Direct models for the computational lambda-calculus

Carsten Führmann
car@dcs.ed.ac.uk

Technical Report ECS–LFCS–98-400
Department of Computer Science
University of Edinburgh

December 2, 1998

Abstract: We give direct categorical models for the computational lambda-calculus. By 'direct' I mean that the model consists of one category together with operators on objects and morphisms for modelling type and program constructors, respectively. Moggi's λ_C-models, for example, are not direct, because the category of program denotations is constructed as the Kleisli category of a monad. We call our models 'direct λ_C-models'. The main result is, loosely speaking, that each λ_C-model generates a direct λ_C-model, and each direct λ_C-model arises in this way. We shall make this precise by showing that the category of direct λ_C-models is reflective in the category of λ_C-models. From this we shall deduce that we can replace λ_C-models by direct λ_C-models without losing or gaining generality. We shall also see that the category of direct λ_C-models is equivalent to the category of λ_C-models that fulfil the equalizing requirement. Moreover, we shall see that we can describe our direct λ_C-models with universally quantified equations, which helps reasoning about programs and models. Finally, we shall see that direct λ_C-models reveal two kinds of well-behaved programs which are not obvious from λ_C-models.

1 Introduction

Cartesian-closed categories validate the β-law, which is false for realistic call-by-value programming languages. (For example, if $\perp$ is a looping program, then $(\lambda x.\lambda y.y) \perp 1 \neq 1$ in a call-by-value language.) By contrast, the theory of Moggi's λ_C-models [Mog88], which is called the computational lambda-calculus, has only equations which are operationally true for a range of realistic call-by-value programming languages. A λ_C-model is a category C with finite products together with a strong monad T and T-exponentials—that is, for all objects A and B, an exponential of TB by A. Environments Γ and types A of the computational lambda-calculus denote objects $[\![\Gamma]\!]$ and $[\![A]\!]$, respectively, in the Kleisli category C_T. A sequent $\Gamma \vdash M : A$ denotes a morphism $[\![\Gamma]\!] \longrightarrow [\![A]\!]$ in C_T, which in C is a morphism $[\![\Gamma]\!] \longrightarrow T[\![A]\!]$. There is a simple example with $C = \mathit{Set}$ and a certain monad T such that, for each set A, we have $TA = \{\bot\} \cup \{\llcorner a \lrcorner : a \in A\}$. Then C_T is isomorphic to the category of sets and partial functions, where $\bot$ serves as 'undefined'. In general, the monad T can be seen as a parameter that depends on the computational effect—there are lifting monads (partiality), state monads, continuations monads, and so on. (You can find the term formation rules and Moggi's semantics of the computational lambda-calculus in appendix A.)

Cartesian-closed categories are *direct* models, by which I mean that the objects and morphisms of the cartesian-closed category give the denotations for types and programs, respectively. λ_C-models are not direct, because the category of program denotations is constructed as the Kleisli category of a monad.

Another approach to semantics of call-by-value languages are *Freyd categories* (see [PT98, PT97]). A Freyd category consists of a category C with finite products, a symmetric premonoidal category K, and an identity-on-objects strict symmetric premonoidal functor $F : C \longrightarrow K$. (I shall explain symmetric premonoidal categories in this article.) The type and program denotations are objects and morphisms, respectively, of K. Freyd categories are not direct because of the auxiliary category C. As we shall see, λ_C-models are equivalent to *closed Freyd categories* [PT98]. A closed Freyd category is a Freyd category $F \dashv G : K \longrightarrow C$ together with *Kleisli exponentials*—that is, for each object A an adjunction $F(-) \otimes A \dashv A \Rightarrow (-) : K \longrightarrow C$. By 'equivalent' I mean an equivalence of categories between an obvious category of λ_C-models and an obvious category of closed Freyd categories. (Some vital parts of this equivalence occur in the work of John Power and Edmund Robinson, for example [PR97].)

In this article we define direct models for the computational lambda-calculus. We call these *direct λ_C-models*. I found them by analysing Hayo Thielecke's $\otimes\neg$-categories [Thi97a, Thi97b], which are direct models for call-by-value languages with higher-order types and continuations. (Roughly speaking, continuations bring the power of jumps into functional programming.) As we shall see, $\otimes\neg$-categories are direct λ_C-models with extra structure.

Loosely speaking, the main result in this article is that each λ_C-model generates a direct λ_C-model, and each direct λ_C-model arises in this way. We shall make this precise by showing that the category of direct λ_C-models is reflective in the category of λ_C-models. From this we shall deduce that we can replace λ_C-models by direct λ_C-models without losing or gaining generality.

The direct λ_C-models are *algebraic*, by which I mean that we can describe them with universally quantified equations. (If this is not clear enough now, it will become clearer later in this article.) This has some benefits, two of which are

- We can do all reasoning by replacing subexpressions along the axioms.
- We have a simple meta-theory—for example, we can form the free model generated by a set of operators and equations, adjoin indeterminates, and so on.

As we shall see in section 5, direct λ_C-models reveal two kinds of well-behaved programs which are not obvious from λ_C-models: thunkable programs and central programs.

We define the direct λ_C-models by giving structure in three steps:

direct λ_C-models

thunk-*force*-$\otimes$-categories

thunk-*force*-categories

(I took the names *thunk* and *force*, which stand for certain natural transformations, from $\otimes\neg$-categories.) The hierarchy of direct models corresponds to the following hierarchy by means of three reflections, one at each level:

λ_C-models

cartesian computational models

monads

2 *thunk*-*force*-categories

I shall now describe *thunk*-*force*-categories and their relation to monads. As we shall see, each monad generates a *thunk*-*force*-category, and every *thunk*-*force*-category arises in this way. We shall see also that a *thunk*-*force*-category has exactly that part of the structure of its generating monad that we need for semantics.

Definition 1. A *thunk-force-category* is

- A category K
- A functor $L : K \longrightarrow K$
- A transformation[1] $thunk : Id \longrightarrow L$
- A natural transformation $force : L \stackrel{\cdot}{\longrightarrow} Id$

such that $thunk L$ is a natural transformation $L \stackrel{\cdot}{\longrightarrow} L^2$, and

$$\begin{tikzcd}
Id \arrow[r, "thunk"] \arrow[d, "thunk"'] & L \arrow[d, "Lthunk"] & Id \arrow[r, "thunk"] \arrow[dr, "id"'] & L \arrow[d, "force"] & L \arrow[r, "thunk L"] \arrow[dr, "id"'] & L^2 \arrow[d, "Lforce"] \\
L \arrow[r, "thunk L"] & L^2 & & Id & & L
\end{tikzcd}$$

Example. The category *Pfn* of sets and partial functions. For a set A, we define $LA = \{\perp\} \cup \{\llcorner a \lrcorner : a \in A\}$. For a partial function $f : A \rightharpoonup B$, we define $Lf : LA \longrightarrow LB$ as the total function that sends

[1] by a transformation from a functor $F : C \longrightarrow D$ to a functor $G : C \longrightarrow D$, I mean a map that sends each object A of C to an arrow $FA \longrightarrow GA$

- $\llcorner a \lrcorner$ to $\llcorner fa \lrcorner$ if f is defined for a
- $\llcorner a \lrcorner$ to $\perp$ if f is not defined for a
- $\perp$ to $\perp$

We define $thunk x = \llcorner x \lrcorner$ and $force : LA \rightharpoonup A$ as the partial function that sends $\llcorner a \lrcorner$ to a and is undefined for $\perp$. As you can easily check, this is a *thunk*-*force*-category.

Note that *thunk*-*force*-categories are algebraic. Note also that L forms a comonad on K with $thunk L$ as the comultiplication and *force* as the counit.

Definition 2. *Tf* is defined as the obvious category whose objects are the *thunk*-*force*-categories, and whose morphisms are functors that strictly preserve L, *thunk*, and *force*.

Definition 3. *Monad* is defined as the obvious category such that an object is a category C together with a monad on C, and a morphism is a functor that strictly preserves the monad data (which are: the endofunctor, the multiplication, and the unit).

So we consider monads as categories with algebraic structure. Now we turn to the main result for monads and *thunk*-*force*-categories. To help understanding the result, let's recall what a reflection is. In [Lan71] on page 87, Mac Lane writes

> A left adjoint to an inclusion functor (of a full subcategory) is called a *reflection*

We adopt the slightly more general definition by Johnstone [Joh92]:

> A *reflection* is an adjunction for which the counit map ε_B is an isomorphism for all B. (This is equivalent to saying that [the right adjoint] G is *full* and *faithful*...—see [Mac Lane 1971], p. 88, Theorem 1.)

If there is a reflection with a right adjoint $C \longrightarrow D$, then I write $C \lhd D$.

Theorem 4. *There is a reflection*

$$Tf \quad \lhd \quad Monad$$

To prove the theorem, we define an intermediate category *Adj*, which is—as we shall see—equivalent to *Monad*.

Definition 5. A *Kleisli adjunction* is an adjunction whose left adjoint is the identity on objects. The category *Adj* is defined as follows. Objects are Kleisli adjunctions. A morphism from $F \dashv G : K \longrightarrow C$ to $F' \dashv G' : K' \longrightarrow C'$ is a pair of functors $h : C \longrightarrow C'$ and $H : K \longrightarrow K'$ such that h strictly preserves the unit, H strictly preserves the counit, and

$$\begin{tikzcd} C \arrow[r, "h"] \arrow[d, "F"'] & C' \arrow[d, "F'"] \\ K \arrow[r, "H"] & K' \end{tikzcd} \qquad \begin{tikzcd} C \arrow[r, "h"] & C' \\ K \arrow[r, "H"] \arrow[u, "G"] & K' \arrow[u, "G'"'] \end{tikzcd}$$

You can easily check the following: If the two squares in definition 5 commute, then h preserves the unit if and only if H preserves the counit, and either is equivalent to saying that h and H together strictly preserve the adjunction iso.

Lemma 6. *The construction of the Kleisli category extends to an equivalence*

$$\mathit{Monad} \quad \simeq \quad \mathit{Adj}$$

Proof. For an object $C = (C, T)$ of Monad, the required adjunction is the well-known adjunction $F_T \dashv G_T : C_T \longrightarrow C$ like in [Lan71]. Thus we get an obvious functor $\mathit{Kleisli} : \mathit{Monad} \longrightarrow \mathit{Adj}$ For a Kleisli adjunction $F \dashv G : K \longrightarrow C$ of Adj with unit η and counit ε, the required monad has the functor $GF : C \longrightarrow C$, the unit η and the multiplication $G\varepsilon F$. Thus we get an obvious functor $X : \mathit{Adj} \longrightarrow \mathit{Monad}$ Trivially, we have $X\mathit{Kleisli} = \mathit{Id}_{\mathit{Monad}}$. Now we prove that $\mathit{Kleisli}X \cong \mathit{Id}_{\mathit{Adj}}$. Suppose that $F \dashv G : K \longrightarrow C$ is a Kleisli adjunction. Applying $\mathit{Kleisli}X$ yields $F_T \dashv G_T : C_T \longrightarrow C$, where T is the monad induced by the adjunction $F \dashv G$. So we have the unique comparison functor $! : C_T \longrightarrow K$ like in [Lan71], page 144, theorem 2, where ! is called L. For all objects A we have $!A = A$, and for all elements f of $C_T(A, B)$, which is equal to $C(A, GB)$, we have $!f = f^\flat$ where $\flat$ is the obvious iso $C(A, GB) \cong K(FA, B)$. Because F is the identity on objects, ! is an isomorphism of categories (proving this is left as an exercise). Let $E_{F \dashv G} = (\mathit{Id}_C, !)$. As you can easily check, $E_{F \dashv G}$ is an iso from $F_T \dashv G_T$ to $F \dashv G$. It remains to prove that E is natural in $F \dashv G$. This is left as an exercise. ☐

The next definition is the key to proving theorem 4:

Definition 7. A morphism $f : A \longrightarrow B$ of a *thunk-force*-category K is called *thunkable* if

$$\begin{tikzcd}
A \arrow[r, "\mathit{thunk}"] \arrow[d, "f"'] & LA \arrow[d, "Lf"] \\
B \arrow[r, "\mathit{thunk}"] & LB
\end{tikzcd}$$

ΘK is the subcategory of K determined by all objects and the thunkable morphisms.

In *Pfn*, as you can easily check, the thunkable morphisms are the total functions.

Proof of theorem 4. By lemma 6 it is enough to prove a reflection $\mathit{Tf} \lhd \mathit{Adj}$. We shall define a reflection

$$\begin{tikzcd}
\mathit{Tf} \arrow[r, shift left, "i"] \arrow[r, phantom, "\top"] & \mathit{Adj} \arrow[l, shift left, "j"]
\end{tikzcd}$$

First we define i. Suppose that K is a *thunk-force*-category, and that *inc* is the inclusion $\Theta K \longrightarrow K$. As you can easily check, we have a Kleisli adjunction

$$\begin{tikzcd} \Theta K \arrow[r, shift left, "inc"] \arrow[r, phantom, "\perp"] & K \arrow[l, shift left, "L"] \end{tikzcd}$$

with unit *thunk* and counit *force*. Let iK be this adjunction. Now for the morphism part of i. Suppose that $H : K \longrightarrow K'$ is a morphism of *thunk-force*-categories. Because H strictly preserves L and *thunk*, H preserves thunkable morphisms. So H has a restriction $h : \Theta K \longrightarrow \Theta K'$. Let $iH = (h, H)$. This is obviously a morphism of Adj.

Now we define j. Suppose that $F \dashv G : K \longrightarrow C$, with unit η and counit ε, is a Kleisli adjunction. Now we define

$$\begin{aligned} L &=_{\mathrm{def}} FG \\ \mathit{thunk} &=_{\mathrm{def}} F\eta \\ \mathit{force} &=_{\mathrm{def}} \varepsilon \end{aligned}$$

and let $j(F \dashv G) = (K, L, \mathit{thunk}, \mathit{force})$. As you can easily check, $j(F \dashv G)$ is a *thunk-force*-category. The morphism part of j is obvious.

As you can easily check, $ji = Id_{Tf}$. Therefore we define the counit $ji \longrightarrow Id_{Tf}$ of the reflection as the identity on Id_{Tf}.

Now for the unit of the reflection. Suppose that $F \dashv G : K \longrightarrow C$ is a Kleisli adjunction. Then $ij(F \dashv G)$ is the adjunction $inc \dashv L : K \longrightarrow \Theta K$. Suppose that f is a morphism of C. Then the square expressing that Ff is thunkable is the image of the square $f; \eta = \eta; GFf$ under F. So F has a corestriction to ΘK. We define $U_{F \dashv G} = (F : C \longrightarrow \Theta K, Id_K)$. As you can easily check, $U_{F \dashv G}$ is a morphism from $F \dashv G : K \longrightarrow C$ to $ij(F \dashv G)$. As you can check by simple arrow chasing, $U_{F \dashv G}$ is natural in $F \dashv G$.

It remains to check the triangular identities of the reflection. Because the counit is the identity, we need to check only that $U_{iK} = Id_{iK}$ and $jU_{F \dashv G} = Id_{j(F \dashv G)}$. Checking these two equations is straightforward. $\square$

We shall now see that Moggi's semantics of the computational lambda-calculus uses the monad only via the generated L, *thunk*, and *force*. Suppose that C is a λ_C-model whose monad is $T = (T, \mu, \eta)$. Let K be the *thunk-force*-category that results from sending C through the right adjoint of the reflection $Monad \lhd Tf$. So

$$\begin{aligned} K &= C_T \\ L &= F_T G_T \\ \mathit{thunk} &= F_T \eta \\ \mathit{force}_A &= id^C_{TA} \end{aligned}$$

As you can easily check, these for equations translate Moggi's two semantic rules of the computational lambda-calculus that use only T, μ, and η, into

$$\frac{[\![\Gamma \vdash M : A]\!] = f : \Gamma \longrightarrow A}{[\![\Gamma \vdash [M] : TA]\!] \;=\; \Gamma \xrightarrow{thunk} L\Gamma \xrightarrow{Lf} LA} \qquad \frac{[\![\Gamma \vdash M : TA]\!] \;=\; f : \Gamma \longrightarrow A}{[\![\Gamma \vdash \mu(M) : A]\!] \;=\; \Gamma \xrightarrow{f} LA \xrightarrow{force} A}$$

So *thunk-force*-categories have all the structure that we need from monads. Moreover, *thunk-force*-categories don't have more structure than we need, because L, *thunk*, and *force* are denotable. For *thunk* and *force*, this is obvious. To see it for L, suppose that $x : A \vdash M : B$ denotes $f : A \longrightarrow B$. Then

$$y : LA \vdash [let\, x = \mu(y)\; in\, M] : LB$$

denotes Lf, as becomes clear from the semantics of *let* in the next section. So we can conclude that *thunk-force*-categories have exactly the structure that we need from a monad.

Because of the reflection $Tf \lhd Monad$, *thunk-force*-categories correspond to a full subcategory of *Monad*. We shall now see which subcategory.

Definition 8. A monad T with unit η fulfils the *equalizing requirement* if, for each object A, η_A is an equalizer of η_{TA} and $T\eta_A$. The category $Monad_{eq}$ is defined as the full subcategory of *Monad* determined by the objects (C, T) such that T fulfils the equalizing requirement.

Theorem 9. *There is an equivalence of categories*

$$Monad_{eq} \;\simeq\; Tf$$

To prove this theorem, we use an intermediate category Adj_{eq}.

Definition 10. Adj_{eq} is defined as the full subcategory of *Adj* determined by the objects $F \dashv G$ such that, if η stands for the unit, then for each object A, η_A is an equalizer of η_{GFA} and $GF\eta_A$.

Lemma 11. *There is an equivalence of categories*

$$Monad_{eq} \;\simeq\; Adj_{eq}$$

This follows directly from lemma 6.

Lemma 12. *Suppose that $F \dashv G : K \longrightarrow C$ is a Kleisli adjunction with defining isomorphism $\sharp : K(FA, B) \cong C(A, GB)$. Then an element f of $K(A, B)$ is thunkable if and only if*

$$\begin{tikzcd} A \arrow[r, "f^\sharp"] \arrow[d, "f^\sharp"'] & GFB \arrow[d, "\eta"] \\ GFB \arrow[r, "GF\eta"'] & GFGB \end{tikzcd}$$

Proof. We apply the inverse of $\sharp$ to either path of the diagram. As you can easily check, we get

$$\begin{tikzcd} A \arrow[r, "F\eta"] \arrow[d, "f"'] & FGA \arrow[d, "FGf"] \\ B \arrow[r, "F\eta"'] & FGB \end{tikzcd}$$

This is the square that states that f is thunkable. $\square$

Lemma 13. *An object $F \dashv G : K \longrightarrow C$ of Adj is in Adj_{eq} if and only if F is faithful and every thunkable morphism of K is in the image of F.*

Proof. Suppose that $F \dashv G : K \longrightarrow C$ is a Kleisli adjunction. For the 'only if', let $F \dashv G$ be an object of Adj_{eq}. Suppose that f is a thunkable element of $K(A, B)$. By lemma 12, we have $f^\sharp; \eta_{GFB} = f^\sharp; GF\eta_B$. Because η_B is an equalizer of η_{GFB} and $GF\eta_B$, there is a unique $g : A \longrightarrow B$ such that $g; \eta_B = f^\sharp$. As you can easily check, the inverse $\flat$ of $\sharp$ sends the equation $g; \eta_B = f^\sharp$ to $Fg = f$. So every thunkable morphism is the image under F of exactly one morphism. Because all morphisms in the image of F are thunkable, F is faithful.

Now for the 'if'. Because η is natural, we have

$$\eta_B; \eta_{GFB} \quad = \quad \eta_B; GF\eta_B$$

for all objects B. Let $g \in C(A, TB)$ such that

$$g; \eta_{GFB} \quad = \quad g; GF\eta_B$$

We need a unique $f \in C(A, B)$ such that $f; \eta = g$. As you can easily check, $\flat$ sends the equation $f; \eta = g$ to $Ff = g^\flat$. By lemma 12, $g^\flat$ is thunkable. So there is exactly one solution f. $\square$

Proof of theorem 9. By lemma 11 it is enough to prove that $Tf \simeq Adj_{eq}$. First we prove that $i : Tf \longrightarrow Adj$ has a corestriction to Adj_{eq}. Suppose that K is a *thunk-force*-category. By definition iK is equal to $inc \dashv L : K \longrightarrow \Theta K$. This, by lemma 13, is an object of Adj_{eq}. It remains to prove that the unit U of the reflection $Tf \lhd Adj$ restricts to an iso $Id_{Adj_{eq}} \cong ij$. Because $U_{F \dashv G : K \longrightarrow C} = (F : C \longrightarrow \Theta K, Id_K)$, this amounts to proving that $F : K \longrightarrow \Theta K$ is an iso. This follows directly from lemma 13. $\square$

3 *thunk-force*-$\otimes$-categories

In this section, we shall define *thunk-force*-$\otimes$-categories—the direct models that correspond to cartesian computational models. Our definition of *thunk-force*-$\otimes$-categories depends on symmetric premonoidal categories. The latter generalise symmetric monoidal categories in that the product $\otimes$ does not have to be a bifunctor, but only a functor in either argument. We shall now introduce symmetric premonoidal categories by means of binoidal categories. (For more on symmetric premonoidal categories, see [PR97].)

Definition 14. A *binoidal category* is

- A category C
- For each object A, a functor $A \otimes (-) : C \longrightarrow C$
- For each object B, a functor $(-) \otimes B : C \longrightarrow C$

such that for all objects A and B

$$(A \otimes (-))(B) \quad = \quad ((-) \otimes B)(A)$$

For the joint value, we write $A \otimes B$, or short AB.

Definition 15. A morphism $f : A \longrightarrow A'$ of a binoidal category is called *central* if for each $g : B \longrightarrow B'$

$$\begin{tikzcd} AB \arrow[r, "fB"] \arrow[d, "Ag"'] & A'B \arrow[d, "A'g"] \\ AB' \arrow[r, "fB'"] & A'B' \end{tikzcd} \qquad \begin{tikzcd} AB \arrow[r, "fB"] \arrow[d, "Ag"'] & A'B \arrow[d, "A'g"] \\ AB' \arrow[r, "fB'"] & A'B' \end{tikzcd}$$

The *centre* of a binoidal category is the subcategory of all objects and central morphisms.

Definition 16. A *symmetric premonoidal category* is

- A binoidal category C
- An object I of C
- Four natural isomorphisms $A(BC) \cong (AB)C$, $IA \cong A$, $AI \cong A$, and $AB \cong BA$ with central components that fulfil the coherence conditions known from symmetric monoidal categories.

The symmetric monoidal categories are the symmetric premonoidal categories that have only central morphisms. As you can easily check, the natural associativity implies that $A \otimes (-)$ and $(-) \otimes A$ preserve central morphisms. Therefore

Proposition 17. *The centre of a symmetric premonoidal category is a symmetric monoidal category.*

Definition 18. Suppose that C and D are symmetric premonoidal categories. Then a functor from C to D is *strict symmetric premonoidal* if it sends central morphisms to such and strictly preserves the multiplication, the unit, and the four structural isomorphisms.

Definition 19. A *thunk-force-⊗-category* K is

- A *thunk-force*-category K
- A symmetric premonoidal structure on K
- Finite products on ΘK that agree with the symmetric premonoidal structure.

such that ΘK is a subcategory of the centre.

Example. We continue our example *Pfn*. Because $\Theta Pfn = Set$ the finite products are obvious. For $f : A \longrightarrow A'$ and $g : B \longrightarrow B'$ we define $f \otimes g : A \otimes A' \longrightarrow B \otimes B'$ as the partial function that sends (a, b) to $(f(a), g(b))$ if f is defined for a and g is defined for b, and is undefined for (a, b) otherwise.

Obviously a *thunk-force*-⊗-category is a Freyd category with $C = \Theta K$ and the inclusion $\Theta K \longrightarrow K$ as F. Now we add three rules to our semantics of λ_C-terms, which so far has rules for μ and $[-]$ (let δ be the diagonal of the cartesian product of ΘK):

$$[\![x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i]\!] \; = \; \pi_i : A_1 \otimes \cdots \otimes A_n \longrightarrow A_i$$

$$\frac{[\![\Gamma \vdash M : A]\!] \; = \; f : \Gamma \longrightarrow A \qquad [\![\Gamma \vdash N : B]\!] \; = \; g : \Gamma \longrightarrow B}{[\![\Gamma \vdash (M, N) : A * B]\!] \; = \; \Gamma \xrightarrow{\delta} \Gamma \otimes \Gamma \xrightarrow{f \otimes \Gamma} A \otimes \Gamma \xrightarrow{A \otimes g} A \otimes B}$$

$$\frac{[\![\Gamma \vdash M : A]\!] \; = \; f : \Gamma \longrightarrow A \qquad [\![\Gamma, x : A \vdash N : B]\!] \; = \; g : \Gamma \otimes A \longrightarrow B}{[\![\Gamma \vdash \mathit{let}\, x = M \,\mathit{in}\, N : B]\!] \; = \; \Gamma \xrightarrow{\delta} \Gamma \otimes \Gamma \xrightarrow{\Gamma \otimes f} \Gamma \otimes A \xrightarrow{g} B}$$

We shall now define two useful concepts for *thunk-force*-⊗-categories. (For an object A of a *thunk-force*-⊗-category K, let $!_A : A \longrightarrow I$ be the unique element of $(\Theta K)(A, I)$.)

Definition 20. Suppose that $f : A \longrightarrow B$ is morphism of a *thunk-force*-⊗-category. f is *copyable* if,

$$\begin{tikzcd} A \arrow[r, "\delta"] \arrow[d, "f"'] & AA \arrow[d, "f\,f"] \\ B \arrow[r, "\delta"] & BB \end{tikzcd}$$

f is *discardable* if

$$\begin{tikzcd} A \arrow[r, "!"] \arrow[d, "f"'] & I \arrow[d, "id"] \\ B \arrow[r, "!"] & I \end{tikzcd}$$

The next proposition has two purposes. First, it helps checking that a structure is a *thunk-force-*$\otimes$-category. Second—as we shall see—it implies that *thunk-force-*$\otimes$-categories are algebraic.

Proposition 21. *Suppose that K is a thunk-force-category together with a binoidal product, an object I, and transformations $\delta_A : A \longrightarrow AA$ and $!_A : A \longrightarrow I$. Then K determines a thunk-force-$\otimes$-category if and only if*

1. *The components of thunk are central, and all morphisms of the form Lf are central.*
2. *All morphisms of the form $A \otimes thunk$, $thunk \otimes A$, $A \otimes Lf$, and $Lf \otimes A$ are thunkable.*
3. *The components of thunk, and all morphisms of the form Lf, are copyable and discardable.*
4. *The components of δ and $!$ are thunkable.*
5. *δ and $!$ determine a comonoid.*
6. *We have*

$$\begin{tikzcd} AB \arrow[r, "\delta"] \arrow[dr, "id"'] & (AB)(AB) \arrow[d, "\pi\pi'"] \\ & AB \end{tikzcd} \quad \textit{where} \quad \begin{cases} \pi = AB \xrightarrow{A!} AI \cong A \\ \pi' = AB \xrightarrow{!B} IB \cong B \end{cases}$$

Proof. First we check the 'only if'. Conditions 1, 2 and 3 hold because the components of *thunk* are in ΘK, and all morphisms of the form Lf are in ΘK. The remaining conditions are obvious.

Now for the 'if'. Condition 1 implies that all thunkable morphisms are central. To see this, let $f \in \Theta K(A, B)$. Then for every $g \in K(A', B')$ we have

$$\begin{aligned} A \otimes g; f \otimes B' &= A \otimes g; f \otimes B'; \mathit{thunk} \otimes B'; \mathit{force} \otimes B' \\ &= A \otimes g; \mathit{thunk} \otimes B'; Lf \otimes B'; \mathit{force} \otimes B' \\ &= \mathit{thunk} \otimes A'; Lf \otimes A'; LB \otimes g; \mathit{force} \otimes B' \\ &= f \otimes A'; \mathit{thunk} \otimes A'; LB \otimes g; \mathit{force} \otimes B' \\ &= f \otimes A'; B \otimes g; \mathit{thunk} \otimes B'; \mathit{force} \otimes B' \\ &= f \otimes A'; B \otimes g \end{aligned}$$

Condition 2 implies that ΘK is closed under $\otimes$. To see this, let $f \in \Theta K(A, B)$. Then

$$\begin{array}{rcl} A \otimes f; \mathit{thunk} & = & A \otimes f; \mathit{thunk}; L(A \otimes \mathit{thunk}); L(A \otimes \mathit{force}) \\ & = & A \otimes f; A \otimes \mathit{thunk}; \mathit{thunk}; L(A \otimes \mathit{force}) \\ & = & A \otimes \mathit{thunk}; A \otimes Lf; \mathit{thunk}; L(A \otimes \mathit{force}) \\ & = & A \otimes \mathit{thunk}; \mathit{thunk}; L(A \otimes Lf); L(A \otimes \mathit{force}) \\ & = & A \otimes \mathit{thunk}; \mathit{thunk}; L(A \otimes \mathit{force}); L(A \otimes f) \\ & = & \mathit{thunk}; L(A \otimes \mathit{thunk}); L(A \otimes \mathit{force}); L(A \otimes f) \\ & = & \mathit{thunk}; L(A \otimes f) \end{array}$$

So ΘK and $\otimes$ together determine a binoidal subcategory of the centre. In particular, $\otimes$ determines a bifunctor $\Theta K \times \Theta K \longrightarrow \Theta K$. Condition 3 implies that every thunkable morphism is copyable and discardable (the proof is left as an exercise). The remaining conditions imply that ΘK together with $\otimes$, I, δ, and $!$ is a category with finite products (the proof is left as an exercise). Every category with finite products determines a symmetric monoidal category (see [Lan71], p. 159). Because the symmetric monoidal product of ΘK agrees with the binoidal product on K, the symmetric monoidal structure on ΘK extends to a symmetric premonoidal structure on K. $\square$

You can easily express all conditions in proposition 21 by equations that are all-quantified over hom-sets. So we have the following corollary:

Corollary 22. *thunk-force-$\otimes$-categories are algebraic.*

Now we turn to generalising theorem 4, which states that there is a reflection $\mathit{Monad} \lhd \mathit{Tf}$. We shall define a category $\mathit{Tf}\otimes$ of *thunk-force-$\otimes$*-categories, and a category Ccm of computational cartesian models, and prove that there is a reflection $\mathit{Tf} \otimes \lhd \mathit{Ccm}$.

Definition 23. $\mathit{Tf}\otimes$ is defined as the obvious category formed by the *thunk-force-$\otimes$*-categories and the functors that strictly preserve all operators.

Note that it is not obvious that morphisms of *thunk-force-$\otimes$*-categories preserve central maps. For suppose that $F : K \longrightarrow K'$ is a morphism of *thunk-force-$\otimes$*-categories, and f is a central morphism of K. That f is central means that f commutes in the sense of definition 15 with all morphisms g of K. Therefore, Ff commutes with all morphisms of the form Fg. But K' may have morphisms that are not in the image of F. Fortunately, we have the following proposition:

Proposition 24. *Suppose that K is a thunk-force-$\otimes$-category. A morphism $f \in K(A, A')$ is central if for all $B \in Ob(K)$*

$$\begin{tikzcd} A \otimes LB \arrow[r, "f \otimes LB"] \arrow[d, "A \otimes \mathit{force}_B"'] & A' \otimes LB \arrow[d, "A' \otimes \mathit{force}_B"] \\ A \otimes B \arrow[r, "f \otimes B"] & A' \otimes B \end{tikzcd}$$

Proof. Let $g \in K(B, B')$. Then $g = \mathit{thunk}; \mathit{force}; g = \mathit{thunk}; Lg; \mathit{force}$. Because *thunk* and Lg are thunkable and therefore central, they commute with f. Because f commutes with *force* too, f commutes with g. □

Because morphisms of *thunk*-*force*-$\otimes$-categories preserve *force*, morphisms of *thunk*-*force*-$\otimes$-categories preserve central morphisms. So we get the following corollary:

Corollary 25. *Morphisms of thunk-force-$\otimes$-categories are strict symmetric premonoidal functors.*

Definition 26. *Ccm* is defined as the obvious category formed by cartesian computational models and the morphisms of *Monad* that strictly preserve the finite products and the strength.

Theorem 27. *There is a reflection*

$$\mathit{Tf}\otimes \quad \lhd \quad \mathit{Ccm}$$

To prove this, we use an intermediate category. First we describe its objects.

Definition 28. A *I-closed Freyd category* consists of a category C with finite products, a symmetric premonoidal category K, and an adjunction $F \dashv G : K \longrightarrow C$ such that F is an identity-on-objects strict symmetric premonoidal functor. □

We call them *I*-closed Freyd categories because in a Freyd category $F : C \longrightarrow K$ the functor F has a right adjoint if and only if $F(-) \otimes I$ has a right adjoint, and therefore *I*-closed Freyd categories herald closed Freyd categories.

Definition 29. The category *IcFreyd* is defined as follows. The objects are the *I*-closed Freyd categories. A morphism from $F \dashv G : K \longrightarrow C$ to $F' \dashv G' : K' \longrightarrow C'$ is a pair (h, H) that consists of a functor $h : C \longrightarrow C'$ that strictly preserves finite products, and a strict symmetric premonoidal functor $H : K \longrightarrow K'$ such that (h, H) is a morphism of Kleisli adjunctions.

Lemma 30. *There is a reflection*

$$\mathit{Tf}\otimes \quad \lhd \quad \mathit{IcFreyd}$$

Proof. We extend the reflection $j \dashv i : \mathit{Tf} \longrightarrow \mathit{Adj}$ to a reflection $j \dashv i : \mathit{Tf}\otimes \longrightarrow \mathit{IcFreyd}$. First we show that i extends to a functor $\mathit{Tf}\otimes \longrightarrow \mathit{IcFreyd}$. Let K be a *thunk*-*force*-$\otimes$-category. By definition, iK is the adjunction $\mathit{inc} \dashv L : K \longrightarrow \Theta K$. This is obviously an *I*-closed Freyd category. For a morphism $H : K \longrightarrow K'$ of *thunk*-*force*-$\otimes$-categories, $iH : iK \longrightarrow iK'$ is obviously a morphism of *I*-closed Freyd categories.

Now we show that j extends to a functor $\mathit{IcFreyd} \longrightarrow \mathit{Tf}\otimes$. Let $f \dashv G : K \longrightarrow C$ be an *I*-closed Freyd category, where η is the unit and ε is the counit. By definition, for $j(F \dashv G)$ is K together with $L = FG$, $\mathit{thunk} = F\eta$, and $\mathit{force} = \varepsilon$. For each object A, let δ_A and $!_A$ be the images under F of the diagonal $A \longrightarrow AA$ and the unique arrow $A \longrightarrow 1$, respectively. With proposition 21 we prove that the *thunk-force*-category K together with $\otimes$, 1, δ, and ! determines a *thunk-force*-$\otimes$-category. Condition 1 of proposition 21 holds because F, which is a strict symmetric premonoidal functor, preserves central morphisms, and *thunk* and Lf are in the image of F. Condition 4 holds because all morphisms in the image of F are thunkable, as you can easily check. Condition 2 holds because

$$A \otimes \mathit{thunk} = FA \otimes F\eta = F(A \otimes \eta)$$
$$A \otimes Lf = FA \otimes FGf = F(A \otimes f)$$

Condition 3 holds because all morphisms in the image of F are copyable and discardable, as you can easily check. Checking the remaining conditions is straightforward—I leave it away here. Checking that j sends morphisms of *I*-closed Freyd categories to morphisms of *thunk-force*-$\otimes$-categories is straightforward—I leave it away here.

As you can easily check, we have $ji = \mathit{Id}_{\mathit{Tf}\otimes}$. We define the counit $ji \longrightarrow \mathit{Id}_{\mathit{Tf}\otimes}$ as the identity. Now for the unit. Let's recall the unit U of the reflection $\mathit{Tf} \lhd \mathit{Adj}$: For a Kleisli adjunction $F \dashv G$, we have $U_{F\dashv G} = (F : C \longrightarrow \Theta K, \mathit{Id}_K)$. It remains to prove that, if $F \dashv G$ is an *I*-closed Freyd category, then $U_{F\dashv G}$ forms a morphism of *I*-closed Freyd categories. This amounts to checking that $F : C \longrightarrow \Theta K$ strictly preserves finite products, which is obvious. The naturality of the unit, and the triangular equations, follow from the corresponding results for the reflection $\mathit{Tf} \lhd \mathit{Adj}$. □

Lemma 31. *The construction of the Kleisli category forms an equivalence*

$$\mathit{Ccm} \quad \simeq \quad \mathit{IcFreyd}$$

Proof. By lemma 6, we have functors $\mathit{Kleisli} : \mathit{Monad} \longrightarrow \mathit{Adj}$ and $X : \mathit{Adj} \longrightarrow \mathit{Monad}$ that form an equivalence. First we extend *Kleisli* to a functor $\mathit{Ccm} \longrightarrow \mathit{IcFreyd}$. Let $C = (C, T)$ be a computational cartesian model, where $T = (T, \mu, \eta)$ and t is the strength. We need a symmetric premonoidal structure in C_T. For objects A and B, let

$$A \otimes B \quad =_{\mathrm{def}} \quad A \times B$$

For an object A and element f of $C_T(B, B')$, which is equal to $C(B, TB')$, let

$$A \otimes f \quad =_{\mathrm{def}} \quad A \times B \xrightarrow{A \times f} A \times TB' \xrightarrow{t} T(A \times B')$$

and $f \otimes A$ symmetrically. Now $\otimes$ forms a symmetric premonoidal structure on C_T such that $F_T \dashv G_T : C_T \longrightarrow C$ is an *I*-closed Freyd category (The proof is left as an exercise, as well as the proof that this gives a functor $\mathit{Ccm} \longrightarrow \mathit{IcFreyd}$).

Now we extend X to a functor $X : \mathit{IcFreyd} \longrightarrow \mathit{Ccm}$. Let $F \dashv G : K \longrightarrow C$ be an *I*-closed Freyd category with iso $\sharp : K(FA, B) \cong C(A, GB)$ and counit ε. By definition, X takes the *I*-closed Freyd category to (C, T), where $T = (GF, G\varepsilon F, \eta)$. What we still need is the strength $t : A \times TB \longrightarrow T(A \times B)$. Let

$$t \quad =_{\mathrm{def}} \quad (A \otimes \varepsilon)^{\sharp}$$

This gives us a computational cartesian model (the proof is left as an exercise, as well as the proof that this gives a functor $X : \mathit{Ccm} \longrightarrow \mathit{IcFreyd}$).

To see that the two extended functors are inverse up to natural iso, it is enough to check that the two natural isos in the proof of lemma 6 preserve the new structure. The proof is left as an exercise). $\square$

Proof of theorem 27. By composition of the reflections $\mathit{Tf}\otimes \lhd \mathit{IcFreyd}$ and $\mathit{IcFreyd} \simeq \mathit{Ccm}$. $\square$

Theorem 32. *Moggi's semantics of the computational lambda-calculus in a λ_C-model C agrees with the semantics in the thunk-force-$\otimes$-category generated by C.*

Proving this amounts to checking the semantic rules for variables, pairs, and *let*. Checking this is left as an exercise.

Let Ccm_{eq} be the full subcategory of Ccm determined by the computational cartesian models whose monad fulfils the equalizing requirement. The following theorem follows directly from theorems 9 and 27:

Theorem 33. *There is an equivalence of categories*

$$\mathit{Ccm}_{eq} \;\simeq\; \mathit{Tf}\otimes$$

4 Direct λ_C-models

Finally, we shall define direct λ_C-models—the direct models that correspond to λ_C-models.

Definition 34. A *direct λ_C-model* is a *thunk-force-$\otimes$*-category K together with, for each object A, a functor $A \Rightarrow (-) : K \longrightarrow \Theta K$ and an adjunction

$$\lambda : K(B\otimes A, C) \;\cong\; (\Theta K)(B, A \Rightarrow C)$$

Example. For *Pfn*, we define

$$A \Rightarrow (-) \;=\; \mathit{Pfn}(A, -) : \mathit{Pfn} \longrightarrow \mathit{Set}$$

(recall that $\Theta\mathit{Pfn} = \mathit{Set}$). λ is obvious.

A direct λ_C-model K is obviously a closed Freyd category with $C = \Theta K$ and F as the inclusion. We write *apply* for the counit $(A \Rightarrow B)\otimes A \longrightarrow B$ of the Kleisli exponentials, and *pair* for the unit $A \longrightarrow B \Rightarrow (A\otimes B)$. Here come the remaining two rules of our semantics of the computational lambda-calculus in direct λ_C-models.

$$\frac{[\![\Gamma, x : A \vdash M : B]\!] \;=\; f : \Gamma\otimes A \longrightarrow B}{[\![\Gamma \vdash \lambda x : A.M : A \Rightarrow B]\!] \;=\; \lambda f : \Gamma \longrightarrow A \Rightarrow B}$$

$$\frac{[\![\Gamma \vdash M : A \Rightarrow B]\!] = f : \Gamma \longrightarrow A \Rightarrow B \qquad [\![\Gamma \vdash N : A]\!] = g : \Gamma \longrightarrow A}{[\![\Gamma \vdash MN : B]\!] = \Gamma \xrightarrow{\delta} \Gamma\Gamma \xrightarrow{f\Gamma} (A \Rightarrow B)\Gamma \xrightarrow{(A \Rightarrow B)g} (A \Rightarrow B)A \xrightarrow{apply} B}$$

The next proposition helps checking that a structure is a direct λ_C-model. We shall need the proposition in later proofs.

Proposition 35. *Suppose that K is a thunk-force-$\otimes$-category together with a functor $A \Rightarrow (-) : K \longrightarrow K$, a natural transformation $apply : (A \Rightarrow B)A \stackrel{\cdot}{\longrightarrow} B$, and a transformation $pair : B \longrightarrow A \Rightarrow (BA)$. Then for each object A there is an adjunction $A \otimes (-) \dashv A \Rightarrow (-) : K \longrightarrow \Theta K$ with unit pair and counit apply if and only if*

1. *All components of pair are thunkable., and all morphisms of the form $A \Rightarrow f$ are thunkable.*
2. *pair is natural for all components of thunk, and for all morphisms of the form Lf.*
3. *We have*

$$\begin{tikzcd}
AB \arrow[r, "pair B"] \arrow[dr, "id"'] & (B \Rightarrow (AB))B \arrow[d, "apply"] \\
& AB
\end{tikzcd}
\qquad
\begin{tikzcd}
B \Rightarrow A \arrow[r, "pair"] \arrow[dr, "id"'] & B \Rightarrow ((B \Rightarrow A)B) \arrow[d, "B \Rightarrow apply"] \\
& B \Rightarrow A
\end{tikzcd}$$

Proof. The 'only if' is obvious. The non-obvious part of the 'if' is to prove that condition 2 implies that *pair* is natural for all thunkable morphisms. To see this, let $f : A \longrightarrow A'$ be a thunkable morphism, and B an object for which we consider the adjunction $(-) \otimes B \dashv B \Rightarrow (-)$. Then

$$\begin{aligned}
f; pair &= f; pair; B \Rightarrow (thunk B); B \Rightarrow (force B) \\
&= f; thunk; pair; B \Rightarrow (force B) \\
&= thunk; Lf; pair; B \Rightarrow (force B) \\
&= pair; B \Rightarrow (thunk B); B \Rightarrow ((Lf)B); B \Rightarrow (force B) \\
&= pair; B \Rightarrow (fB); B \Rightarrow (thunk B); B \Rightarrow (force B) \\
&= pair; B \Rightarrow (fB)
\end{aligned}$$

□

Because *thunk*-*force*-$\otimes$-categories are algebraic, and all conditions in proposition 35 are algebraic, we have the following corollary:

Proposition 36. *Direct λ_C-models are algebraic.*

Definition 37. We define three categories as follows:

- $D\lambda_C$ is defined as the obvious category formed by direct λ_C-models and the morphisms of *thunk-force-*$\otimes$-categories that preserve Kleisli exponentials.
- λ_C is defined as the obvious category formed by λ_C-models and the morphisms of computational cartesian models that preserve T-exponentials.
- *CFreyd* is defined as the obvious category formed by closed Freyd categories and morphisms of *I*-closed Freyd categories that strictly preserve Kleisli exponentials.

Theorem 38. *There is a reflection*

$$D\lambda_C \quad \lhd \quad \mathit{CFreyd}$$

Proof. We extend the reflection $j \dashv i : \mathit{Tf}\otimes \longrightarrow \mathit{IcFreyd}$ to a reflection $j \dashv i : D\lambda_C \longrightarrow \mathit{CFreyd}$ Extending i is obvious: For a direct λ_C-model K, iK is $inc \dashv L : K \longrightarrow \Theta K$. Now for j. Let $F \dashv C : K \longrightarrow C$ be a closed Freyd category. Let

$$\begin{aligned} A \Rightarrow' (-) \quad &=_{\mathrm{def}} \quad K \xrightarrow{A\Rightarrow(-)} C \xrightarrow{F} \Theta K \\ \mathit{pair}' \quad &=_{\mathrm{def}} \quad F\mathit{pair} \\ \mathit{apply}' \quad &=_{\mathrm{def}} \quad \mathit{apply} \end{aligned}$$

With proposition 35 we prove that the new data determine the required adjunction. Condition 1 holds because all morphisms of the form Fg are thunkable. Condition 2 holds because, as you can easily check, pair' is natural for all morphisms of the form Fg. Checking the remaining conditions of proposition 35 is very easy—I leave it away here.

Obviously we have $ji = Id_{D\lambda_C}$. So it remains to check that the unit of the reflection extends. This means checking that for each direct λ_C-model $F \dashv G : K \longrightarrow C$, the morphism of *I*-closed Freyd categories (F, Id_K) into $inc \dashv L : K \longrightarrow \Theta K$ preserves Kleisli exponentials. Checking this is straightforward and left as an exercise. $\square$

Theorem 39. *The construction of the Kleisli category forms an equivalence*

$$\mathit{CFreyd} \quad \simeq \quad \lambda_C$$

Proof. By lemma 31 we have functors $\mathit{Kleisli} : \mathit{Ccm} \longrightarrow \mathit{IcFreyd}$ and $X : \mathit{IcFreyd} \longrightarrow \mathit{Ccm}$ that form an equivalence of categories. First we extend *Kleisli* to a functor $\lambda_C \longrightarrow \mathit{CFreyd}$. Let C be a λ_C-model whose monad is $T = (T, \mu, \eta)$ and whose T-exponentials are determined by

$$\lambda : C(AB, TC) \cong C(A, (TC)^B)$$

Let C' be the full subcategory of C whose objects are those of the form TA. For $f \in C'(TA, TA')$ and an object B, let

$$f^B \quad =_{\mathrm{def}} \quad \lambda\left((TA)^B \times B \xrightarrow{ev} TA \xrightarrow{f} TA'\right)$$

As you can easily check, $(-)^A$ determines a functor $C' \longrightarrow C$. Let

$$A \Rightarrow (-) \quad =_{\text{def}} \quad (G_T(-))^A$$

Now we claim that λ determines a natural iso

$$\lambda : C_T(FA \otimes B, C) \;\cong\; C(A, B \Rightarrow C)$$

What we must check is the naturality of λ. Checking this is straightforward—I leave it away here.

Now we extend X to a functor $\mathit{CFreyd} \longrightarrow \lambda_C$. Suppose that $F \dashv G : K \longrightarrow C$ is a direct λ_c-model, and $\sharp : K(FA, C) \cong C(A, GC)$ is the obvious adjunction iso. By definition, the computational cartesian model $X(F \dashv G)$ has the monad $T = (GF, G\varepsilon F, \eta)$. Let

$$\begin{aligned} (TB)^A \quad &=_{\text{def}} \quad A \Rightarrow B \\ ev \quad &=_{\text{def}} \quad \mathit{apply}^\sharp \end{aligned}$$

This determines T-exponentials on C—checking this is left as an exercise. Checking the morphism part of X is straightforward, so I leave it away here.

Obviously $X\mathit{Kleisli} : \lambda_C \longrightarrow \lambda_C$ doesn't change T-exponentials. Therefore $X\mathit{Kleisli} = Id_{\lambda_C}$.

It remains to prove that the components of the natural iso $\mathit{Kleisli}X \cong Id_{\mathit{IcFreyd}}$ preserve Kleisli exponentials. Let $F \dashv G : K \longrightarrow C$ be a direct λ_c-model, and let $\flat : C(A, GB) \cong K(FA, B)$ be the obvious natural iso. Applying $\mathit{Kleisli}X$ yields the direct λ_c-model $F_T \dashv G_T : C_T \longrightarrow C$ where $T = (GF, G\varepsilon F, \eta)$. The component of the iso $\mathit{Kleisli}X \cong Id_{\mathit{IcFreyd}}$ at $F \dashv G$ is $(Id_C, !)$ where $!$ is the unique comparison functor $C_T \longrightarrow K$. Now we prove that $(Id_C, !)$ preserves Kleisli exponentials. For every object B let $B \Rightarrow_T (-)$ and $B \Rightarrow (-)$ be the Kleisli exponential functors of $F_T \dashv G_T$ and $F \dashv G$, respectively, and let λ_T and λ be the natural isos for the respective Kleisli exponentials. First we prove that

$$Id_C(B \Rightarrow_T (-)) \quad = \quad !B \Rightarrow !(-)$$

For an object A we get

$$Id_C(B \Rightarrow_T A) \;=\; B \Rightarrow_T A \;=\; (TA)^B \;=\; B \Rightarrow A \;=\; !B \Rightarrow !A$$

For $f \in C_T(A, A')$ we have

$$\begin{aligned} Id_C(B \Rightarrow_T f) \quad &= \quad (G_T f)^B \;=\; (G(f^\flat))^B \\ !B \Rightarrow !f \quad &= \quad B \Rightarrow f^\flat \end{aligned}$$

We prove that $(G(f^\flat))^B = B \Rightarrow f^\flat$ by checking

$$\begin{tikzcd} (TA)^B \times B \arrow[r, "ev"] \arrow[d, "(B \Rightarrow f^\flat) \times B"'] & TA \arrow[d, "G(f^\flat)"] \\ (TA')^B \times B \arrow[r, "ev"] & TA' \end{tikzcd}$$

This is true because

$$\begin{array}{rcl}
(ev; G(f^\flat))^\flat & = & F(ev; G(f^\flat)); \varepsilon \;=\; F(apply^\sharp; G(f^\flat)); \varepsilon \\
& = & F(\eta; Gapply; G(f^\flat)); \varepsilon \;=\; F\eta; FGapply; FG(f^\flat); \varepsilon \\
& = & thunk; Lapply; L(f^\flat); force \;=\; thunk; force; apply; f^\flat \\
& = & apply; f^\flat \;=\; F(B \Rightarrow f^\flat) \otimes B; apply \\
& = & F((B \Rightarrow f^\flat) \times B); ev^\flat \;=\; F((B \Rightarrow f^\flat) \times B); Fev; \varepsilon \\
& = & F((B \Rightarrow f^\flat) \times B; ev); \varepsilon \;=\; ((B \Rightarrow f^\flat) \times B; ev)^\flat
\end{array}$$

It remains to prove that for each $f \in C_T(A \times B, C)$

$$Id_C(\lambda_T f) \;=\; \lambda(!f)$$

We prove this by checking

$$\begin{tikzcd}
(TC)^B \times B \arrow[r, "ev"] & TC \\
A \times B \arrow[u, "\lambda(!f) \times B"] \arrow[ur, "f"'] &
\end{tikzcd}$$

This is true because

$$(\lambda(!f) \times B; ev)^\flat \;=\; (\lambda(f^\flat) \times B; ev)^\flat \;=\; F(\lambda(f^\flat) \times B); apply \;=\; F(\lambda(f^\flat)) \otimes B; apply \;=\; f^\flat$$

□

By composing the reflections $D\lambda_C \lhd CFreyd$ and $CFreyd \simeq \lambda_C$ we get

Theorem 40. *There is a reflection*

$$D\lambda_C \;\lhd\; \lambda_C$$

Theorem 41. *Moggi's semantics of the computational lambda-calculus in a λ_C-model C is equal to the semantics in the direct λ_C-model generated by C.*

Proving this amounts to checking the semantic rules for lambda abstraction and application. Checking this is left as an exercise.

Let $\lambda_{C\,eq}$ be the full subcategory of λ_C determined by the λ_C-models whose monads fulfil the equalizing requirement. Let $CFreyd_{eq}$ be the subcategory of $CFreyd$ determined by the closed Freyd categories whose induced monads fulfil the equalizing requirement The following theorem follows directly from theorems 9, 38, and 39:

Theorem 42.

$$D\lambda_C \;\simeq\; CFreyd_{eq} \;\simeq\; \lambda_{C\,eq}$$

Proposition 43. *Let K be a direct λ_C-model. Then L and $I \Rightarrow (-)$ are naturally isomorphic, and we have*

$$
\begin{tikzcd}
LA \arrow[d, "force"'] & I \Rightarrow A \arrow[l, "\cong"'] \arrow[d, "\cong"] \\
A & I \Rightarrow AI \arrow[l, "apply"']
\end{tikzcd}
\qquad
\begin{tikzcd}
LA \arrow[r, "\cong"] & I \Rightarrow A \\
A \arrow[u, "thunk"] \arrow[ur, "\lambda r"] \arrow[r, "pair"'] & I \Rightarrow (AI) \arrow[u, "\cong"']
\end{tikzcd}
$$

Proof. As you can easily check, each *thunk*-*force*-category K has an adjunction

$$K(A, B) \;\cong\; (\Theta K)(A, LB)$$

with unit *thunk* and counit *force*. So we have a natural iso

$$(\Theta K)(A, LB) \cong K(A, B) \cong K(A \otimes I, B) \cong (\Theta K)(A, I \Rightarrow B)$$

Checking the two diagrams is left as an exercise. □

Therefore, the program transformation that replaces all occurrences of TA with $I \Rightarrow A$, all occurrences of $[M]$ with $\lambda x : I.M$, and all occurrences of $\mu(M)$ with $M*$ preserves meaning up to natural isomorphism. So T, $\mu(-)$, and $[-]$ are redundant. But we may want to keep the three. To see this, note that we want to consider languages with many computational effects. Therefore we may need models with one *thunk*-*force*-structure per computational effect. So the codomain of λ is no longer obvious. In particular, the functor $I \Rightarrow (-)$ may not be isomorphic to the functor L of any computational effect.

5 Thunkable and central programs

By definition of a *thunk*-*force*-$\otimes$-category, every thunkable morphism is central. The converse does not hold: In *Pfn*, for example, every partial function is central, but only the total functions are thunkable. As we shall see later, in $\otimes\neg$-categories all central morphisms are thunkable, but not all morphisms are central. Let's see now when the denotation of a program is thunkable and central, respectively. Suppose that K is a direct λ_C-model K, and that $\Gamma \vdash M : A$ denotes $f : \Gamma \longrightarrow A$ in K. Then, as you can easily check,

$$
\begin{aligned}
\Gamma \vdash \mathit{let}\, x = M\, \mathit{in}\, [y] : LA \quad &\text{denotes} \quad \Gamma \xrightarrow{f} A \xrightarrow{thunk} LA \\
\Gamma \vdash [M] : LA \quad &\text{denotes} \quad A \xrightarrow{thunk} LA \xrightarrow{Lf} LB
\end{aligned}
$$

Now suppose that Δ is an environment that doesn't share variables with Γ, and $\Delta \vdash N : B$ denotes $g : \Delta \longrightarrow B$. Then the two sequents

$$
\begin{aligned}
&\Gamma, \Delta \vdash \mathit{let}\, x = M\, \mathit{in}\, \mathit{let}\, y = N\, \mathit{in}\, (x, y) : A * B \\
&\Gamma, \Delta \vdash \mathit{let}\, y = N\, \mathit{in}\, \mathit{let}\, x = M\, \mathit{in}\, (x, y) : A * B
\end{aligned}
$$

denote, respectively,

$$\Gamma \otimes \Delta \xrightarrow{f \otimes \Delta} A \otimes \Delta \xrightarrow{A \otimes g} A \otimes B$$
$$\Gamma \otimes \Delta \xrightarrow{\Gamma \otimes g} \Gamma \otimes B \xrightarrow{f \otimes B} A \otimes B$$

This directly implies the 'only if'-part of the following proposition:

Proposition 44. *Suppose that K is a direct λ_C-model. The denotation of $\Gamma \vdash M : A$ in K is central if and only if for every program $\Delta \vdash N : B$ such that Δ doesn't share variables with Γ,*

$$\begin{array}{rcl} \Gamma, \Delta & \vdash & \mathit{let}\, x = M \,\mathit{in}\, \mathit{let}\, y = N \,\mathit{in}\, (x, y) \\ & = & \mathit{let}\, y = N \,\mathit{in}\, \mathit{let}\, x = M \,\mathit{in}\, (x, y) \;:\; A * B \end{array}$$

Proof. The proof of the first claim is straightforward. The second claim requires some care. If the denotation of $\Gamma \vdash M : A$ is central, then obviously the equation in the statement of the second claim holds. The converse is not obvious, because K may have morphisms that cannot be denoted by any $\Delta \vdash N : B$. By lemma 24, f is central if it commutes with $g = \mathit{force}_B$ for all objects B. And force_B is denoted by $z : TB \vdash \mu(z) : B$. □

6 $\otimes\neg$-categories as direct λ_C-models

The theory of $\otimes\neg$-categories can be seen as an extension of the theory of direct λ_C-models. The only extra operator is a functor

$$\neg : K^{op} \longrightarrow \Theta K$$

This functor can model a unary type constructor *cont* like in SML of New Jersey. The axioms for $\neg$ imply that

$$x \Rightarrow y \quad = \quad \neg(x \otimes \neg y)$$

For a full definition of $\otimes\neg$-categories, see [Thi97a, Thi97b]. $\otimes\neg$-categories are algebraic, which was observed by Peter Selinger (his control categories and co control-categories [Sel98] are algebraic, and the latter are $\otimes\neg$-categories together with sums). To my surprise I found that in a $\otimes\neg$-category every central morphism is thunkable (see [Füh98]).

7 A direction for further research

Direct λ_C-models may be a good basis for finding direct models for call-by-value programming languages with several computational effects. I would like to keep the theories for several computational effects algebraic, because

- We can do all reasoning by replacing subexpressions along the axioms.
- We can cope with changes of language features by simply adding and removing operators and equations, respectively.
- We have a simple meta-theory—for example, we can form the free algebraic theory generated by a set of operators and equations, adjoin indeterminates, and so on.

Acknowledgments. I am indebted to Hayo Thielecke, whose $\otimes\neg$-categories were the main inspiration for my analysis of direct models. Thanks a lot to John Power for explaining to me premonoidal categories and more, and commenting on my work. Thanks to Peter Selinger for many discussions, in particular for making me aware of the algebraicity-criterion for models. Thanks to Stuart Anderson for commenting on several versions of this article. And a historical remark: Recently, Alex Simpson made me aware of his 1993 LFCS Lab-Lunch talk '(Not very far) Towards algebraic semantics of programming languages' [Sim93]. There he sketched what I call direct models. He had already found the essence of my reflection theorem for monads (theorem 4). But there was no way to transfer this to strong monads, because premonoidal categories had not yet emerged.

References

[Füh98] Carsten Führmann. Relating two models of continuations. submitted, November 1998.

[Joh92] Peter T. Johnstone. *Stone Spaces*. Cambridge studies in advanced mathematics. Cambridge University Press, 1992.

[Lan71] Saunders Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics. Springer-Verlag, 1971.

[Mog88] E. Moggi. Computational lambda-calculus and monads. Technical Report ECS-LFCS-88-66, Edinburgh Univ., Dept. of Comp. Sci., 1988.

[PR97] John Power and Edmund Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 7(5):453–468, October 1997.

[PT97] John Power and Hayo Thielecke. Environments, continuation semantics and indexed categories. In *Proceedings TACS'97*, volume 1281 of *LNCS*, pages 391–414. Springer Verlag, 1997.

[PT98] John Power and Hayo Thielecke. Environments in Freyd categories and κ-categories. submitted, 1998.

[Sel98] Peter Selinger. Control categories. http:// www.math.lsa.umich.edu/ ∼selinger/ papers.html, 1998.

[Sim93] Alex Simpson. Towards algebraic semantics of programming languages. Notes for a talk at the LFCS Lab Lunch, March 1993.

[Thi97a] Hayo Thielecke. *Categorical Structure of Continuation Passing Style*. PhD thesis, University of Edinburgh, 1997.

[Thi97b] Hayo Thielecke. Continuation semantics and self-adjointness. In *Proceedings MFPS XIII*, Electronic Notes in Theoretical Computer Science. Elsevier, 1997.

A Moggi's semantics of the λ_C-calculus

λ_C-calculus	λ_C-model
$x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i$	$A_1 \times \cdots \times A_n \xrightarrow{\pi_i} A_i \xrightarrow{\eta} TA_i$
$\dfrac{\Gamma \vdash M : A}{\Gamma \vdash [M] : TA}$	$\Gamma \xrightarrow{f} TA$ $\Gamma \xrightarrow{f} TA \xrightarrow{\eta} TTA$
$\dfrac{\Gamma \vdash M : TA}{\Gamma \vdash \mu(M) : A}$	$\Gamma \xrightarrow{f} TTA$ $\Gamma \xrightarrow{f} TTA \xrightarrow{\mu} TA$
$\dfrac{\Gamma \vdash M : A \quad \Gamma, x : A \vdash N : B}{\Gamma \vdash let\, x = M\, in\, N : B}$	$\Gamma \xrightarrow{f} TA$ $\Gamma \times A \xrightarrow{g} TB$ $\Gamma \xrightarrow{\langle id, f\rangle} \Gamma \times TA \xrightarrow{t} T(\Gamma \times A) \xrightarrow{Tg} TTB \xrightarrow{\mu} TB$
$\dfrac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A * B}$	$\Gamma \xrightarrow{f} TA$ $\Gamma \xrightarrow{g} TB$ $\Gamma \xrightarrow{\langle f, g\rangle} TA \times TB \xrightarrow{\psi} T(A \times B)$
$\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \Rightarrow B}$	$\Gamma \times A \xrightarrow{f} TB$ $\Gamma \xrightarrow{\lambda f} TB^A \xrightarrow{\eta} T((TB)^A)$
$\dfrac{\Gamma \vdash M : A \Rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$	$\Gamma \xrightarrow{f} (TB)^A$ $\Gamma \xrightarrow{g} TA$ $\Gamma \xrightarrow{\langle f, g\rangle} (TB)^A \times TA \xrightarrow{\psi} T((TB)^A \times A) \xrightarrow{Tev} TTB \xrightarrow{\mu} TB$

B Direct semantics of the λ_C-calculus

λ_C-calculus	direct λ_C-model
$x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i$	$A_1 \cdots A_n \xrightarrow{\pi_i} A_i$
$\dfrac{\Gamma \vdash M : A}{\Gamma \vdash [M] : TA}$	$\Gamma \xrightarrow{f} A$ $\Gamma \xrightarrow{thunk} L\Gamma \xrightarrow{Lf} LA$
$\dfrac{\Gamma \vdash M : TA}{\Gamma \vdash \mu(M) : A}$	$\Gamma \xrightarrow{f} LA$ $\Gamma \xrightarrow{f} LA \xrightarrow{force} A$
$\dfrac{\Gamma \vdash M : A \quad \Gamma, x : A \vdash N : B}{\Gamma \vdash let\, x = M\, in\, N : B}$	$\Gamma \xrightarrow{f} A$ $\Gamma A \xrightarrow{g} B$ $\Gamma \xrightarrow{\delta} \Gamma\Gamma \xrightarrow{\Gamma f} \Gamma A \xrightarrow{g} B$
$\dfrac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash (M, N) : A * B}$	$\Gamma \xrightarrow{f} A$ $\Gamma \xrightarrow{g} B$ $\Gamma \xrightarrow{\delta} \Gamma\Gamma \xrightarrow{f\Gamma} A\Gamma \xrightarrow{Ag} AB$
$\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : A \Rightarrow B}$	$\Gamma A \xrightarrow{f} B$ $\Gamma \xrightarrow{\lambda f} B^A$
$\dfrac{\Gamma \vdash M : A \Rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$	$\Gamma \xrightarrow{f} B^A$ $\Gamma \xrightarrow{g} A$ $\Gamma \xrightarrow{\delta} \Gamma\Gamma \xrightarrow{f\Gamma} B^A\Gamma \xrightarrow{B^A g} B^A A \xrightarrow{apply} B$