Categorical Models of Explicit Substitutions

Neil Ghani, Valeria de Paiva, and Eike Ritter

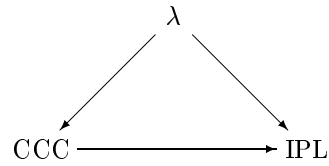
University of Birmingham, England

Abstract. Indexed categories provide models of cartesian calculi of explicit substitutions. However, these structures are inherently non-linear and hence cannot be used to model linear calculi of explicit substitution. This paper replaces indexed categories with pre-sheaves, thus providing a categorical semantics covering both the linear and cartesian cases. We further justify our models by proving soundness and completeness results.

1 Introduction

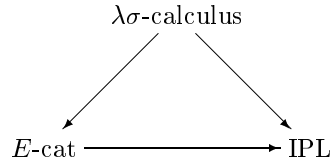
Functional programming languages are based on the λ-calculus and model computation by β-reduction $(\lambda x.t)u \Rightarrow t[u/x]$. This process can duplicate the redexes in u and hence is highly inefficient from the implementational perspective. Abstract machines avoid this problem by reducing terms in an environment — the contraction of a β-redex creates a new substitution which is added to the existing environment and only evaluated when needed. In order to study such machines, *calculi of explicit substitutions* incorporate substitutions directly into the syntax of the λ-calculus rather than treating them as meta-theoretic operations.

Category theory aids the design of abstract machines [7,16] by providing a semantics for explicit substitutions based upon the Curry-Howard triangle relating typed λ-calculi, intuitionistic logic and their categorical models. The best known example relates the simply typed λ-calculus, the positive fragment of intuitionistic propositional logic (IPL) and cartesian closed categories (CCC's).



Indexed categories seem to provide the correct semantic framework for cartesian calculi of explicit substitutions by interpreting substitutions in the base, terms in the fibres and the application of a substitution to a term via re-indexing. Indexed categories also arise in the semantics of dependent types [8] and also in models of the simply-typed λ-calculus where not all objects are exponentiable [12]. Unfortunately indexed categories cannot be used as models of linear calculi of explicit substitution as identities cannot be defined in the fibres. Dropping the identities from our models leads to what we call E-categories and hence our triangle now looks like:

W. Thomas (Ed.): FOSSACS'99, LNCS 1578, pp. 197-211, 1999.



where the $\lambda\sigma$-calculus [1](a simply-typed λ-calculus with explicit substitution) is derived as the internal language of E-categories — this ensures that we have a Curry-Howard triangle. As we shall see, E-categories are essential in generalising the monoidal adjunction semantics of linear logic to cover explicit substitutions.

Now we turn to linear logic. The Curry-Howard correspondence between the linear λ-calculus, the standard categorical models and intuitionistic linear logic is described, for example, in [4]. These categorical models are essentially symmetric monoidal closed categories (SMCCs) with extra structure to model the modality !. However, categorical combinators based on SMCCs ([13] [14]) are not adequate for modelling resource allocations, the idea behind linear functional languages.

Linear categorical abstract machines are designed to implement linear logic and we require linear analogues of the modifications described above. In particular, we want a linear λ-calculus extended with explicit substitutions, a categorical model for the calculus and a Curry-Howard relationship between them. The calculus appears in full in [10] and this paper concentrates on the more refined categorical models for the linear λ-calculus extended with explicit substitutions.

Indexed categories cannot be used as models of linear calculi of explicit substitution as they are an inherently non-linear structure. Asking that the fibres form a category requires identities which in turn corresponds to weakening which is not admissible in linear λ-calculi. Hence we alter the notion of an indexed category to a presheaf (*i.e.*, a functor with **Set** as codomain rather than **Cat**), and call this structure a *linear context-handling category. Cartesian context-handling categories* analogously model cartesian calculi of explicit substitutions.

The tensor, unit and linear implication are then modelled by adding natural isomorphisms to the "fibres" of linear context handling categories — we call these structures L-categories. Modelling contexts by structure in the base and the logical connectives by structure in the fibres distinguishes our models from the usual SMCC's where the same semantic structure is used to model both the behaviour of contexts and the tensor connective. Similarly, intuitionistic implications and conjunctions are modelled by adding structure to the fibres of a cartesian context handling category obtaining the previously mentioned E-categories. The modalities of linear logic are modelled by a monoidal adjunction between (the bases of) an L-category and an E-category.

2 Context Handling Categories

The traditional categorical semantics of explicit substitutions is based on *indexed categories*, ie a *base* category $\mathcal{B}$ and a contravariant functor $E\colon \mathcal{B}^{op} \to \mathbf{Cat}$. The objects of $\mathcal{B}$ model the contexts, the morphisms of $\mathcal{B}$ interpret the explicit substitutions and the fibres interpret the types and terms of the calculus. Unfortunately, indexed categories do not generalise to the linear setting as the

identity on A in a fibre $E(\Gamma)$ corresponds to the non-linear typing judgement $\Gamma, x{:}\, A \vdash x{:}\, A$ [16, 8]. This paper replaces indexed categories with pre-sheaves, thus providing a categorical semantics covering both the linear and cartesian cases. That is, we change the codomain of E from **Cat** to **Set**, thus removing the identities from the fibres. As motivate for our definition, consider the most primitive form of a linear or cartesian calculus of explicit substitutions. Such a calculus has the following components:

- **Types**: A set of types $\mathcal{T}$.
- **Contexts**: Contexts are obtained by "glueing", in a linear or cartesian manner, variable-type pairs $(x{:}\, A)$ together.
- **Substitutions**: Given contexts Γ and Δ, there is a collection of *explicit substitutions* which are judgements of the form $\Gamma \vdash f : \Delta$
- **Terms**: Given a context Γ and type $B \in \mathcal{T}$, there is a collection of *terms* which are judgements $\Gamma \vdash t : B$. Applying a substitution $\Gamma \vdash f : \Delta$ to a term $\Delta \vdash t : A$ results in another term $\Gamma \vdash f * t{:}\, A$.

These properties can be captured by a presheaf $L{:}\, \mathcal{B}^{op} \longrightarrow \mathbf{Set}^{\mathcal{T}}$ with additional structure to capture the formation and behaviour of explicit substitutions:

Definition 1 *Let $\mathcal{B}$ be a (symmetric) monoidal category with distinguished collection of objects $\mathcal{T} \subseteq |\mathcal{B}|$. A* linear context handling category *is a functor $L{:}\, \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ such that for each $A \in \mathcal{T}$ there exists a natural isomorphism*

$$\mathtt{Sub}_A{:}\, L(-)_A \cong \mathsf{Hom}_{\mathcal{B}}(-, A){:}\, \mathtt{Term}_A$$

Each component of Definition 1 corresponds to part of the description of a term assignment system given previously:

- **The Base** $\mathcal{B}$: The base $\mathcal{B}$ of a context handling category models contexts as objects and substitutions as morphisms — types are treated semantically as singleton contexts [1]. That $\mathcal{B}$ forms a category means that substitutions can be sequentially composed and there is an identity substitution. Contexts and substitutions can be put into parallel and there is a an empty context — these features are described by the monoidal structure on $\mathcal{B}$.
- **The Functor** L: The functor L associates to each context Γ and type A a set, written $L(\Gamma)_A$, which we think of as the terms of type A in context Γ. Given a substitution $f : \Gamma \to \Delta$, and any type A, the contravariance of L gives a function $L(f)_A{:}\, L(\Delta)_A \to L(\Gamma)_A$. This re-indexing is exactly what is used to model the application of a substitution to a term.
- **The Natural Transformations** $\mathtt{Sub}_A$ **and** $\mathtt{Term}_A$**:** $\mathtt{Sub}_A$ describes the formation of new substitutions by converting elements in the fibres to morphisms in the base, ie taking a term t and constructing the substitution $\langle t/x \rangle$ [2]. By the Yoneda-Lemma, $\mathtt{Term}_{\mathtt{A}}$ can be replaced by elements $\mathtt{Var}_A \in$

[1] although we simplify notation by sometimes writing A for $x : A$

[2] where x is the variable associated to the singleton context A

$L(A)_A$ and $\mathtt{Term}_{\mathtt{A}}(f)$ is then given by $f * \mathtt{Var}_{\mathtt{A}}$ — thus $\mathtt{Term}_A$ evaluates a substitution at a variable. The condition that $\mathtt{Sub}$ and $\mathtt{Term}$ are natural isomorphisms is then replaced by the equations

$$\mathtt{Sub}_A(t) * \mathtt{Var}_A = t \qquad \mathtt{Sub}_A(\mathtt{Var}_A) = \mathsf{Id}$$

In order to model a calculus of *cartesian* contexts we use *cartesian context handling* categories whose definition only differs in requiring the monoidal structure in the base is actually a product so that weakening and contraction can be interpreted. This notion of a cartesian handling of contexts is implicit in most of the work on categorical modelling of higher-order typed calculi [8] [11].

Definition 2 *Let $\mathcal{B}$ be a cartesian category with distinguished collection of objects $\mathcal{T} \subseteq |\mathcal{B}|$. A* cartesian context handling category *is a functor $E\colon \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ such that for each $A \in \mathcal{T}$ there exists a natural isomorphism*

$$\mathtt{Sub}_A\colon E(-)_A \cong \mathsf{Hom}_{\mathcal{B}}(-, A)\colon \mathtt{Term}_A$$

Notation: We use $\Gamma, \Delta, \cdots$ as generic objects in $\mathcal{B}$, $f, g, \ldots$ as generic morphisms in $\mathcal{B}$ and $A, B, C, \ldots$, as generic elements of $\mathcal{T}$. We write $f*-$ for $E(f)$ (or $L(f)$) for the functor on morphisms. When $\mathcal{B}$ is monoidal the unit is denoted $[\,]$, the tensor product of objects $\Gamma_1, \ldots, \Gamma_n$ is denoted $(\Gamma_1, \ldots, \Gamma_n)$ and similarly the tensor product of two morphisms f and g is written (f, g). In addition, if $\mathcal{B}$ is cartesian, we write Fst and Snd for the two projections.

Should $\mathtt{Sub}$ and $\mathtt{Term}$ be isomorphisms? The equation $\mathtt{Sub}_A(t) * \mathtt{Var}_A = t$ formalises our understanding that $x[t/x] = t$. However, the other equation, namely that if f is a substitution for the variable x, then $f = (f * x)/x$, does not carry the same force, and *intensional* definitions requiring only a retraction $\mathtt{Term}_A \circ \mathtt{Sub}_A = \mathsf{Id}$ could be considered. The situation would be analogous to the intensionality of function spaces: In the same way as two functions are not intensionally equal if they produce the same result when applied to the same arguments, two substitutions are not intensionally equal if applied to the same variable they produce the same result. The formal definition is

Definition 3 *An intensional (cartesian) context category consists of the same data as a (cartesian) context handling category but the natural transformations $\mathtt{Sub}_A$ and $\mathtt{Term}_A$ need only form a retraction $\mathtt{Term}_A \circ \mathtt{Sub}_A = \mathsf{Id}$.*

The next lemma proves that an intensional context handling category where $\mathtt{Sub}_A(\mathtt{Var}_A) = \mathsf{Id}$ is actually extensional. Since we think of $\mathtt{Sub}_A(\mathtt{Var}_A)$ as the substitution $[x/x]$, this is rather a mild condition and dropping it, as intensional structures require, seems counter-intuitive.

Proposition 4 *Let L be an intensional context handling category with types $\mathcal{T}$. If for all $A \in \mathcal{T}$, $\mathtt{Sub}_A(\mathtt{Var}_A) = \mathsf{Id}_A$, then L is a linear context handling category.*

Proof. If $A \in \mathcal{T}$, then we show $\mathtt{Sub}_A \circ \mathtt{Term}_A = \mathsf{Id}$. So let $f : \Gamma \to A$. Then

$$f = f; \mathsf{Id} = f; \mathtt{Sub}_A(\mathtt{Var}_A) = \mathtt{Sub}(f * \mathtt{Var}_A) = \mathtt{Sub} \circ \mathtt{Term}(f)$$

where the first equality holds by definition, the second by assumption, the third is the naturality of $\mathtt{Sub}_A$ and the fourth by definition.

The natural isomorphism between $\mathsf{Hom}(-, A)$ and $E(-)_A$ (or $L(-)_A$ in the linear setting) has several consequences. Firstly, the fibres of a context handling category are determined up to isomorphism by the base of the category. Secondly, all substitutions are *extensional*, ie morphisms are determined by their effects on terms: $f = g$ iff for all terms t, $f * t = g * t$. Finally, models of λ-calculi based on context handling categories can be compared with the standard categorical models by constructing an "internal category" from the fibres. The objects of this category are the elements of $\mathcal{T}$ and the set of morphisms from A to B is the fibre $E(A)_B$. The identity on A is the term $\mathtt{Var}_A$ and the composition of two morphisms $t \in E(A)_B$ and $s \in E(B)_C$ is given by $\mathtt{Sub}(t) * s$. Clearly this internal category is isomorphic to the full subcategory of $\mathcal{B}$ whose objects are $\mathcal{T}$.

3 The Cartesian Model

Context handling categories model the basic features of explicit substitutions, namely the ability to form substitutions from terms, put them in parallel and apply them to terms. This structure is insufficient to model a calculus of explicit substitutions as no mention is made of the connectives. We now consider a canonical extension of the simply typed λ-calculus with explicit substitutions and the extra structure required to model it. Our presentation varies slightly from the original [1], eg we use names rather than De Bruijn numbers.

3.1 The $\lambda\sigma$-calculus

The types of the $\lambda\sigma$-calculus are ground types, the unit type 1, function types $A \to B$ and product types $A \times B$. The raw expressions of $\lambda\sigma$ are:

$$t ::= x \mid \lambda x{:}\, A.t \mid tt \mid \langle t, t\rangle \mid \pi_i t \mid \bullet \mid f * t$$
$$f ::= \langle\rangle \mid \langle f, t/x\rangle \mid f; f$$

where x is a variable. The term $f * t$ represents the application of the explicit substitution f to the term t and $\bullet$ represents the canonical element of the unit type. The substitution $\langle\rangle$ should be thought of as a substitution of variables for themselves, while $\langle f, t/x\rangle$ represents the parallel composition of the substitution f with the substitution of the term t for the variable x. Finally, $f; g$ represents the composition of the substitutions f and g and models iterated substitution.

Contexts are lists $x_1{:}\, A_1, \ldots x_n{:}\, A_n$ where the x's are distinct variables and the A's are types — the domain of the context is $\{x_1, \ldots, x_n\}$ and we write $\Gamma \subseteq \Gamma'$ if the domain of Γ is contained in the domain of Γ'. The $\lambda\sigma$-calculus has term judgements $\Gamma \vdash t : A$ and substitution judgements $\Gamma \vdash f : \Delta$ — these judgements are generated by the inference rules of Table 1. The inference rules for declaring variables and the introduction and elimination rules for function spaces and conjunctions are standard. All the free variables of t are bound in $f * t$ and similarly all the free variables of g are bound in $f; g$. For a full presentation of the meta-theory of $\lambda\sigma$ see [1, 15].

Table 1. Typing Judgements for the $\lambda\sigma$-calculus

– Term Judgements

$$\frac{x{:}A \text{ declared in } \Gamma}{\Gamma \vdash x{:}A} \qquad \frac{\Gamma \vdash f : \Delta \qquad \Delta \vdash t : A}{\Gamma \vdash f * t : A}$$

$$\frac{\Gamma, x{:}A \vdash t{:}B}{\Gamma \vdash \lambda x{:}A.t{:}A \to B} \qquad \frac{\Gamma \vdash t{:}A \to B \qquad \Gamma \vdash u{:}A}{\Gamma \vdash tu{:}B}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \vdash u : B}{\Gamma \vdash \langle t, u\rangle : A \times B} \qquad \frac{\Gamma \vdash t : A_1 \times A_2}{\Gamma \vdash \pi_i(t) : A_i} \qquad \frac{}{\Gamma \vdash \bullet : 1}$$

– Substitution Judgements

$$\frac{\Gamma' \subseteq \Gamma}{\Gamma \vdash \langle\rangle : \Gamma'} \qquad \frac{\Gamma \vdash f{:}\Delta \qquad \Gamma \vdash t{:}A}{\Gamma \vdash \langle f, t/x\rangle : \Delta, x : A} \qquad \frac{\Gamma \vdash f{:}\Delta \qquad \Delta \vdash g{:}\Psi}{\Gamma \vdash f; g : \Psi}$$

where in the second rule x is not in the domain of Δ.

3.2 Modelling the $\lambda\sigma$-calculus in an E-category

Cartesian context handling categories model the behaviour of explicit substitutions, eg their formation from terms and their application to other terms. We now add extra structure to cartesian context handling categories to model the types of the $\lambda\sigma$-calculus. Since these types define new terms, and terms are interpreted in the fibres, this extra structure is defined on the fibres:

Definition 5 *An* E-category *is a cartesian handling category* $E{:}\mathcal{B}^{op}{\to}\mathbf{Set}^{\mathcal{T}}$ *with a distinguished type* $1 \in \mathcal{T}$ *and such that for two types* $A, B \in \mathcal{T}$*, there are types* $A \Rightarrow B, A \times B \in \mathcal{T}$*. In addition* 1 *is terminal in* $\mathcal{B}$*, and there are isomorphisms, natural in* Γ *between* $E((\Gamma, A))_B$ *and* $E(\Gamma)_{A\Rightarrow B}$ *as well as between* $E(\Gamma)_A \times E(\Gamma)_B$ *and* $E(\Gamma)_{A\times B}$*.*

Definition 5 implies that $A \times B$ is isomorphic to the product of A and B in $\mathcal{B}$, namely (A, B) — this is consistent with our philosophy that the semantics of context concatenation and the connective $\times$ are, although related, conceptually distinct. Similarly the type 1 is isomorphic to the empty context []. E-categories also provide a theory of equality judgements for the $\lambda\sigma$-calculus which are of the form $\Gamma \vdash t = t'$ and $\Gamma \vdash f = f'$ — these are given in Table 2 and used in proving soundness and completeness.

Similar structures to our E-categories have been considered in the literature. Jacobs [12] defines a λ1-category as an indexed category $\mathcal{B}^{op}{\to}\mathbf{Cat}$ such that $\mathcal{B}$ has finite products; morphisms in the fibre from A to B are morphisms $\Gamma \times A$ to B in the base category together with the condition that the fibration defined by the indexed category has $\mathcal{T}$-products. Such a λ1-category is an extensional E-category where the fibres are categories and not sets and where the isomorphism between substitutions and terms is the identity.

Table 2. Equality Judgements for the $\lambda\sigma$-calculus

(1) $\langle\rangle ; f = f$	(2) $\langle\rangle * t = t$
(3) $(f;g);h = f;(g;h)$	(4) $(f;g) * t = f * (g * t)$
(5) $\langle f, t/x\rangle * x = t$	(6) $\langle f, t/x\rangle * y = f * y$
(7) $\dfrac{\Gamma \vdash f : \_}{\Gamma \vdash f = \langle\rangle}$	(8) $f;\langle g, t/x\rangle = \langle f;g, (f * t)/x\rangle$
(9) $\dfrac{\Gamma \vdash t : 1}{\Gamma \vdash t = \bullet}$	(10) $\dfrac{\Gamma \vdash \langle\rangle : \Gamma' \qquad \Gamma' = x_1 : X_1, \ldots, x_n : X_n}{\Gamma \vdash \langle\rangle = \langle x_1/x_1, \ldots, x_n/x_n\rangle}$
(11) $t = \lambda x : A.tx$	(12) $(\lambda x : A.t)u = \langle\langle\rangle, u/x\rangle * t$
(13) $f * (tu) = (f * t)(f * u)$	(14) $f * \lambda x : A.t = \lambda y : A.\langle f, y/x\rangle * t$

where in equation 2, $x \notin \mathtt{FV}(t)$

***E*-categories and CCC's** Since the $\lambda\sigma$-calculus contains the λ-calculus, every model of the $\lambda\sigma$-calculus should contain a model of the λ-calculus, ie every E-category should contain an underlying CCC. In addition, one of the key-meta-theoretic properties of the $\lambda\sigma$-calculus is that every $\lambda\sigma$-term is provably equal to a λ-term. The semantic counterpart to this is that every CCC should extend to an E-category. The following theorem makes this relationship clear.

Theorem 6 *(i) Let $E : \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ be an extensional E-category. Then the full subcategory of $\mathcal{B}$ generated by $\mathcal{T}$ is a CCC.*

(ii) Let $\mathcal{C}$ be a CCC and $\mathcal{T}$ be the set of objects of $\mathcal{C}$. Define a functor $E : \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{T}}$ by $E(\_)_A = \mathsf{Hom}_{\mathcal{C}}(\_, A)$, then E is an extensional E-category.

(iii) If $E : \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ is an extensional E-category, then the E-category constructed in (ii) from the CCC constructed in (i) is isomorphic to the restriction of E to the full subcategory of $\mathcal{B}$ generated by $\mathcal{T}$.

Soundness and Completeness We now prove that we can model the $\lambda\sigma$-calculus in any E-category.

Theorem 7 *Let $E : \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ be any E-category. Then there is a canonical interpretation map $[\![\_]\!]$ which assigns to any term of the $\lambda\sigma$-calculus with set $\mathcal{T}$ of base types an element of a fibre and assigns to every substitution a morphism.*

Proof. The types of the λ-calculus are interpreted as elements of $\mathcal{T}$ and, using the product structure of $\mathcal{B}$, this extends to an interpretation of contexts as objects of $\mathcal{B}$. We now define $[\![t]\!]$ by induction over the structure of t:

- Variable are interpreted by $\pi * \mathtt{Var}_A$ where π is a projection in the base
- λ-abstraction, application, product and projections are interpreted via the natural transformations occurring in definition 5. Finally the application of a substitution to a term is modelled by re-indexing.
- The substitution $\langle\rangle$ is interpreted as a projection in $\mathcal{B}$ and $[\![f;g]\!]$ is the composition of $[\![f]\!]$ and $[\![g]\!]$. Finally, $[\![\langle f, t/x\rangle]\!] = \langle [\![f]\!], \mathtt{Sub}([\![t]\!])\rangle$, where the right-hand side uses pairing in $\mathcal{B}$.

That $[\![.]\!]$ respects the equality judgements of the $\lambda\sigma$-calculus relies on proving by induction on t that $[\![t[\boldsymbol{s}/\boldsymbol{x}]]\!] = \langle \mathtt{Sub}([\![s_1]\!]), \ldots, \mathtt{Sub}([\![s_n]\!])\rangle * [\![t]\!]$.

E-categories form a complete class of models for the $\lambda\sigma$-calculus.

Theorem 8 *Let $\Gamma \vdash f : \Gamma'$ and $\Gamma \vdash f' : \Gamma'$ be $\lambda\sigma$-substitution judgements and $[\![\_]\!]$ be the interpretation function of Theorem 7. If for every E-category, $[\![f]\!] = [\![f']\!]$, then $\Gamma \vdash f = f' : \Gamma'$ is provable.*

Proof. The proof is by the standard term-model construction. We sketch this in three stages: (i) first we construct the base; (ii) next we construct the pre-sheaf structure; and (iii) we finally describe the natural transformations `Term` and `Sub`.

- **The Base Category $\mathcal{B}$**: In the term model, the base has as objects contexts and equivalence classes of substitutions as morphisms. The identity will be $\langle\rangle$ while composition is given by ;. Equations (1), (3) and (10) ensure that this structure does indeed define a category.
- **Cartesian Structure on $\mathcal{B}$:** By equation (7), the empty context is terminal in $\mathcal{B}$. On objects, the product structure is context concatenation while pairing is defined via the $\langle\rangle$-combinator. Universality follows from equation (10).
- **The Pre-sheaf:** The functor E maps a context Γ and singleton context $(x : A)$ to the set of equivalence classes of terms of type A in context Γ. On morphisms $E(f)$ maps a term t to the term $f * t$ and E is a functor, ie E preserves identities and composition, by equations (2) and (4)
- **The Natural Transformations `Term` and `Sub`:** The natural transformation `Term` maps $f : \Gamma \to (x : A)$ to $f * x$, while `Sub` maps an element of $E(\Gamma)_{x:A}$, ie a term t, to the substitution $\langle t/x\rangle$. Naturality of `Term` follows from equation (4), while naturality of `Sub` is implicit in equation (8). Equations (5) and (10) imply `Term` and `Sub` are isomorphisms.

4 The Multiplicative Structure

What extra structure must be added to a linear context handling category to model the $(I, \otimes, \multimap)$ connectives from linear logic? Following the definition of E-categories, we may try adding natural isomorphisms to the fibres of linear context handling categories. While this works for the linear function space, there is a complication for the tensor and unit. Recall from section 3 that the structure used to interpret product types is defined on the fibres and then induces an isomorphism in the base between the contexts $z : A \times B$ and $x : A, y : B$. However, imposing structure on the fibres of linear context handling category does not induce such an isomorphism and hence we require one explicitly.

Formally, a L-category requires an object $I \in \mathcal{T}$ to model the unit and binary operations $\otimes$ and $\multimap$ to model the tensor and linear implication. As argued before, I will be isomorphic to the unit $[]$ of the monoidal structure of $\mathcal{B}$. Similarly, $\otimes$ will not be equal to the tensor of $\mathcal{B}$ but will be isomorphic to it.

Definition 9 *An L-category is a linear context-handling category $(\mathcal{B}, \mathcal{T})$ st:*

(i) *There is a type $I \in \mathcal{T}$, and given $A, B \in \mathcal{T}$, there are types $A \otimes B$ and $A \multimap B$.*
(ii) *For every type A and B, there are isomorphisms $n_I : [\,] \cong I : n_I^{-1}$ and $n_\otimes : (A, B) \cong A \otimes B : n_\otimes^{-1}$.*
(iii) *Given types A, B and C, there is a Γ-natural isomorphism between $L((\Gamma, A))_B$ and $L(\Gamma)_{A \multimap B}$.*

Theorem 6 generalises to the linear setting, ie every L-category has an underlying SMCC and every SMCC generates an L-category. This reflects the fact that a linear calculus of explicit substitutions contains an underlying linear λ-calculus and every term is equal to a term of the underlying linear λ-calculus.

Theorem 10 *(i) Let $L : \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ be an L-category. Then the full subcategory of $\mathcal{B}$ generated by $\mathcal{T}$ is a symmetric monoidal closed category.*
(ii) Let $\mathcal{C}$ be any symmetric monoidal closed category with objects $\mathcal{T}$. The functor $L : \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{T}}$ defined by $L(\_)_\Delta = \mathsf{Hom}_{\mathcal{C}}(\_, \Delta)$, is an L-category.
(iii) If $L : \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ is an L-category, then the L-category constructed in (ii) from the underlying SMC defined in (i) is naturally isomorphic to the restriction of L to the full subcategory of $\mathcal{B}$ generated by $\mathcal{T}$.

Proof. The same proof as for Theorem 6 works.

L-categories provide models for a linear λ-calculus with the $(\otimes, I, \multimap)$-type structure extended with explicit substitutions — we call this calculus the monoidal $\lambda\sigma$-calculus. Formally, the raw expressions are

$$t ::= x \mid \lambda x{:}A.t \mid tu \mid t \otimes t \mid \bullet \mid f * t \mid \texttt{let}\ t\ \texttt{be}\ p\ \texttt{in}\ t$$
$$f ::= \langle\rangle \mid \langle f, t/x\rangle \mid f; f \mid \texttt{let}\ t\ \texttt{be}\ p\ \texttt{in}\ f$$

where x is a variable and p is of the form $\bullet, x \otimes y$. The calculus contains the usual terms of the linear λ-calculus, substitution constructs we have already seen and finally there are two new forms of substitution given by `let`-expressions. That is, not only do we have terms of the form $\texttt{let}\ t\ \texttt{be}\ p\ \texttt{in}\ u$ (where u is a term) but also substitutions of the form $\texttt{let}\ t\ \texttt{be}\ p\ \texttt{in}\ f$ (where f is a substitution). These `let`-expressions ensure the context $z : A \otimes B$ is isomorphic to the context $x : A, y : B$ as required by the definition of an L-category. Formally, the typing judgements are of the form $\Gamma \vdash t : A$ and $\Gamma \vdash f : \Gamma'$ and are given in Table 3, while the equality judgements for the calculus are given in Table 4. The η-equations are derived from Ghani's adjoint rewriting [9].

Theorem 11 *Let $L : \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ be a L-category. Then there is an interpretation map $[\![.]\!]$ sending terms of the monoidal $\lambda\sigma$-calculus with ground types $\mathcal{T}$ to elements of the fibres and substitutions to morphisms.*

Proof. The proof is similar to that of Theorem 7. Variables are interpreted by the elements $\mathtt{Var}_A$, while $\langle\rangle$ is interpreted via the identity in the base, parallel composition via the tensor on $\mathcal{B}$ and sequential composition via composition in the base. The isomorphism $A \otimes B \to (A, B)$ is used to interpret both the term

Table 3. Typing Judgements of the Monoidal $\lambda\sigma$-calculus

The term judgements of are

$$\frac{}{x\colon A \vdash x\colon A} \qquad \frac{\Gamma, x\colon A \vdash t\colon B}{\Gamma \vdash \lambda x\colon A.t\colon A \multimap B} \qquad \frac{\Gamma_1 \vdash t\colon A \multimap B \qquad \Gamma_2 \vdash u\colon A}{\Gamma \vdash tu\colon B}$$

$$\frac{\Gamma_1 \vdash f\colon \Gamma_2 \qquad \Gamma_2 \vdash t\colon A}{\Gamma_1 \vdash f * t\colon A} \qquad \frac{}{\_ \vdash \bullet\colon I} \qquad \frac{\Gamma_1 \vdash t\colon I \qquad \Gamma_2 \vdash u\colon A}{\Gamma \vdash \mathtt{let}\ t\ \mathtt{be}\ \bullet\ \mathtt{in}\ u\colon A}$$

$$\frac{\Gamma_1 \vdash t\colon A \quad \Gamma_2 \vdash u\colon B}{\Gamma \vdash t \otimes u\colon A \otimes B} \qquad \frac{\Gamma_1 \vdash u\colon A \otimes B \qquad \Gamma_2, x\colon A, y\colon B \vdash t\colon C}{\Gamma \vdash \mathtt{let}\ u\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ t\colon C}$$

The substitution judgements are

$$\frac{}{\Gamma \vdash \langle\rangle : \Gamma} \qquad \frac{\Gamma_1 \vdash f\colon \Gamma_2 \qquad \Gamma_2 \vdash g\colon \Gamma_3}{\Gamma_1 \vdash f; g\colon \Gamma_3} \qquad \frac{\Gamma_1 \vdash f : \Gamma' \qquad \Gamma_2 \vdash t\colon A}{\Gamma \vdash \langle f, t/x\rangle\colon \Gamma', x\colon A}$$

$$\frac{\Gamma_1 \vdash t\colon I \qquad \Gamma_2 \vdash f\colon \Gamma'}{\Gamma \vdash \mathtt{let}\ t\ \mathtt{be}\ \bullet\ \mathtt{in}\ f\colon \Gamma'} \qquad \frac{\Gamma_1 \vdash u\colon A \otimes B \qquad \Gamma_2, x\colon A, y\colon B \vdash f\colon \Gamma'}{\Gamma \vdash \mathtt{let}\ u\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ f\colon \Gamma'}$$

The rules for substitutions assume x, y are fresh and, where applicable, Γ_1, Γ_2 are disjoint and Γ is any permutation of Γ_1, Γ_2.

$\mathtt{let}\ u\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ t$ and the substitution $\mathtt{let}\ u\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ f$. Similarly the corresponding isomorphism for the unit is used to interpret the other $\mathtt{let}$-expression. The verification that the map $[\![t]\!]$ respects equality judgements relies on a substitution lemma similar to that of Theorem 7.

Theorem 12 *L-categories form a complete class of models for the monoidal $\lambda\sigma$-calculus.*

Proof. A term model is constructed with contexts as objects and equivalence classes of substitutions as morphisms. The pre-sheaf structure is added as in Theorem 8 and, finally, we get an L-category from the inverse morphisms

$$x : X, y : Y \vdash \langle (x \otimes y)/z \rangle : z : A \otimes B$$
$$z : A \otimes B \vdash \mathtt{let}\ z\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ \langle x/x, y/y \rangle : (x\colon X, y\colon Y)$$

Similarly, I is isomorphic to the empty context.

5 The Modalities

The standard categorical model of the modalities of linear logic is via a co-Kleisli construction [17] [6]. Benton [5] proposes the equivalent LNL-categories consisting of a monoidal adjunction between a cartesian closed category (CCC) and a symmetric monoidal closed category (SMCC). The adaptation of this approach to our framework is more succinct and hence used here.

Definition 13 *An* $!L$-category *is an L-category $L\colon \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ together with an E-category $E : \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{S}}$ and monoidal adjunction $F \dashv G : \mathcal{C} \to \mathcal{B}$ such that if $A \in \mathcal{S}$, then $FA \in \mathcal{T}$, and conversely, if $B \in \mathcal{T}$, then $GB \in \mathcal{S}$.*

Table 4. Equality Judgements of the Monoidal $\lambda\sigma$-calculus

Let @ be either ; or $*$ depending whether h is a term or a substitution.

– β- and η-equality:

$$(\lambda x{:}\,A.t)u = \langle u/x\rangle * t \qquad \lambda x : A.tx = x$$
$$\texttt{let } v{\otimes}u \texttt{ be } x{\otimes}y \texttt{ in } h = \langle v/x, u/y\rangle * h \qquad \texttt{let } \bullet \texttt{ be } \bullet \texttt{ in } h = h$$
$$\frac{\Gamma_1 \vdash u : I \qquad \Gamma_2, z : I \vdash f : \Gamma'}{\Gamma \vdash f[u/z] = \texttt{let } u \texttt{ be } \bullet \texttt{ in } \langle \bullet/z\rangle; f} \qquad \frac{\Gamma_1 \vdash u : A{\otimes}B \qquad \Gamma_2, z : A{\otimes}B \vdash f : \Gamma'}{\Gamma \vdash f[u/z] = \texttt{let } u \texttt{ be } x{\otimes}y \texttt{ in } \langle (x{\otimes}y)/z\rangle; f}$$

– Application of Substitutions:

$$\langle\rangle; f = f \qquad \langle\rangle * t = t \qquad (f;g);h = f;(g;h)$$
$$\langle f, t/x\rangle * x = t \qquad \langle f, t/y\rangle * x = f * x \qquad (f;g) * t = f * (g * t)$$
$$\frac{\Gamma \vdash f : -}{\Gamma \vdash f = \langle\rangle} \qquad \frac{\Gamma \vdash \langle\rangle : \Gamma \qquad \Gamma = x_i{:}\,X_i}{\Gamma \vdash \langle\rangle = \langle x_i/x_i\rangle}$$

– If $f = \langle t_1/x_1, \ldots, t_n/x_n\rangle$ then f_t is f restricted to the free variables of t.

$$f; \langle g, t/x\rangle = \langle f_g; g, (f_t * t)/x\rangle$$
$$f * (u \otimes v) = (f_u * u) \otimes (f_v * v) \qquad f * \bullet = \bullet$$
$$f * \lambda y{:}\,A.u = \lambda z{:}\,A.\langle f, z/y\rangle * u \qquad f * uv = (f_u * u)(f_v * v)$$
$$f@\texttt{let } t \texttt{ be } p \texttt{ in } h = \texttt{let } (f_t * t) \texttt{ be } p \texttt{ in } f_h@h$$

As in Theorem 6 LNL-categories embed into $!L$-categories and vice versa.

Theorem 14 *(i) If* $(L{:}\,\mathcal{B}^{op}{\to}\mathbf{Set}^{\mathcal{T}}, E{:}\,\mathcal{C}^{op}{\to}\mathbf{Set}^{\mathcal{S}})$ *is a* $!L$*-category, the full subcategory of* $\mathcal{B}$ *defined by* $\mathcal{T}$ *and of* $\mathcal{C}$ *defined by* $\mathcal{S}$ *is a LNL-category.*

(ii) Let $F \dashv G : \mathcal{C}{\to}\mathcal{B}$ *be a monoidal adjunction between a cartesian closed category* $\mathcal{C}$ *with objects* $\mathcal{S}$ *and a symmetric monoidal closed category* $\mathcal{B}$ *with objects* $\mathcal{T}$*. If we define functors* $E{:}\,\mathcal{C}^{op}{\to}\mathbf{Set}^{\mathcal{S}}$ *by* $E(\_)_\Delta = \mathsf{Hom}_{\mathcal{C}}(\_, \Delta)$ *and* $L{:}\,\mathcal{B}^{op}{\to}\mathbf{Set}^{\mathcal{T}}$ *by* $L(\_)_\Delta = \mathsf{Hom}_{\mathcal{B}}(\_, \Delta)$ *then* (L, E) *is an* $!L$*-category.*

(iii) If $(L{:}\,\mathcal{B}^{op}{\to}\mathbf{Set}^{\mathcal{T}}, E{:}\,\mathcal{C}^{op}{\to}\mathbf{Set}^{\mathcal{S}})$ *be a* $!L$*-category, then the* $!L$*-category constructed in (ii) from the monoidal adjunction constructed in (i) is isomorphic* [3] *to the original* L*-category.*

5.1 xDILL - A Linear Calculus of Explicit Substitutions

We now extend the monoidal $\lambda\sigma$-calculus with !-types and prove that $!L$-categories form a sound and complete class of models for this calculus. Underlying our extended calculus is Barber's DILL [3] — hence we call our calculus xDILL [10]. We use DILL because it incorporates the semantic separation of linear and non-linear contexts directly within the syntax although we could have started from Bierman's linear λ-calculus. Formally, the types of xDILL are base types, unit, function, tensor and !-types and the raw expressions are

$$t ::= x \mid \lambda x{:}\,A.t \mid tu \mid t \otimes t \mid \bullet \mid\ !t \mid f * t \mid \texttt{let } t \texttt{ be } p \texttt{ in } t$$
$$f ::= \langle\rangle \mid \langle f, t/x_I\rangle \mid \langle f, t/x_L\rangle \mid f; f \mid \texttt{let } t \texttt{ be } p \texttt{ in } f$$

[3] in the obvious component-wise sense

where x is a variable and p is of the form $\bullet, x \otimes y$ or $!x$. Like DILL, xDILL contains both linear and intuitionistic variables and hence has zoned contexts of the form $\Gamma|\Delta$. Weakening and contraction are only permitted for variables declared in Γ and the !-type constructor controls the interaction between the intuitionistic and linear zones of a context, thus allowing terms of !-type to be copied and discarded. As in section 4, the `let`-expressions must be generalised to *substitutions as well as terms*. Formally, the typing judgements of xDILL are of the form $\Gamma|\Delta \vdash t : A$ and $\Gamma|\Delta \vdash f : \Gamma'|\Delta'$ and are given in Table 5, while the equality judgements of xDILL as presented in Table 6.

Proposition 15 *There is a canonical interpretation* $[\![\_]\!]$ *of xDILL with base types in* $\mathcal{T}$ *in a* $!L$*-category* $(L\colon \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}, E\colon \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{S}})$ *with monoidal adjunction* $F \dashv G : \mathcal{C} \to \mathcal{B}$.

Proof. Firstly interpret the types in $\mathcal{T}$ — for !-types set $[\![!A]\!] = FG[\![A]\!]$. This gives an interpretation of xDILL contexts using the monoidal structure of $\mathcal{B}$

$$[\![\Gamma|\Delta]\!] = (FG([\![A_1]\!]), \cdots, FG([\![A_n]\!]), [\![B_1]\!], \cdots, [\![B_m]\!])$$

where $\Gamma = x_1\colon A_1, \ldots, x_n\colon A_n$ and $\Delta = y_1\colon B_1, \ldots, y_m\colon B_m$. Now any xDILL term judgement $\Gamma|\Delta \vdash t : A$ is interpreted as an element of $L([\![\Gamma|\Delta]\!])_{[\![A]\!]}$ and any xDILL substitution judgement $\Gamma|\Delta \vdash f : \Gamma'|\Delta'$ is interpreted as a $\mathcal{B}$-map $[\![f]\!] : [\![\Gamma|\Delta]\!] \to [\![\Gamma'|\Delta']\!]$. This map $[\![\_]\!]$ is defined inductively, eg

$$\begin{aligned}
[\![!t]\!] &= \delta_\Gamma * m_\Gamma * FG(\mathtt{Sub}(t)) * \mathtt{Var}_A \\
[\![\mathtt{let}\ t\ \mathtt{be}\ !x\ \mathtt{in}\ u]\!] &= (\mathsf{Id}, \mathtt{Sub}([\![t]\!]), \mathsf{Id}) * [\![u]\!] \\
[\![\mathtt{let}\ t\ \mathtt{be}\ !x\ \mathtt{in}\ f]\!] &= (\mathsf{Id}, \mathtt{Sub}([\![t]\!]), \mathsf{Id}); [\![f]\!]
\end{aligned}$$

where $\delta_\Gamma : (!X_1, \ldots, !X_n) \to (!!X_1, \ldots, !!X_n)$ is derived via the co-multiplication of the comonad on $\mathcal{B}$ and $m_\Gamma : (!!X_1, \ldots, !!X_n) \to !(!X_1, \ldots, !X_n)$ is derived from the monoidal transformation $!X, !Y \to !(X, Y)$.

Completeness of $!L$-categories as models of xDILL is proven by constructing a term model. We only define the structure involved and (mostly) omit the (lengthy, but routine) verification that the structure has the required properties. First the functor $L\colon \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ is defined. The objects of $\mathcal{B}$ are contexts $\Gamma|\Delta$ and morphisms are substitution judgements $\Gamma|\Delta \vdash f : \Gamma'|\Delta'$. Context union makes $\mathcal{B}$ monoidal. Next define $\mathcal{T}$ to be the set of xDILL types, $L(\Gamma|\Delta)_A$ to be the judgements $\Gamma \vdash |\Delta \vdash t : A$, $L(f)(t)$ to be $f * t$, $\mathtt{Var}_A$ to be a canonical variable and set $\mathtt{Sub}(t)$ to be the substitution $\langle t/x \rangle$. This makes L a linear context handling category. Section 4 shows how to turn L into a L-category.

Now we turn to the modalities. The category $\mathcal{C}$ has as objects contexts Γ and morphisms $\mathcal{C}(\Gamma, \Delta)$ are tuples of judgements $\Gamma|\_ \vdash t : A_i$ where Δ is the context $x_1 : A_1, \ldots, x_n : A_n$. Note that in $\mathcal{C}$ there are no `let`-substitutions — this corresponds exactly to the syntactic restrictions on term substitutions that arise in the meta-theory of xDILL [10]. Composition in $\mathcal{C}$ is given by substitution with tuples of variables forming the identities. $\mathcal{S}$ is the set of types and define

Table 5. xDILL Typing Judgements

The term judgements of xDILL are

$$\Gamma, x\colon A, \Gamma' | \_ \vdash x\colon A \qquad \frac{\Gamma_1|\Delta_1 \vdash f\colon \Gamma_2|\Delta_2 \qquad \Gamma_2|\Delta_2 \vdash t\colon A}{\Gamma_1|\Delta_1 \vdash f * t\colon A}$$

$$\Gamma|x\colon A \vdash x\colon A \qquad \frac{\Gamma|\Delta, x\colon A \vdash t\colon B}{\Gamma|\Delta \vdash \lambda x\colon A.t\colon A \multimap B} \qquad \frac{\Gamma|\Delta_1 \vdash t\colon A \multimap B \qquad \Gamma|\Delta_2 \vdash u\colon A}{\Gamma|\Delta \vdash tu\colon B}$$

$$\Gamma|\_ \vdash \bullet\colon I \qquad \frac{\Gamma|\Delta_1 \vdash t\colon I \qquad \Gamma|\Delta_2 \vdash u\colon A}{\Gamma|\Delta \vdash \texttt{let } t \texttt{ be } \bullet \texttt{ in } u\colon A}$$

$$\frac{\Gamma|\Delta_1 \vdash t\colon A \quad \Gamma|\Delta_2 \vdash u\colon B}{\Gamma|\Delta \vdash t \otimes u\colon A \otimes B} \qquad \frac{\Gamma|\Delta_1 \vdash u\colon A \otimes B \qquad \Gamma|\Delta_2, x\colon A, y\colon B \vdash t\colon C}{\Gamma|\Delta \vdash \texttt{let } u \texttt{ be } x \otimes y \texttt{ in } t\colon C}$$

$$\frac{\Gamma|\_ \vdash t\colon A}{\Gamma|\_ \vdash !t\colon !A} \qquad \frac{\Gamma|\Delta_1 \vdash t\colon !A \qquad \Gamma, x\colon A|\Delta_2 \vdash u\colon B}{\Gamma|\Delta \vdash \texttt{let } t \texttt{ be } !x \texttt{ in } u\colon B}$$

The substitution judgements of xDILL are

$$\frac{\Gamma' \subseteq \Gamma}{\Gamma|\Delta \vdash \langle\rangle : \Gamma'|\Delta} \qquad \frac{\Gamma_1|\Delta_1 \vdash f\colon \Gamma_2|\Delta_2 \qquad \Gamma_2|\Delta_2 \vdash g\colon \Gamma_3|\Delta_3}{\Gamma_1|\Delta_1 \vdash f; g\colon \Gamma_3|\Delta_3}$$

$$\frac{\Gamma|\Delta \vdash f : \Gamma'|\Delta' \qquad \Gamma|\_ \vdash t\colon A}{\Gamma|\Delta \vdash \langle f, t/x_I\rangle\colon \Gamma', x\colon A|\Delta'} \qquad \frac{\Gamma|\Delta_1 \vdash f : \Gamma'|\Delta' \qquad \Gamma|\Delta_2 \vdash t\colon A}{\Gamma|\Delta \vdash \langle f, t/x_L\rangle\colon \Gamma'|\Delta', x\colon A}$$

$$\frac{\Gamma|\Delta_1 \vdash t\colon I \qquad \Gamma|\Delta_2 \vdash f\colon \Gamma'|\Delta'}{\Gamma|\Delta \vdash \texttt{let } t \texttt{ be } \bullet \texttt{ in } f\colon \Gamma'|\Delta'} \qquad \frac{\Gamma|\Delta_1 \vdash t\colon !A \qquad \Gamma, x\colon A|\Delta_2 \vdash f\colon \Gamma'|\Delta'}{\Gamma|\Delta \vdash \texttt{let } t \texttt{ be } !x \texttt{ in } f\colon \Gamma'|\Delta'}$$

$$\frac{\Gamma|\Delta_1 \vdash u\colon A \otimes B \qquad \Gamma|\Delta_2, x\colon A, y\colon B \vdash f\colon \Gamma'|\Delta'}{\Gamma|\Delta \vdash \texttt{let } u \texttt{ be } x \otimes y \texttt{ in } f\colon \Gamma'|\Delta'}$$

The rules for substitutions assume x, y are fresh and, where applicable, Δ_1, Δ_2 are disjoint and Δ is a permutation of Δ_1, Δ_2.

$E\colon \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{T}}$ by setting $E(\Gamma)_A$ to be the set of typing judgements $\Gamma|\_ \vdash t : A$. This makes E a cartesian context handling category. E can be made into an E-category using Girard's decomposition of intuitionistic function spaces $A \to B$ into linear function spaces $!A \multimap B$.

Finally we construct a $!L$-category. This is greatly simplified by observing that $\mathcal{B}$ is naturally isomorphic to the full subcategory $\mathcal{B}_0$ whose objects are $\_|x : A$ — again these isomorphisms use the `let`-substitutions of xDILL. Thus we define a monoidal adjunction $F \dashv G : \mathcal{C} \to \mathcal{B}_0$ which then extends to a monoidal adjunction on $\mathcal{B}$. The functor F is given by $F(\Gamma) = \_|z : (!X_1 \otimes \cdots \otimes !X_n)$, where $\Gamma = x_1 : X_1, \ldots, x_n : X_n$ and z is some canonical choice of variable. To define F on morphisms, let $\Gamma|\_ \vdash t_j : Y_j$. Then since there is an isomorphism $\iota^{-1} : F(\Gamma) \to \Gamma$, there are judgements $F(\Gamma) \vdash \iota^{-1}; !t_j : !Y_j$. Hence $F(t_1, \ldots t_n) = \langle(\iota^{-1}; !t_1 \otimes \cdots \otimes \iota^{-1}; !t_m)/x\rangle$. We define G on objects by $G(\_|x : A) = z : A$ — this makes G right-adjoint to F as the required natural isomorphism on sets of derivations follows from the isomorphism in $\mathcal{B}$ between Γ and $F(\Gamma)$. Moreover one can show that we have the required additional data to form a $!L$-category. Hence we have shown the following Theorem:

Theorem 16 *The term model is a $!L$-category.*

Table 6. Equality Judgements for xDILL

Let @ be either ; or $*$ depending whether h is a term or a substitution.

– β- and η-equality:

$$(\lambda x{:}\,A.t)u = \langle u/x\rangle * t \qquad \lambda x : A.tx = x$$

$$\texttt{let } v\otimes u \texttt{ be } x\otimes y \texttt{ in } h = \langle v/x, u/y\rangle * h \qquad \texttt{let } \bullet \texttt{ be } \bullet \texttt{ in } h = h$$

$$\frac{\Gamma|\Delta_1 \vdash u : A\otimes B \qquad \Gamma|\Delta_2, z : A\otimes B \vdash f : \Gamma'|\Delta'}{\Gamma|\Delta \vdash f[u/z] = \texttt{let } u \texttt{ be } x\otimes y \texttt{ in } \langle (x\otimes y)/z\rangle; f} \qquad \texttt{let } !u \texttt{ be } !x \texttt{ in } t = \langle u/x\rangle * t$$

$$\frac{\Gamma|\Delta_1 \vdash u{:}I \qquad \Gamma|\Delta_2, z : I \vdash f : \Gamma'|\Delta'}{\Gamma|\Delta \vdash f[u/z] = \texttt{let } u \texttt{ be } \bullet \texttt{ in } \langle \bullet/z\rangle; f} \qquad \frac{\Gamma|\Delta_1 \vdash u{:}!A \qquad \Gamma|\Delta_2 z :!A \vdash f : \Gamma'|\Delta'}{\Gamma|\Delta \vdash f[u/z] = \texttt{let } u \texttt{ be } !x \texttt{ in } \langle !x/z\rangle; f}$$

– Application of Substitutions:

$$\langle\rangle; f = f \qquad \langle\rangle * t = t \qquad (f;g);h = f;(g;h)$$

$$\langle f, t/x\rangle * x = t \qquad \langle f, t/y\rangle * x = f * x \qquad (f;g) * t = f * (g * t)$$

$$\frac{\Gamma|\Delta \vdash f : -|-}{\Gamma|\Delta \vdash f = \langle\rangle} \qquad \frac{\Gamma|\Delta \vdash \langle\rangle : \Gamma'|\Delta' \qquad \Gamma' = x_i{:}\,X_i \qquad \Delta' = y_i{:}\,Y_j}{\Gamma|\Delta \vdash \langle\rangle = \langle x_i/x_i, y_j/y_j\rangle}$$

– If f is of the form $\langle t_1/x_1, \ldots, t_n/x_n\rangle$ and f_t is f restricted to the free variables of t.

$$f;\langle g, t/x\rangle = \langle f_g; g, (f_t * t)/x\rangle \qquad f*!u = !(f * u)$$

$$f * (u \otimes v) = (f_u * u) \otimes (f_v * v) \qquad f * \bullet = \bullet$$

$$f * \lambda y{:}\,A.u = \lambda z{:}\,A.\langle f, z/y\rangle * u \qquad f * uv = (f_u * u)(f_v * v)$$

$$f@\texttt{let } t \texttt{ be } p \texttt{ in } h = \texttt{let } (f_t * t) \texttt{ be } p \texttt{ in } f_h@h$$

6 Summary and Discussion

We have modularly defined new categorical models for λ-calculi extended with explicit substitutions. We took our intuitions from indexed category theory but had to make alterations so as to accommodate linear calculi in the same framework as cartesian calculi. We have also related these models to the well-established categorical models for their underlying λ-calculi and proved appropriate soundness and completeness results.

Recapitulating from the introduction, the reason for describing these models is our goal of designing an abstract machine based on the linear lambda-calculus that is conceptually clean (and easy to prove correct!). These models have already been used to derive a linear lambda-calculus with explicit substitutions [10] and an abstract machine, which has been implemented by Alberti [2].

However there are two questions which remain unresolved and require further research. Firstly we have been unable to find concrete instances of our proposed models. Secondly, and perhaps more importantly, our definition of a context handling category $L{:}\,\mathcal{B}^{op}\to\mathbf{Set}^{\mathcal{T}}$ distinguishes between isomorphic entities, eg the functor $L(A)$ and the hom-functor $\mathsf{Hom}_{\mathcal{B}}(-, A)$. This goes somewhat against the grain of category theory which tends to regard isomorphic structures as being indistinguishable. However, were we to take the alternative approach of identifying the functor L with the hom-functors and dropping the transformations Sub

and `Term`, while maintaining a Curry-Howard correspondence, this would entail dropping the crucial combinator which forms substitutions from terms from the associated calculus of explicit substitutions. Hence we keep the functor L and the natural isomorphisms `Sub` and `Term` in Definition 1.

We would like to thank Peter Dybjer, Martin Hofmann, Andrea Schalk and Martin Hyland for discussions on the subject of this paper.

References

1. Martin Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jaques Lévy. Explicit substitutions. *Journal of Functional Programming*, 1(4):375–416, 1991.
2. F.J Alberti. An abstract machine based on linear logic and explicit substitutions. Master's thesis, School of Computer Science, University of Birmingham, 1997.
3. A. Barber and G. Plotkin. Dual intuitionistic linear logic. Technical report, LFCS, University of Edinburgh, 1997.
4. N. Benton, G. Bierman, V. de Paiva, and M. Hyland. A term calculus for intuitionistic linear logic. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 75–90. Springer Verlag, 1993.
5. Nick Benton. A mixed linear and non-linear logic: Proofs, terms and models. In *Proceedings of Computer Science Logic '94, Kazimierz, Poland*. Lecture Notes in Computer Science No. 933, Berlin, Heidelberg, New York, 1995.
6. Gavin Bierman. *On Intuitionistic Linear Logic*. Phd-thesis, University of Cambridge, 1994. Also available as Technical Report No. 346.
7. Guy Cousineau, Pierre-Louis Curien, and Michel Mauny. The categorical abstract machine. *Science of Computer Programming*, 8:173–202, 1987.
8. Thomas Ehrhard. A categorical semantics of constructions. In *Third Annual Symposium on Logic in Computer Science*, pages 264–273. IEEE, 1988.
9. N. Ghani. *Adjoint Rewriting*. PhD thesis, University of Edinburgh, 1995.
10. N. Ghani, V. de Paiva, and E. Ritter. Linear explicit substitutions. In *Proc. of Westapp'98*, 1998. Full version submitted for publication.
11. J. M. E. Hyland and A. M. Pitts. The theory of constructions: Categorical semantics and topos theoretic models. *Contemporary Mathematics*, 92:137–198, 1989.
12. Bart Jacobs. Simply typed and untyped lambda calculus revisited. In Michael Fourman, Peter Johnstone, and Andrew Pitts, editors, *Applications of Categories in Computer Science*, LMS Lecture Note Series 177, pages 119–142. CUP, 1992.
13. Yves Lafont. The linear abstract machine. *Theoretical Computer Science*, 59:157–180, 1988.
14. Ian Mackie. Lilac: A functional programming language based on linear logic. *Journal of Functional Programming*, 4(4):395–433, 1994.
15. E. Ritter and V. de Paiva. On explicit substitution and names (extended abstract). In *Proc. of ICALP'97*, LNCS 1256, pages 248–258, 1997.
16. Eike Ritter. Categorical abstract machines for higher-order lambda calculi. *Theoretical Computer Science*, 136(1):125–162, 1994.
17. R. A. G. Seely. Linear logic, *-autonomous categories and cofree algebras. *Contemporary Mathematics*, 92, 1989.