mathrm{dupl}_A)$ *is a commutative comonoid,*
   (b) $\mathrm{disc}_A$ *and* $\mathrm{dupl}_A$ *are coalgebra maps, and*
   (c) *all coalgebra maps between free coalgebras preserve the comonoid structure.*

The relation between adjoint models and linear models is given by the following two propositions:

**Proposition 2.7** *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.* □

**Proposition 2.8** *Any linear model gives rise to an adjoint model in which* $\mathcal{C}$ *is a full subcategory of the Eilenberg-Moore category* $\mathcal{L}^!$ *of coalgebras; though it is not, in general, unique.* □

An adjoint or linear model may also have finite products $(1, \&)$ in $\mathcal{L}$. This is essential for the direct translation but not for the call-by-name and call-by-value translations. The correspondence between adjoint and linear models is maintained in the presence of such products.

## 3. Term calculi

The syntax and type rules of our three source calculi $\lambda$,$\lambda_\perp$ and $\lambda_v$ are just those of the usual simply typed lambda calculus with pairing and a unit type. Briefly, the difference between these calculi is that they have different equational theories: $\lambda$ satisfies the usual $\beta$ and $\eta$ rules, $\lambda_\perp$ just satisfies the $\beta$ rules and $\lambda_v$ satisfies the restricted $\beta_v$ and $\eta_v$ laws [16].

The computational metalanguage, $\lambda_m$, extends the simply typed calculus with a new unary type constructor $T$:

$$X, Y, Z \quad ::= \quad 1 \mid X \times Y \mid X \to Y \mid TX$$

and adds the following introduction and elimination forms to the syntax:

$$T\text{-I}\ \frac{\Theta \vdash t : X}{\Theta \vdash [t] : TX}$$

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

The types of the linear term calculus $\lambda_l$ are given by

$$A, B, C \quad ::= \quad I \mid A \otimes B \mid 1 \mid A \,\&\, B \mid A \multimap B \mid !A$$

and the syntax is shown in Figure 1. We adopt the conventions that distinct linear contexts mention disjoint sets of variable names and that any variable name occuring only in the RHS of a definition, or in the conclusion of an inference rule, is chosen fresh. Which constructs bind variables, and hence how capture-avoiding substitution is defined, should be obvious from the typing rules.

We are using the syntax of [6], which is now fairly standard and well-known, although this does make the terms associated with the translations rather large and unwieldy. The alternative would have been to use the DILL (for 'dual intuitionistic linear logic') syntax due to Wadler [18] and, independently, to Plotkin and Barber [3]. Using DILL makes two of the term translations appear more elegant, but makes the connection with the commonest presentations of ILL a little more indirect. It should be stressed that everything works equally well with either syntax. We omit the standard details of the interpretation of linear and monadic calculi in their respective models.

## 4. Translations

Before presenting the translations into the monadic and linear calculi, we note, without giving the details, that there are two natural *erasures* $|\text{-}|_m \colon \lambda_m \to \lambda$ and $|\text{-}|_l \colon \lambda_l \to \lambda$ which map the monadic and linear calculi into our source calculi. For each of the translations which we shall present, following the translation by the appropriate erasure is the identity on source calculus typing judgements, which can be used to show that each translation not only preserves, but also reflects, typing judgements.

From now on, we assume that we are working in an given adjoint model $(\mathcal{L}, \mathcal{C}, \ldots)$, with the linear calculus interpreted in $\mathcal{L}$ and the monadic calculus in $\mathcal{C}$.

### 4.1. Direct translations

The direct translation $\bullet \colon \lambda \to \lambda_m$ is just the trivial inclusion of the simply typed lambda calculus into the computational metalanguage which makes no use of the computation type constructor. The corresponding translation $\circ \colon \lambda \to \lambda_l$ into the linear calculus is usually known as 'the Girard translation', and is shown in Figure 2.

That the two translations $\circ$ and $\bullet$ give equivalent semantics is hardly surprising, as it was one of motivations for Girard's original formulation of linear logic and has subsequently been a touchstone in the formulation of various categorical models. However, the details are often glossed over (a notable exception being [8]). Firstly we observe that $GX^\circ \cong X^\bullet$ for each source type $X$. For the unit type this follows because $G$ preserves limits, whilst for compound types it follows from a little inductive calculation:

$$\begin{array}{ll} & G(X \times Y)^\circ \\ = & \quad \{ \text{ def'n } \circ \ \} \\ & G(X^\circ \,\&\, Y^\circ) \\ \cong & \quad \{ \ G + \text{limits } \} \\ & GX^\circ \times GY^\circ \\ \cong & \quad \{ \text{ induction } \} \\ & X^\bullet \times Y^\bullet \\ = & \quad \{ \text{ def'n } \bullet \ \} \\ & (X \times Y)^\bullet, \end{array} \qquad \begin{array}{ll} & G(X \to Y)^\circ \\ = & \quad \{ \text{ def'n } \circ \ \} \\ & G(FGX^\circ \multimap Y^\circ) \\ \cong & \quad \{ \text{ Lemma 2.2 } \} \\ & GX^\circ \to GY^\circ \\ \cong & \quad \{ \text{ induction } \} \\ & X^\bullet \to Y^\bullet \\ = & \quad \{ \text{ def'n } \bullet \ \} \\ & (X \to Y)^\bullet. \end{array}$$

Now write $\varphi$ for the top-to-bottom composite of the following chain of natural bijections between homsets:

$$\begin{array}{ll} & \hom_{\mathcal{C}}(X_1^\bullet \times \cdots \times X_n^\bullet,\, X^\bullet) \\ \cong & \quad \{ \text{ relation of } X^\bullet \text{ to } X^\circ \ \} \\ & \hom_{\mathcal{C}}(GX_1^\circ \times \cdots \times GX_n^\circ,\, GX^\circ) \\ \cong & \quad \{ \text{ adjunction } \} \\ & \hom_{\mathcal{C}}(F(GX_1^\circ \times \cdots \times GX_n^\circ),\, X^\circ) \\ \cong & \quad \{ \text{ Lemma 2.2 } \} \\ & \hom_{\mathcal{C}}(FGX_1^\circ \otimes \cdots \otimes FGX_n^\circ,\, X^\circ) \end{array}$$

The relation between the translations is the following:

**Proposition 4.1** *If* $\Theta \vdash t \colon X$ *in* $\lambda$ *then*

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

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

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

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

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

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

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

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

Figure 1. Linear lambda calculus ($\lambda_l$)

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

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

Figure 2. The direct translation $\circ$

**Proof.** Induction on $t$. One can use the categorical semantics to perform the calculations directly, but it is more convenient to use the adjoint calculus, which is the internal language of adjoint models. We describe the adjoint calculus in Section 5. □

### 4.2. Lifted translations

The translations associated with the lifted call-by-name calculus $\lambda_\perp$, shown in Figure 3 are $\dagger\colon \lambda_\perp \to \lambda_m$, which is Moggi's CBN translation, and $\ddagger\colon \lambda_\perp \to \lambda_l$, which is a slight variant of the Girard translation which appears to be new. The $\ddagger$ translation adds an extra ! to the translations of products and function spaces compared with the $\circ$ translation (note that the 'Seely isomorphisms' $I \cong !1$ and $(!A \otimes !B) \cong !(A \,\&\, B)$ hold in any linear category with products). By similar reasoning to that which we used for the direct translation, we can then calculate that $GX^\ddagger \cong GFX^\dagger$ for each $\lambda_\perp$ type $X$ and that there is a natural bijection

$$\phi : \hom_{\mathcal{C}}(GFX_1^\dagger \times \cdots \times GFX_n^\dagger,\, GFX^\dagger) \to \hom_{\mathcal{L}}(FGX_1^\ddagger \otimes \cdots \otimes FGX_n^\ddagger,\, X^\ddagger).$$

We can then use induction on $t$ to show

**Proposition 4.2** *If* $\Theta \vdash t\colon X$ *in* $\lambda_\perp$ *then*

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

□

### 4.3. Call-by-value translations

The call-by-value translations are $\star\colon \lambda_v \to \lambda_m$, which is Moggi's CBV translation [14], and $*\colon \lambda_v \to \lambda_l$ which is another translation due to Girard [10]. (Interestingly, although the CBV translation is central to Moggi's work, Girard thought his $*$ translation was "not of much interest".) These translations are shown in Figure 4.

The terms given in Figure 4 for the $*$ translation make use of a family of auxiliary macros $P_X(\cdot)$, indexed by the types of $\lambda_v$, which correspond to admissible rules of ILL. This complexity, which also arises if one uses the DILL syntax, is caused by the fact that the types of translated free variables are not all explicitly !-ed in this translation, so they cannot be directly weakened or contracted. For each $\lambda_v$ type $X$, however, we can define a macro $P_X(\cdot)$ such that $a\colon X^* \vdash P_X(a)\colon !X^*$, which does enable values of that type to be copied and discarded. We need that $\forall a\colon X^*.\ \text{derelict}\,(P_X(a)) = a$, though the other equation which would make $P_X(\cdot)$ a coalgebra structure map does not seem to be necessary.

$$\begin{aligned}
P_1(a) &= \text{let } * = a \text{ in promote } - \text{ for } - \text{ in } * \\
P_{X\times Y}(a) &= \text{let } b \otimes c = a \text{ in promote } P_X(b), P_Y(c) \\
&\quad \text{for } b', c' \text{ in derelict } b' \otimes \text{derelict } c' \\
P_{X\to Y}(a) &= \text{promote } a \text{ for } b \text{ in } b
\end{aligned}$$

A simple inductive calculation shows that for any type $X$ of $\lambda_v$, $X^* \cong FX^\star$ and thus that there is a natural bijection

$$\psi : \hom_{\mathcal{C}}(X_1^\star \times \cdots \times X_n^\star,\, GFX^\star) \to \hom_{\mathcal{L}}(X_1^* \otimes \cdots \otimes X_n^*,\, X^*).$$

such that the relation between the two translations is

**Proposition 4.3** *If* $\Theta \vdash t\colon X$ *in* $\lambda_v$ *then*

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

□

It is worth remarking that if we wished to extend the $*$ translation to incorporate other base types in the source language, then for each base type $B$, $B^*$ has to be a linear type on which we can define an appropriate $P_B(\cdot)$. The easiest way to ensure this is to take $B^*$ to be $!L_B$ for some $L_B$ so that we can use the free coalgebra structure. In general, however, there may other definitions of $P_B(\cdot)$, possibly involving extralogical constants, which work. Of course, if we also wished Proposition 4.3 to remain true, we should have to check that the linear and monadic translations of each base type and constant symbol were appropriately related, but this is necessary for any of our translation pairs.

## 5. The adjoint calculus

In this section we sketch the adjoint calculus $\lambda_a$ of [4] and describe the translations of $\lambda_l$ and $\lambda_m$ into $\lambda_a$ which were used to prove the main results presented here. The adjoint calculus includes both linear types and terms, corresponding to objects and arrows of $\mathcal{L}$, and non-linear types and terms, corresponding to objects and arrows of $\mathcal{C}$. The syntax of the adjoint calculus is shown in Figure 5.

Note that non-linear terms only have non-linear free variables, but that linear terms may have both linear and non-linear free variables. Corresponding to each judgement $x_1\colon X_1, \ldots, x_m\colon X_m \vdash_{\mathcal{C}} s\colon Y$ is an arrow $X_1 \times \cdots \times X_m \xrightarrow{s} Y$ in $\mathcal{C}$ and corresponding to each judgement $x_1\colon X_1, \ldots, x_m\colon X_m, a_1\colon A_1, \ldots, a_n\colon A_n \vdash_{\mathcal{L}} e\colon B$ is an arrow $F(X_1 \times \cdots \times X_m) \otimes A_1 \otimes \cdots \otimes A_n \xrightarrow{e} B$ in $\mathcal{L}$. Further details of both the semantics and the proof theory of $\lambda_a$ may be found in [4].

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

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

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

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

Figure 3. Lifted call-by-name translations

Both the linear and the monadic calculi may be translated into the adjoint calculus. These translations are shown in Figure 6, where we have omitted some entirely obvious clauses, such as that which states that $\otimes$-introduction in $\lambda_l$ is translated as $\otimes$-introduction in $\lambda_a$. Both of these translations preserve semantics in the sense that for any linear term $e$, the interpretation of $e^\sharp$ in the adjoint model $(\mathcal{L}, \mathcal{C}, \ldots)$ is equal to the one-step interpretation of $e$ in $\mathcal{L}$ *qua* linear model, and similarly for monad calculus terms.

The equational axioms for the adjoint calculus are shown in Figure 7. So, for example, the proof of Proposition 4.2 may be obtained by showing that if $\Theta \vdash t\colon X$ in $\lambda_\perp$ then $\phi((t^\dagger)^\flat) = (t^\ddagger)^\sharp$ using the rules of Figure 7 (plus the usual rules for equality, including congruence), where $\phi$ is an a macro corresponding to the bijection between homsets.

## 6. Coproducts

A linear category with binary coproducts models the extension of $\lambda_l$ which includes the additive sum $\oplus$. Similarly a monad model with binary coproducts models the extension of $\lambda_m$ with binary sums. Thus we are led to consider adjoint models with binary coproducts in both $\mathcal{L}$ and $\mathcal{C}$. The correspondence of Proposition 2.8 is not quite maintained when we add coproducts, since an adjoint model $(\mathcal{L}, \mathcal{C}, \ldots)$ derived from a linear category $(\mathcal{L}, \ldots)$ with coproducts has, in general, only *weak* coproducts in $\mathcal{C}$. This happens in the concrete domain theoretic case when $\mathcal{C}$ is the category of domains and continuous maps (the Kleisli category of the lift comonad on $\mathcal{L}$). Rather than work with weak coproducts, however, we choose to assume that we are working in a adjoint model with coproducts in both

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

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

Translation $* : \lambda_v \to \lambda_l$

$$
\begin{array}{rcl}
1^* & = & I \\
(X \times Y)^* & = & X^* \otimes Y^* \\
(X \to Y)^* & = & !(X^* \multimap Y^*) \\
\\
(\Theta \vdash t : X)^* & = & \Theta^* \vdash t^* : X^* \\
\\
(\vec{x_i}\colon \vec{X_i}, x\colon X \vdash x\colon X)^* & = & \vec{x_i}\colon \vec{X_i^*}, x\colon X^* \vdash \text{discard } P_{\vec{X_i}}(\vec{x_i}) \text{ in } x\colon X^* \\
(\vec{x_i}\colon \vec{X_i} \vdash ()\colon 1)^* & = & \vec{x_i}\colon \vec{X_i^*} \vdash \text{discard } P_{\vec{X_i}}(\vec{x_i}) \text{ in } *\colon I \\
(\vec{x_i}\colon \vec{X_i} \vdash (s,t)\colon X \times Y)^* & = & \vec{x_i}\colon \vec{X_i^*} \vdash \text{copy } P_{\vec{X_i}}(\vec{x_i}) \text{ as } \vec{a_i}, \vec{b_i} \text{ in} \\
 & & \quad s^*[\text{derelict } \vec{a_i}/\vec{x_i}] \otimes t^*[\text{derelict } \vec{b_i}/\vec{x_i}]\colon X^* \otimes Y^* \\
(\vec{x_i}\colon \vec{X_i} \vdash \text{fst } s\colon X)^* & = & \vec{x_i}\colon \vec{X_i^*} \vdash \text{let } a \otimes b = s^* \text{ in discard } P_Y(b) \text{ in } a\colon X^* \\
(\vec{x_i}\colon \vec{X_i} \vdash \text{snd } s\colon Y)^* & = & \vec{x_i}\colon \vec{X_i^*} \vdash \text{let } a \otimes b = s^* \text{ in discard } P_X(a) \text{ in } b\colon Y^* \\
(\vec{x_i}\colon \vec{X_i} \vdash (\lambda x\colon X.\, s)\colon X \to Y)^* & = & \vec{x_i}\colon \vec{X_i^*} \vdash \text{promote } P_{\vec{X_i}}(\vec{x_i}) \text{ for } \vec{a_i} \text{ in } \lambda x\colon X^*.\, s^*[\text{derelict } \vec{a_i}/\vec{x_i}]\colon !(X^* \multimap Y^*) \\
(\vec{x_i}\colon \vec{X_i} \vdash s\ t\colon Y)^* & = & \vec{x_i}\colon \vec{X_i^*} \vdash \text{copy } P_{\vec{X_i}}(\vec{x_i}) \text{ as } \vec{a_i}, \vec{b_i} \text{ in} \\
 & & \quad (\text{derelict } s^*[\text{derelict } \vec{a_i}/\vec{x_i}])\ (t^*[\text{derelict } \vec{b_i}/\vec{x_i}])\colon Y^*
\end{array}
$$

Figure 4. Call-by-value translations

categories, with the motivating concrete example being that in which $\mathcal{C}$ is the category of predomains and continuous maps. In such a model $F(X+Y) \cong FX \oplus FY$ because $F$ has a right adjoint. We add sums to our source calculi, and to $\lambda_l$ and $\lambda_m$, using the usual syntax for injections and case analysis.

To extend the lifted call-by-name translations, we take $(X+Y)^\dagger = TX^\dagger + TY^\dagger$ and $(X+Y)^\ddagger = !X^\ddagger \oplus !Y^\ddagger$. The isomorphism $GX^\ddagger \cong GFX^\dagger$ still holds with this definition, and the term translations are shown in Figure 8.

The call-by-value translations take $(X+Y)^\star = X^\star + Y^\star$ and $(X+Y)^* = X^* \oplus Y^*$. The isomorphism $X^* \cong FX^\star$ still holds and the terms are shown in Figure 8. In this case we also have to define $P_{X+Y}$:

$$
\begin{array}{l}
P_{X+Y}(a) = \text{case } a \text{ of} \\
\quad \text{inl } b \to \text{promote } P_X(b) \text{ for } b' \text{ in inl (derelict } b') \\
\quad \mid \text{inr } c \to \text{promote } P_Y(c) \text{ for } c' \text{ in inr (derelict } c')
\end{array}
$$

The extension of the direct translations is slightly more subtle. The Girard translation defines $(X+Y)^\circ = !X^\circ \oplus !Y^\circ$. Hence, to keep $GX^\circ \cong X^\bullet$, we have to take $(X+Y)^\bullet = T(X^\bullet + Y^\bullet)$, so that the $\bullet$ translation is no longer a trivial inclusion. (Cf. the fact that the naive unlifted domain-theoretic semantics of a PCF-like language interprets sum types by the separated sum of domains.) The extended $\bullet$ translation also uses a macro $D_X(\cdot)$ such that $x\colon TX^\bullet \vdash D_X(x)\colon X^\bullet$ and $\forall y\colon X^\bullet.\, D_X([y]) = y$. The other equation which would make $D_X(\cdot)$ an algebra structure map does not seem to be necessary.

| | | | |
|---|---|---|---|
| variables (intuitionistic) | $x, y, z$ | | |
| variables (linear) | $a, b, c$ | | |
| types (intuitionistic) | $X, Y, Z$ | $::=$ | $1_{\mathcal{C}} \mid A \times B \mid A \to B \mid GA$ |
| types (linear) | $A, B, C$ | $::=$ | $I \mid 1_{\mathcal{L}} \mid A \,\&\, B \mid A \otimes B \mid A \multimap B \mid FX$ |
| contexts (intuitionistic) | $\Theta$ | $::=$ | $x_1 : X_1, \ldots, x_n : X_n$ |
| contexts (linear) | $\Gamma, \Delta$ | $::=$ | $a_1 : A_1, \ldots, a_n : A_n$ |

$$\mathrm{Id}_{\mathcal{C}}\ \frac{}{\Theta, x : X \vdash_{\mathcal{C}} x : X} \qquad \mathrm{Id}_{\mathcal{L}}\ \frac{}{\Theta;\ a : A \vdash_{\mathcal{L}} a : A}$$

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

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

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

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

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

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

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

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

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

Figure 5. Adjoint lambda calculus ($\lambda_a$)

$$\begin{aligned}
D_1(x) &= () \\
D_{X\times Y}(x) &= (D_X(\mathrm{let}\ y \leftarrow x\ \mathrm{in}\ [\mathrm{fst}\ y]), \\
&\qquad D_Y(\mathrm{let}\ z \leftarrow x\ \mathrm{in}\ [\mathrm{snd}\ z])) \\
D_{X\to Y}(x) &= \lambda y{:}X^{\bullet}.\, D_Y(\mathrm{let}\ f \leftarrow x\ \mathrm{in}\ [f\ y]) \\
D_{X+Y}(x) &= \mathrm{let}\ z \leftarrow x\ \mathrm{in}\ z
\end{aligned}$$

The extended term translations are shown in Figure 8.

As our extension of the $\bullet$ translation to sums depends on $D_X(\cdot)$, if we were to add other base types $B$ to $\lambda$ then $B^{\bullet}$ should be a $\lambda_m$ type equipped with an appropriate $D_B(\cdot)$. However (cf. our earlier remark about the $*$ translation), there seems no reason to demand that this be the free algebra structure on a type of the form $TM_B$.

**Proposition 6.1** *Each of Propositions 4.1, 4.2 and 4.3 remains true when the languages and translations are extended with coproducts as described above.* □

## 7. Recursion

Our results also extend to languages with recursion, though there is some choice in exactly how recursion is treated. Brauner has already presented an extension of Proposition 4.1 in a linear model with parameterised fixpoints of all endomaps in the Kleisli category [8]. Bearing the case where $\mathcal{C}$ is predomains in mind, we instead use stronger version, due to Freyd, of Crole and Pitts's notion of *fixpoint object* [9] so that we only require certain fixpoints in $\mathcal{C}$.

Translation $\flat : \lambda_m \to \lambda_a$ (extract).

$$\begin{array}{rcl}
1^\flat & = & 1_{\mathcal{C}} \\
(X \times Y)^\flat & = & X^\flat \times Y^\flat \\
(X \to Y)^\flat & = & X^\flat \to Y^\flat \\
(TX)^\flat & = & GFX^\flat \\
\\
(\Theta \vdash u\colon Y)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} u^\flat\colon Y^\flat \\
\\
(\Theta \vdash [t]\colon TX)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} GFt^\flat\colon GFX^\flat \\
(\Theta \vdash \text{let } x \leftarrow s \text{ in } u\colon TY)^\flat & = & \Theta^\flat \vdash_{\mathcal{C}} G(\text{let } Fx = G^{-1}s^\flat \text{ in } G^{-1}u^\flat)\colon GFY^\flat
\end{array}$$

Translation $\sharp : \lambda_l \to \lambda_a$ (extract).

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

Figure 6. Translation of monadic and linear calculi into adjoint calculus

**Definition 7.1** *A monad model has a* Freyd fixpoint object *(or* ffpo*) if it has an initial $T$-algebra $\sigma\colon T\Omega \to \Omega$ such that $\sigma^{-1}\colon \Omega \to T\Omega$ is a terminal coalgebra. (Here we intend algebra and coalgebra to be understood in the sense of algebra for a functor rather than algebra for a monad.)*

*Similarly, we say a linear model has a* linear ffpo *if there is an initial $!$-algebra $\bar{\sigma}\colon !\bar{\Omega} \to \bar{\Omega}$ such that $\bar{\sigma}^{-1}\colon \bar{\Omega} \to !\bar{\Omega}$ is a terminal coalgebra.*

In a monad model with a ffpo, we can model an extension of $\lambda_m$ which adds the syntax

$$\frac{\Theta, x\colon TX \vdash t\colon X}{\Theta \vdash \text{mfix}(x.t)\colon X} \quad \text{by} \quad \frac{\Theta \times TX \xrightarrow{t} X}{\Theta \xrightarrow{\langle 1, \langle\rangle ; \omega\rangle ; \tilde{t}} X}$$

where $\omega$ is the unique map such that $\omega ; \sigma^{-1} = \eta_1 ; T\omega$ and $\tilde{t}$ is the unique map such that

$$\begin{array}{ccc}
\Theta \times T\Omega & \xrightarrow{\langle \pi_1, \tau ; T\tilde{t}\rangle} & \Theta \times TX \\
{\scriptstyle 1 \times \sigma}\downarrow & & \downarrow{\scriptstyle t} \\
\Theta \times \Omega & \xrightarrow[\tilde{t}]{} & X
\end{array}$$

commutes. Note that this requires that $\sigma\colon T\Omega \to \Omega$ be initial in a parameterised sense, but this is a consequence of the closed structure in $\mathcal{C}$.

In a linear model with a linear ffpo, we can model an extension of $\lambda_l$ which adds the syntax

$$\frac{\begin{array}{c}\Delta_1 \vdash e_1\colon !A_1 \\ \vdots \\ \Delta_n \vdash e_n\colon !A_n \qquad \vec{a_i}\colon !\vec{A_i}, b\colon !B \vdash f\colon B\end{array}}{\Delta_1, \ldots, \Delta_n \vdash \text{box } \vec{e_i} \text{ for } \vec{a_i} \text{ in } \text{lfix}(b.f)\colon B}$$

by

$$\frac{\Delta_i \xrightarrow{e_i} !A_i \qquad (\bigotimes_i !A_i) \otimes !B \xrightarrow{f} B}{\bigotimes_i \Delta_i \xrightarrow{\otimes_i e_i} \bigotimes_i !A_i \xrightarrow{r ; 1 \otimes \bar{\omega}} (\bigotimes_i !A_i) \otimes \bar{\Omega} \xrightarrow{\hat{f}} B}$$

$\beta$ rules:

$$\frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon A \qquad \Theta;\ \Gamma \vdash_{\mathcal{L}} f\colon B}{\Theta;\ \Gamma \vdash_{\mathcal{L}} \mathrm{fst}\ (e,f) = e : A} \qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon X \qquad \Theta \vdash_{\mathcal{C}} t\colon Y}{\Theta \vdash_{\mathcal{C}} \mathrm{fst}\ (s,t) = s : X}$$

$$\frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon A \qquad \Theta;\ \Gamma \vdash_{\mathcal{L}} f\colon B}{\Theta;\ \Gamma \vdash_{\mathcal{L}} \mathrm{snd}\ (e,f) = f : B} \qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon X \qquad \Theta \vdash_{\mathcal{C}} t\colon Y}{\Theta \vdash_{\mathcal{C}} \mathrm{snd}\ (s,t) = t : Y}$$

$$\frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon A \qquad \Theta;\ \Gamma' \vdash_{\mathcal{L}} f\colon B \qquad \Theta;\ \Delta, a\colon A, b\colon B \vdash_{\mathcal{L}} g\colon C}{\Theta;\ \Gamma,\Gamma',\Delta \vdash_{\mathcal{L}} (\mathrm{let}\ a \otimes b = e \otimes f\ \mathrm{in}\ g) = g[e/a, f/b] : C} \qquad \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon A}{\Theta;\ \Gamma \vdash_{\mathcal{L}} (\mathrm{let}\ * = *\ \mathrm{in}\ e) = e : A}$$

$$\frac{\Theta;\ \Gamma, a\colon A \vdash_{\mathcal{L}} e\colon B \qquad \Theta;\ \Delta \vdash_{\mathcal{L}} f\colon A}{\Theta;\ \Gamma,\Delta \vdash_{\mathcal{L}} (\lambda a\colon A.\, e)\ f = e[f/a] : B} \qquad \frac{\Theta, x\colon X \vdash_{\mathcal{C}} s\colon Y \qquad \Theta \vdash_{\mathcal{C}} t\colon X}{\Theta \vdash_{\mathcal{C}} (\lambda x\colon X.\, s)\ t = s[t/x] : Y}$$

$$\frac{\Theta \vdash_{\mathcal{C}} s\colon X \qquad \Theta, x\colon X;\ \Gamma \vdash_{\mathcal{L}} e\colon A}{\Theta;\ \Gamma \vdash_{\mathcal{L}} (\mathrm{let}\ F(x) = F(s)\ \mathrm{in}\ e) = e[s/x] : A} \qquad \frac{\Theta;\ \text{-} \vdash_{\mathcal{L}} e\colon A}{\Theta;\ \text{-} \vdash_{\mathcal{L}} G^{-1}(G(e)) = e : A}$$

$\eta$ rules:

$$\frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon 1_{\mathcal{L}}}{\Theta;\ \Gamma \vdash_{\mathcal{L}} e = ()_{\mathcal{L}} : 1_{\mathcal{L}}} \qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon 1_{\mathcal{C}}}{\Theta \vdash_{\mathcal{C}} s = ()_{\mathcal{C}} : 1_{\mathcal{C}}}$$

$$\frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon A \,\&\, B}{\Theta;\ \Gamma \vdash_{\mathcal{L}} (\mathrm{fst}\ e, \mathrm{snd}\ e) = e : A \,\&\, B} \qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon X \times Y}{\Theta \vdash_{\mathcal{C}} (\mathrm{fst}\ s, \mathrm{snd}\ s) = s : X \times Y}$$

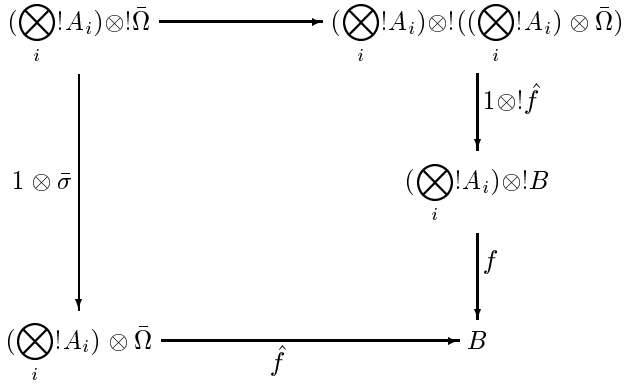
$$\frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon FX \qquad \Theta;\ \Delta, a\colon FX \vdash_{\mathcal{L}} f\colon B}{\Theta;\ \Gamma,\Delta \vdash_{\mathcal{L}} (\mathrm{let}\ F(x) = e\ \mathrm{in}\ f[F(x)/a]) = f[e/a] : B} \qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon X \to Y}{\Theta \vdash_{\mathcal{C}} (\lambda x\colon X.\, s\ x) = s : X \to Y}$$

$$\frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon I \qquad \Theta;\ \Delta, a\colon I \vdash_{\mathcal{L}} f\colon B}{\Theta;\ \Gamma,\Delta \vdash_{\mathcal{L}} (\mathrm{let}\ * = e\ \mathrm{in}\ f[*/a]) = f[e/a] : B} \qquad \frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon A \multimap B}{\Theta;\ \Gamma \vdash_{\mathcal{L}} (\lambda a\colon A.\, e\ a) = e : A \multimap B}$$

$$\frac{\Theta;\ \Gamma \vdash_{\mathcal{L}} e\colon A \otimes B \qquad \Theta;\ \Delta, c\colon A \otimes B \vdash_{\mathcal{L}} f\colon C}{\Theta;\ \Gamma,\Delta \vdash_{\mathcal{L}} (\mathrm{let}\ a \otimes b = e\ \mathrm{in}\ f[a \otimes b/c]) = f[e/c] : C} \qquad \frac{\Theta \vdash_{\mathcal{C}} s\colon X}{\Theta \vdash_{\mathcal{C}} G(G^{-1}(s)) = s : X}$$

Figure 7. Equational axioms for $\lambda_a$

where $\bar{\omega}$ is the unique map such that $\bar{\omega}; \bar{\sigma}^{-1} = q; !\bar{\omega}$ and $\hat{f}$ is the unique map such that




commutes (where we have omitted the details of some context manipulation). Again, the existence of $\hat{f}$ depends on suitably parameterised initiality, but this is a consequence of ordinary initiality in the presence of the closed structure in $\mathcal{L}$.

**Lemma 7.2** *In any adjoint model $(\mathcal{L}, \mathcal{C}, \ldots)$, $\mathcal{C}$ has an ffpo if and only if $\mathcal{L}$ has a linear ffpo.* □

So we now assume that we are working in an adjoint model with ffpos.

If we add recursion to $\lambda_\perp$ by

$$\frac{\Theta, x\colon X \vdash t\colon X}{\Theta \vdash \mathrm{rec}(x.t)\colon X}$$

Direct translations

$$
\begin{array}{rcl}
(\Theta \vdash \mathrm{inl}\, t\colon X+Y)^\bullet & = & \Theta^\bullet \vdash [\mathrm{inl}\, t^\bullet]\colon T(X^\bullet + Y^\bullet) \\
(\Theta \vdash \mathrm{case}\ s\ \mathrm{of}\ \mathrm{inl}\, x \to t \mid \mathrm{inr}\, y \to u\colon Z)^\bullet & = & \Theta^\bullet \vdash D_Z(\mathrm{let}\ z \leftarrow s^\bullet\ \mathrm{in}\ [\mathrm{case}\ z\ \mathrm{of}\ \mathrm{inl}\, x \to t^\bullet \mid \mathrm{inr}\, y \to u^\bullet])\colon Z^\bullet \\
(\vec{x_i}\colon \vec{X_i} \vdash \mathrm{inl}\, t\colon X+Y)^\circ & = & \mathrm{inl}\,(\mathrm{promote}\ \vec{x_i}\ \mathrm{for}\ \vec{a_i}\ \mathrm{in}\ t^\circ[\vec{a_i}/\vec{x_i}])\colon\, !X^\circ \oplus !Y^\circ \\
(\vec{x_i}\colon \vec{X_i} \vdash \mathrm{case}\ s\ \mathrm{of}\ \mathrm{inl}\, x \to t \mid \mathrm{inr}\, y \to u\colon Z)^\circ & = & \vec{x_i}\colon !\vec{X_i}^{\,\circ} \vdash \mathrm{copy}\ \vec{x_i}\ \mathrm{as}\ \vec{a_i}, \vec{b_i}\ \mathrm{in\ case}\ s^\circ[\vec{a_i}/\vec{x_i}]\ \mathrm{of} \\
 & & \quad \mathrm{inl}\, x \to t^\circ[\vec{b_i}/\vec{x_i}] \mid \mathrm{inr}\, y \to u^\circ[\vec{b_i}/\vec{x_i}]\colon Z^\circ
\end{array}
$$

Lifted translations

$$
\begin{array}{rcl}
(\Theta \vdash \mathrm{inl}\, t\colon X+Y)^\dagger & = & T\Theta^\dagger \vdash [\mathrm{inl}\, t^\dagger]\colon T(TX^\dagger + TY^\dagger) \\
(\Theta \vdash \mathrm{case}\ s\ \mathrm{of}\ \mathrm{inl}\, x \to t \mid \mathrm{inr}\, y \to u\colon Z)^\dagger & = & T\Theta^\dagger \vdash \mathrm{let}\ w \leftarrow s^\dagger\ \mathrm{in\ case}\ w\ \mathrm{of}\ \mathrm{inl}\, x \to t^\dagger \mid \mathrm{inr}\, y \to u^\dagger\colon TZ^\dagger \\
(\vec{x_i}\colon \vec{X_i} \vdash \mathrm{inl}\, t\colon X+Y)^\ddagger & = & \vec{x_i}\colon !\vec{X_i}^{\,\ddagger} \vdash \mathrm{inl}\,(\mathrm{promote}\ \vec{x_i}\ \mathrm{for}\ \vec{a_i}\ \mathrm{in}\ t^\ddagger[\vec{a_i}/\vec{x_i}])\colon\, !X^\ddagger \oplus !Y^\ddagger \\
(\vec{x_i}\colon \vec{X_i} \vdash \mathrm{case}\ s\ \mathrm{of}\ \mathrm{inl}\, x \to t \mid \mathrm{inr}\, y \to u\colon Z)^\ddagger & = & \vec{x_i}\colon !\vec{X_i}^{\,\ddagger} \vdash \mathrm{copy}\ \vec{x_i}\ \mathrm{as}\ \vec{a_i}, \vec{b_i}\ \mathrm{in\ case}\ s^\ddagger[\vec{a_i}/\vec{x_i}]\ \mathrm{of} \\
 & & \quad \mathrm{inl}\, x \to t^\ddagger[\vec{b_i}/\vec{x_i}] \mid \mathrm{inr}\, y \to u^\ddagger[\vec{b_i}/\vec{x_i}]\colon Z^\ddagger
\end{array}
$$

Call-by-value translations

$$
\begin{array}{rcl}
(\Theta \vdash \mathrm{inl}\, t\colon X+Y)^\star & = & \Theta^\star \vdash \mathrm{let}\ x \leftarrow t^\star\ \mathrm{in}\ [\mathrm{inl}\, x]\colon T(X^\star + Y^\star) \\
(\Theta \vdash \mathrm{case}\ s\ \mathrm{of}\ \mathrm{inl}\, x \to t \mid \mathrm{inr}\, y \to u\colon Z)^\star & = & \Theta^\star \vdash \mathrm{let}\ w \leftarrow s^\star\ \mathrm{in\ case}\ w\ \mathrm{of}\ \mathrm{inl}\, x \to t^\star \mid \mathrm{inr}\, y \to u^\star\colon TZ^\star \\
(\Theta \vdash \mathrm{inl}\, t\colon X+Y)^* & = & \Theta^* \vdash \mathrm{inl}\, t^*\colon X^* \oplus Y^* \\
(\vec{x_i}\colon \vec{X_i} \vdash \mathrm{case}\ s\ \mathrm{of}\ \mathrm{inl}\, x \to t \mid \mathrm{inr}\, y \to u\colon Z)^* & = & \vec{x_i}\colon \vec{X_i}^{\,*} \vdash \mathrm{copy}\ P_{\vec{X_i}}(\vec{x_i})\ \mathrm{as}\ \vec{a_i}, \vec{b_i}\ \mathrm{in\ case}\ s^*[\mathrm{derelict}\ \vec{a_i}/\vec{x_i}]\ \mathrm{of} \\
 & & \quad \mathrm{inl}\, x \to t^*[\mathrm{derelict}\ \vec{b_i}/\vec{x_i}] \mid \mathrm{inr}\, y \to u^*[\mathrm{derelict}\ \vec{b_i}/\vec{x_i}]\colon Z^*
\end{array}
$$

Figure 8. Translations of coproducts

then we can extend the lifted translations by

- $(\Theta \vdash \mathrm{rec}(x.t)\colon X)^\dagger \quad = \quad T\Theta^\dagger \vdash \mathrm{mfix}(y.\mathrm{let}\ x \leftarrow y\ \mathrm{in}\ t^\dagger)\colon TX^\dagger$
- $(\vec{x_i}\colon \Theta, \vdash \mathrm{rec}(x.t)\colon X)^\ddagger \quad = \quad \vec{x_i}\colon !\Theta^\ddagger \vdash \mathrm{box}\ \vec{x_i}\ \mathrm{for}\ \vec{a_i}\ \mathrm{in}\ \mathrm{lfix}(x.t^\ddagger[\vec{a_i}/\vec{x_i}])\colon X^\ddagger$

In the call-by-value calculus $\lambda_v$, we add recursion just at function types:

$$\frac{\Theta, f\colon X \to Y, x\colon X \vdash t\colon Y}{\Theta \vdash (\mathrm{rec}\ f\ x = t)\colon X \to Y}$$

and we can then extend the call-by-value translations as follows:

$$
\begin{array}{l}
\bullet (\Theta \vdash (\mathrm{rec}\ f\ x = t)\colon X \to Y)^\star \quad = \\
\Theta^\star \vdash [\mathrm{mfix}(z.\lambda x\colon X^\star.\ \mathrm{let}\ f \leftarrow z\ \mathrm{in}\ t^\star)]\colon T(X^\star \to TY^\star) \\
\bullet (\vec{x_i}\colon \vec{X_i} \vdash (\mathrm{rec}\ f\ x = t)\colon X \to Y)^* \quad = \\
\vec{x_i}\colon \vec{X_i}^{\,*} \vdash \mathrm{promote}\ P_{\vec{X_i}}(\vec{x_i})\ \mathrm{for}\ \vec{a_i}\ \mathrm{in\ box}\ \vec{a_i}\ \mathrm{for}\ \vec{b_i} \\
\quad \mathrm{in}\ \mathrm{lfix}(f.\lambda x\colon X^*.\ t^*[\mathrm{derelict}\ \vec{b_i}/\vec{x_i}])\colon\, !(X^* \multimap Y^*)
\end{array}
$$

The double variable rebinding caused by the fact that both promote and box... lfix build in substitution is unfortunate but could easily be avoided by incorporating promotion into the lfix rule.

Extending the correspondence between the direct translations to a language with recursion also involves using the $D_X(\cdot)$ macro which we introduced when considering coproducts. We add recursion to $\lambda$ by

$$\frac{\Theta, x\colon X \vdash t\colon X}{\Theta \vdash \mathrm{rec}(x.t)\colon X}$$

and then extend the direct translations by

- $(\Theta \vdash \mathrm{rec}(x.t)\colon X)^\bullet \quad = \quad \Theta^\bullet \vdash \mathrm{mfix}(y.t^\bullet[D_X(y)/x])\colon X^\bullet$
- $(\vec{x_i}\colon \vec{X_i} \vdash \mathrm{rec}(x.t)\colon X)^\circ \quad = \quad \vec{x_i}\colon !\vec{X_i}^{\,\circ} \vdash \mathrm{box}\ \vec{x_i}\ \mathrm{for}\ \vec{a_i}\ \mathrm{in}\ \mathrm{lfix}(x.t^\circ[\vec{a_i}/\vec{x_i}])\colon X^\circ$

**Proposition 7.3** *Proposition 6.1 remains true when each of the languages and translations is extended with recursion as described above.* □

The proof of Proposition 7.3 may be obtained by extending the adjoint calculus with terms and equations corresponding to the fixpoint objects in the model. By Lemma 7.2, it suffices to add such syntax to the nonlinear part of the calculus, as the linear versions then arise as derived rules. We omit the rather messy details, however.

## 8. Conclusions and Further Work

Are the linear and monadic calculi two sides of the same coin? More precisely, are they equivalent calculi, but viewed from different sides of the adjunction? The straightforward answer is certainly no, not least because adjoint models give rise to all linear models, but only to commutative monad models. (We do not yet know if it is possible to define a non-commutative linear calculus which corresponds to a wider class of monad models.) But to the extent that the two calculi can be used to give equivalent interpretations of each of our three source calculi, the answer seems to be yes.

Perhaps results formally similar to those presented here could be obtained for other intuitionistic modal logics, such as the inuitionistic S4 of [7]. It may be that the relationship between the translations is a consequence more of the $\Box$-like character of ! than of the restrictions on weakening and contraction in $\lambda_l$.

The significance of these results extends well beyond the treatment of partiality by lifting and many suggestive similarities remain. For example, both linear and monadic types have been used to add updateable state to functional languages in a controlled way [2, 15]. We hope that our work may serve as a starting point for a formal comparison of these approaches.

## 9. Acknowledgements

We should like to thank Gavin Bierman, Martin Hyland, Eugenio Moggi, Valeria de Paiva, Andy Pitts, Gordon Plotkin, Robert Seely and the attendees of the SCILL workshop (Syntactic Control of Interference and Linear Logic), held at Glasgow in August 1995, for fruitful discussions, advice and inspiration.

## References

[1] S. Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Directions in Functional Programming*, chapter 4, pages 65–116. Addison-Wesley, 1990.

[2] P. Achten and R. Plasmeijer. The ins and outs of Clean I/O. *Journal of Functional Programming*, 5(1):81–110, Jan. 1992.

[3] A. Barber. Dual intuitionistic linear logic. Draft paper, University of Edinburgh, Apr. 1995.

[4] P. N. Benton. A mixed linear and non-linear logic: Proofs, terms and models. In *Proceedings of Computer Science Logic 1994, Kazimierz, Poland*, volume 933 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995. Full version available as Technical Report 352, Computer Laboratory, University of Cambridge, October 1994.

[5] 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.

[6] P. N. Benton, G. M. Bierman, J. M. E. Hyland, and V. C. V. de Paiva. Term assignment for intuitionistic linear logic. Technical Report 262, Computer Laboratory, University of Cambridge, Aug. 1992.

[7] G. M. Bierman and V. C. V. de Paiva. Intuitionistic necessity revisited. In *Proceedings of Logic at Work Conference, Amsterdam*, Dec. 1992.

[8] T. Brauner. The Girard translation extended with recursion. In *Proceedings of Computer Science Logic 1994, Kazimierz, Poland*, volume 933 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[9] R. L. Crole and A. M. Pitts. New foundations for fixpoint computations: FIX-hyperdoctines and the FIX-logic. *Information and Computation*, 98(2), June 1992.

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

[11] P. Lincoln and J. Mitchell. Operational aspects of linear lambda calculus. In *Proceedings of the Seventh Symposium on Logic in Computer Science, Santa Cruz*. IEEE Press, June 1992.

[12] I. Mackie. *The Geometry of Implementation*. PhD thesis, Imperial College, London, 1994.

[13] 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, Mar. 1995.

[14] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[15] S. Peyton Jones and J. Launchbury. State in Haskell. *Lisp and Symbolic Computation*, 8(4):293–341, Dec. 1995.

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

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

[18] P. Wadler. A syntax for linear logic. In *Proceedings of the Ninth International Conference on the Mathematical Foundations of Programming Semantics, New Orleans, Louisiana*, volume 802 of *Lecture Notes in Computer Science*. Springer Verlag, Apr. 1993.