Categorical Models of Explicit Substitutions

Article · August 1998

Source: CiteSeer

CITATIONS
2

READS
308

3 authors, including:

Valeria De Paiva
Topos Institute
203 PUBLICATIONS **2,901** CITATIONS

SEE PROFILE

Eike Ritter
University of Birmingham
75 PUBLICATIONS **1,074** CITATIONS

SEE PROFILE

Categorical Models of Explicit Substitutions

Neil Ghani Valeria de Paiva Eike Ritter*

June 29, 1998

Abstract

This paper concerns itself with the categorical semantics of λ-calculi extended with explicit substitutions. For the simply-typed λ-calculus, indexed categories seem to provide the right categorical framework but because these structures are inherently non-linear, alternate models are needed for linear λ-calculi extended with explicit substitutions.

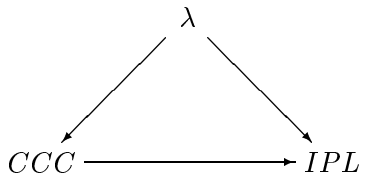
We propose to replace indexed categories by presheaves and obtain a semantics which can be specialised to both the linear and the intuitionistic case. The basic models of a calculi of linear (or intuitionistic) explicit substitutions are called *linear (or cartesian) context-handling categories*. Then we add extra categorical structure to model the connectives of the logic, obtaining L-categories as models of the $(\otimes, I, \multimap)$-fragment of intuitionistic linear logic and E-categories as models of the simply typed λ-calculus. The ! type constructor is then modelled by a monoidal adjunction between an E and L-category. Finally, soundness and completeness of our categorical model is proven.

1 Introduction

This paper is concerned with the mathematical foundations of the implementation of *linear* functional programming languages. The usefulness of a linear functional programming language is described in several papers by Abramsky [Abr93], Wadler [Wad90], and others and will not concern us here. Towards our goal, we want to describe a linear categorical abstract machine [Laf88] [CCM87] along the lines of Ritter's abstract machine for the Calculus of Constructions [Rit94].

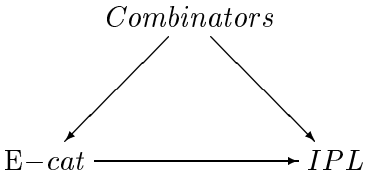
Categorical abstract machines are based on the Curry-Howard triangle, relating typed λ-calculi, intuitionistic logic and their categorical models. The best known example relates the simply typed λ-calculus, the positive fragment of intuitionistic propositional logic (IPL) and cartesian closed categories (CCC's).

*University of Birmingham, School of Computer Science



This picture changes when implementations of the simply typed λ-calculus are considered. In particular, the usual form of β-reduction $(\lambda x.t)u \Rightarrow t[u/x]$ is highly inefficient as those redexes contained in u may be duplicated arbitrarily often in the reduct. Environment machines seek to avoid this problem by reducing terms in an environment so that when a β-redex is contracted, a new substitution is created and added to the existing environment. In order to reason about such machines, we therefore require a calculus within which these environments can be defined and a semantics within which they can be interpreted.

In the case of the simply typed λ-calculus, several calculi of *explicit substitutions* have been proposed while indexed categories seem to provide the correct semantic framework for these calculi. These models interpret substitutions in the base, terms in the fibres and the application of an substitution to a term via re-indexing. As we shall see later, reformulating indexed categories to take into account linearity constraints gives us E-categories. Hence our triangle now looks like:



where the combinators are derived as the internal language of E-categories — this guarantees that we do indeed get a Curry-Howard triangle. E-categories have appeared in the literature, but only implicitly, in Ehrhard's work on D-categories. There is also a slightly different version, Jacob's $\lambda 1$-categories[Jac92], which are not oriented to implementations [1]. Note that we only consider E-categories so that models of linear and cartesian calculi of explicit substitution fall within the same semantic domain and then !-type constructor can be modelled as a monoidal adjunction between such structures.

A similar situation arises in linear logic. The Curry-Howard correspondence between the linear λ-calculus, the standard categorical models and intuitionistic linear logic is described, for example, in [BBdPH93]. These categorical models are essentially symmetric monoidal closed categories (SMCCs) with extra structure to model the modality !. Categorical combinators based on

[1] explain "not oriented towards implementations. Kinoshita-san

SMCCs have been devised ([Laf88] [Mac94]), but, as with their simply typed λ-calculus counterparts, the machines thus obtained are hard to prove correct [2].

Linear categorical abstract machines are designed to implement linear logic and hence our methodology requires 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 a companion paper [GdPR98] and some of the rewriting properties of (a fragment of) the system are described in [NdPR94]. In this paper we concentrate 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 correspond to weakening in the calculus. Hence we weaken 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*. The base of a linear context-handling category can be thought of as a category whose objects are contexts and whose morphisms are substitutions and there is sufficient structure to model a primitive logic of (identity) axioms and cut. We also describe a *cartesian context-handling category*, which as the name indicates, deals with a primitive logic of axioms, cut and the usual structural rules of contraction and weakening.

We then add natural isomorphisms to the "fibres" of linear context handling categories to model the tensor, unit and linear implication — 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, one adds structure to the fibres of a cartesian context handling category to model intuitionistic implications and conjunctions giving what we call an E-category. The exponentials (or modalities) of linear logic are modelled by requiring a monoidal adjunction between (the bases of) an L-category and an E-category. We finish by proving soundness and completeness via a term model construction.

In summary, the definition of context handling categories is motivated by the implementation of λ-calculi where one must model both environments and terms. Category theory provides a semantic structure whose internal language gives a syntax for explicit substitutions which is automatically related to its semantics via a Curry-Howard correspondence. On a more theoretical

[2] why?

level, the semantics of dependent types is usually given in terms of indexed categories and hence our framework is well suited for a generalisation to model linear dependent types. [3]

2 Context Handling Categories

Recall that categorical abstract machines are based upon extending the Curry-Howard correspondence to cover λ-calculi enriched with explicit substitutions. In the cartesian case, the traditional categorical semantics is based on *indexed categories*, ie a *base* category $\mathcal{B}$ and a contravariant functor $E\colon \mathcal{B}^{op} \rightarrow \mathbf{Cat}$. The objects of $\mathcal{B}$ model the contexts of the calculus, 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 any fibre $E(\Gamma)$ corresponds to the non-linear typing judgement $\Gamma, x\colon A \vdash x\colon A$ [Rit94] [Ehr88].

This paper proposes a unified semantic model which specialises to both cartesian and linear calculi of explicit substitutions. Our idea is to retain the functor E but change its codomain from **Cat** to **Set**, thus replacing indexed categories by *presheaves* and hence removing the need for identities in the fibres. To fully motivate our definition, we start by the reducing the idea of a linear or cartesian term assignment system of explicit substitutions to its most primitive form. Such a system will have the following components:

- A set of types $\mathcal{T}$.
- A collection of contexts, which are obtained by "glueing" (in a linear or cartesian manner) variable-type pairs $(x\colon A)$.
- For contexts Γ, Δ there is a collection of *explicit substitutions* which are judgements of the form

  $$\Gamma \vdash f : \Delta$$

  In the linear case, there is in general no substitution in context $x\colon A, y\colon B$ of type $x\colon A$. In the cartesian setting such morphisms can be derived from weakening, while contraction corresponds to a substitution in context $z\colon A$ of type $x\colon A, y\colon A$.
- For each context Γ and type $B \in \mathcal{T}$, there is a collection of terms usually given by judgements of the form

  $$\Gamma \vdash t : B$$

[3] rewrite the dependent types comment

- Given a substitution $\Gamma \vdash f : \Delta$ and a term judgement $\Delta \vdash t : A$, one can apply the substitution to the term to obtain another term judgement $\Gamma \vdash f * t : A$.

This list of 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. The precise definition is:

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*

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

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 [Ehr88] [HP89].

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 $E : \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ such that for each $A \in \mathcal{T}$ there exists a natural isomorphism*

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

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 $L(f)$) when we need 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 $(\Gamma_1, \ldots, \Gamma_n)$ [4] 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.

Intuitively context handling categories associate to each object Γ of $\mathcal{B}$ a $\mathcal{T}$-indexed set $L(\Gamma)_A$ or $E(\Gamma)_A$ which is thought of as consisting of all terms of type A in context Γ, i.e. the typing judgements $\Gamma \vdash t : A$. The monoidal or cartesian structure in the base category models the behaviour of contexts, and the application of a substitution f to t is modelled by $f * t$ — see Figure 1 for an illustration. The natural transformation $\mathtt{Sub}$ takes a term t and forms the substitution $\langle t/x \rangle$ which can then be put in parallel with other substitutions by the tensor or product in the base. By the Yoneda-Lemma, the natural transformation $\mathtt{Term}_{\mathtt{A}}$ can be replaced by elements $\mathtt{Var}_A \in L(A)_A$ or $\mathtt{Var}_A \in E(A)_A$ and the transformation $\mathtt{Term}_{\mathtt{A}}(f)$ is then given by $f * \mathtt{Var}_{\mathtt{A}}$.

[4] we suppress the bracketing and the associated coherence questions

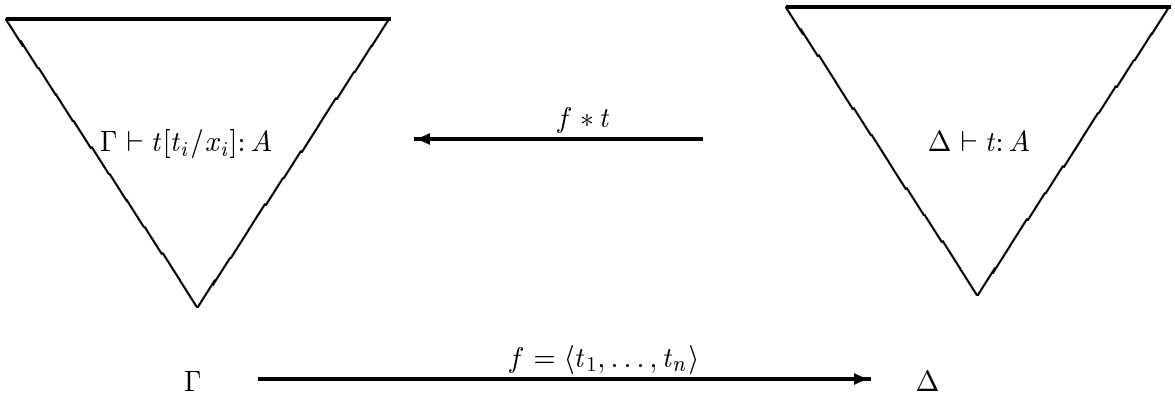


Figure 1: Modelling of explicit substitution

The condition that $\mathtt{Sub}$ and $\mathtt{Term}$ are natural isomorphisms is then replaced by the equations

$$\begin{aligned} \mathtt{Sub}_A(t)^*\mathtt{Var}_A &= t \\ \mathtt{Sub}(\mathtt{Var}) &= \mathsf{Id}. \end{aligned}$$

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 upto isomorphism by the base of the category. Secondly, all substitutions are *extensional*, ie all morphisms are uniquely 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 $\mathcal{T}$.

2.1 Intensional Context Handling Categories

One could imagine a weaker *intensional* definition where one does not ask for a natural isomorphism between $\mathsf{Hom}(-, A)$ and $E(-)_A$ but only for a retraction, ie the existence of two natural transformations $\mathtt{Sub}$ and $\mathtt{Term}$ such that $\mathtt{Term}(\mathtt{Sub}(t)) = t$. As a concrete example of an intensional context handling category, take as a base category $\mathcal{B}$ the free symmetric monoidal (cartesian) category with one generating object and two generating morphisms called Id and f with $f; f = \mathsf{Id}$. Let $E(1) = \{\mathtt{Var}, c_1, c_2\}$ and $E(f)$ to be function which maps c_1 to c_2, c_2 to c_1 and $\mathtt{Var}$ to $\mathtt{Var}$ and $E(1, \ldots, 1) = E(1) \times \cdots E(1)$ with $\mathtt{Var}_{(1,\ldots,1)} = (\mathtt{Var}, \ldots, \mathtt{Var})$. This is an intentional context handling category but $\mathtt{Sub}_1(\mathtt{Var}_1) = f \neq \mathsf{Id}$ and hence extensionality fails. This situation is analogous to non-extensional (weak) products where the product of

two projections is not the identity.

In such an intensional context handling category it is in general impossible to define a category from the fibres as outlined above for extensional context handling categories. The only possible candidate for the identity is the term $\mathtt{Var}$, but in general $\mathtt{Sub}(\mathtt{Var}) * t \neq t$. Hence one cannot compare intensional context handling categories with the standard categorical models of λ-calculi. Any attempt to use the full-subcategory of the base generated by $\mathcal{T}$ also fails as the categorical structure used to interpret the type constructors is defined on the fibres and will not lift to the base in the absence of extensionality. As the only concrete, intensional context handling categories we know of which are not extensional are artificially created (like the one above) we do not consider intensional context handling categories in the sequel. Our goal was to derive a term calculus and a categorical model related by a Curry-Howard correspondence and for this purpose the better behaved extensional context handling categories suffice.

3 The Cartesian Model

Context handling categories model the basic features of explicit substitutions, namely the ability to form explicit substitutions from terms, to put them in parallel with other explicit substitutions and to apply them to terms. Of course this structure is in general insufficient to model a calculus of explicit substitutions as no mention is made of the underlying type structure. In this section we consider a canonical extension of the simply typed λ-calculus with explicit substitutions, called the $\lambda\sigma$-calculus, and consider what extra categorical structure must be added to cartesian context handling categories so as to be able to interpret function spaces and conjunctions.

3.1 The $\lambda\sigma$-calculus

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

$$\begin{array}{rcl} t & ::= & x \mid \lambda x{:}\, A.t \mid tt \mid \langle t, t\rangle \mid \pi_i t \mid f * t \\ f & ::= & \langle\rangle_X \mid \langle f, t/x\rangle \mid f; f \end{array}$$

where x is a variable and X is a set of variables. The term $f * t$ represents the application of the explicit substitution f to the term t, while $\langle\rangle_X$ should be thought of as a substitution of variables for themselves — the suffix X represents those variables discarded. [5] The substitution $\langle f, t/x\rangle$ 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

[5] in this paper can we drop X as we are doing semantics?

substitutions f and g and models iterated substitution.

A context is of the form $x_1 : A_1, \ldots x_n : A_n$ where the x's are distinct variables and the A's are types — the domain of the context is the set of variables $\{x_1, \ldots, x_n\}$. Given a context Γ and a set of variables X, the context $\Gamma \setminus X$ is obtained by removing X from the domain of Γ.

The $\lambda\sigma$-calculus has term judgements of the form $\Gamma \vdash t : A$ and substitution judgements of the form $\Gamma \vdash f : \Delta$ — these judgements are generated by the inference rules of Table 1. The inference rules for declaring variables and the $\rightarrow$-introduction and elimination rules are standard. All the free variables of t are bound in any well typed expression $f * t$ and similarly all the free variables of g are bound in any well formed expression $f; g$. It is not our intention to study the $\lambda\sigma$-calculus in depth here — see [RdP97] for a full presentation of its meta-theory.

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

- Term Judgements

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

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

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

- Substitution Judgements

$$\frac{}{\Gamma \vdash \langle\rangle_X : \Gamma \setminus X} \qquad \frac{\Gamma \vdash f{:}\,\Delta \qquad \Gamma \vdash t{:}\,A}{\Gamma \vdash \langle f, t/x\rangle : \Delta, x : A} \qquad \frac{\Gamma \vdash f{:}\,\Delta \qquad \Delta \vdash g{:}\,\Psi}{\Gamma \vdash f; g : \Psi}$$

  where in the second rule $x \notin \mathtt{dom}\Delta$.

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

Cartesian context handling categories are designed to 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 so as to be able to model the type structure of the $\lambda\sigma$-calculus. Since this type structure allows new terms to be formed, and terms are interpreted in the fibres, the natural approach is to require this extra structure to be defined on the fibres:

Definition 3. *An* E-category *is a cartesian handling category 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

$$\frac{E((\Gamma, A))_B}{\equiv}$$
$$E(\Gamma)_{A \Rightarrow B}$$

$$\frac{E(\Gamma)_A \times E(\Gamma)_B}{\equiv}$$
$$E(\Gamma)_{A \times B}$$

Note that the addition of function spaces for implications is a specialisation of the definition which models dependent products in indexed categories. Since we work with extensional structures, we have $A \times B$ is isomorphic to the product of A and B in $\mathcal{B}$, namely (A, B). One may be tempted to use a definition of cartesian context handling categories which simply requires that $\mathcal{T}$ is closed under $\mathcal{B}$-products. However this would run counter to the philosophy of this paper, namely that the semantics of context concatenation and the connective $\times$ are, although related, conceptually distinct. Definition 3 makes this distinction clear by treating the type $A \times B$ differently from the context (A, B). Similarly we distinguish between the type 1 and the given terminal object of $\mathcal{B}$, namely the empty context $[]$.

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 term rewrites 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 4. *(i) Let* $E: \mathcal{B}^{op} \to \mathbf{Cat}$ *be an extensional* E*-category. Then the full subcategory of* $\mathcal{B}$ *generated by* $\mathcal{T}$ *is cartesian closed.*

(ii) Let $\mathcal{C}$ *be any cartesian closed category and* $\mathcal{T}$ *be the set of objects of* $\mathcal{C}$*. Define a functor* $E: \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{T}}$ *by* $E(\_)_\Delta = \mathsf{Hom}_{\mathcal{C}}(\_, \Delta)$*, 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* T*.*

We now prove that we can model the $\lambda\sigma$-calculus built over ground types T in any E-category with types T.

Proposition 5. *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 of* $\mathcal{B}$*.*

Proof. There is a canonical interpretation of the types of the λ-calculus 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 as follows

- 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 3.
- The substitution $\langle\rangle_X$ 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 on the right-hand side we use pairing in $\mathcal{B}$.

The map $[\![.]\!]$ respects the equality judgements of the λ-calculus relies on a substitution lemmata: one shows by induction over the structure of t that $[\![t[\vec{s}/\vec{x}]]\!] = \langle \mathtt{Sub}([\![s_1]\!]), \ldots, \mathtt{Sub}([\![s_n]\!])\rangle * [\![t]\!]$. This shows the soundness of our categorical model. □

Similar structures to our E-categories have been considered. Jacobs [Jac92] defines a $\lambda 1$-category as an indexed category $\mathcal{B}^{op} \rightarrow \mathbf{Cat}$ 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 $\lambda 1$-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.

4 The Multiplicative Structure

We now turn to the $(I, \otimes, -\circ)$ fragment of intuitionistic linear logic. Linear context handling categories model the way substitutions can be formed and combined via parallel and sequential composition in a linear manner. We now follow the approach of the previous section and consider how one adds structure to linear context handling categories to model the logical connectives of the language. We require $\mathcal{T}$ to contain an object I to model the unit and binary operations $\otimes$ and $-\circ$ to model the tensor and linear implication. As argued before, I will not be equal to the unit [] of the monoidal structure of $\mathcal{B}$ but will be isomorphic. Similarly, $\otimes$ will not be equal to the tensor of $\mathcal{B}$ but will be isomorphic.

Definition 6. *An L-*category *is a linear context-handling category $(\mathcal{B}, \mathcal{T})$ with the following additional properties:*

(i) There is a distinguished type $I \in \mathcal{T}$, and for any two types $A, B \in \mathcal{T}$, there are types $A \otimes B$ and $A - \circ B$ in $\mathcal{T}$.

(ii) For every type A and B, there are isomorphisms

$$n_I : [\,] \cong I : n_I^{-1}$$

$$n_\otimes : (A, B) \cong A \otimes B : n_\otimes^{-1}$$

(iii) Given types A, B and C, there is a Γ-natural isomorphism

$$\frac{L((\Gamma, A))_B}{L(\Gamma)_{A \multimap B}},$$

We denote $n_\otimes * \mathtt{Var}_{A \otimes B}$ by $\otimes$ and $n_I * \mathtt{Var}_I$ by ρ. Theorem 4 generalises to the linear setting, ie every L-category has an underlying SMCC and every SMCC can be extended to L-category. This reflects the fact that a linear calculi of explicit substitutions contains an underlying linear logic and that every term of such a calculus of explicit substitutions is equal to a term of the underlying linear λ-calculus.

Theorem 7. *(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 4 works. □

L-categories provide models for linear lambda calculi with the $(\otimes, I, \multimap)$-type structure extended with explicit substitutions. This calculus has raw terms and substitutions given by

$$\begin{array}{rcl} t & ::= & x \mid f * t \mid \lambda x{:}A.t \mid tt \mid t \otimes t \mid \mathtt{let}\ t\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ t \\ & & \bullet \mid \mathtt{let}\ t\ \mathtt{be}\ \bullet\ \mathtt{in}\ t \\ f & ::= & \langle\rangle \mid \langle f, t/x \rangle \mid \mathtt{let}\ t\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ f \mid \mathtt{let}\ t\ \mathtt{be}\ \bullet\ \mathtt{in}\ f \end{array}$$

and has term judgements of the form $\Gamma \vdash t : A$ and substitution judgements of the form $\Gamma \vdash f : \Delta$ — these judgements are generated by the inference rules of Table 4. Firstly there are the usual terms of the linear λ-calculus, secondly the substitution constructs we have already seen. Finally there are two new forms of substitution given by `let`-expressions. These `let`-expressions are required to ensure the context $z : A \otimes B$ is isomorphic to the context $x : A, y : B$.

We now prove that L-categories form a sound model.

Table 2: The Linear λ-calculus without Exponentials

- Term Judgements

$$\frac{}{x{:}\,A \vdash x{:}\,A} \qquad \frac{\Gamma \vdash f : \Delta \qquad \Delta \vdash t : A}{\Gamma \vdash f * t : A}$$

$$\frac{\Gamma, x{:}\,A \vdash t{:}\,B}{\Gamma \vdash \lambda x{:}\,A.t{:}\,A \multimap B} \qquad \frac{\Gamma_1 \vdash t{:}\,A \multimap B \qquad \Gamma_2 \vdash s{:}\,A}{\Gamma \vdash ts{:}\,B}$$

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

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

- Substitution Judgements

$$\frac{}{\Gamma \vdash \langle\rangle : \Gamma} \qquad \frac{\Gamma_1 \vdash f{:}\,\Delta \qquad \Gamma_2 \vdash t{:}\,A}{\Gamma \vdash \langle f, t/x\rangle : \Delta, x : A}$$

$$\frac{\Gamma \vdash f{:}\,\Delta \qquad \Delta \vdash g{:}\,\Psi}{\Gamma \vdash f; g : \Psi} \qquad \frac{\Gamma_1 \vdash t{:}\,I \qquad \Gamma_2 \vdash f{:}\,\Delta}{\Gamma \vdash \mathtt{let}\ t\ \mathtt{be}\ \bullet\ \mathtt{in}\ f{:}\,\Delta}$$

$$\frac{\Gamma_1 \vdash u{:}\,A \otimes B \qquad \Gamma_2, x{:}\,A, y{:}\,B \vdash f{:}\,\Delta}{\Gamma \vdash \mathtt{let}\ u\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ f{:}\,\Delta}$$

where in those rules with multiple premises Γ is a permutation of Γ_1 and Γ_2

Proposition 8. *Let $L{:}\,\mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ be any L-category. Then there is a canonical interpretation map $[\![.]\!]$ sending terms of the linear λ-calculus with explicit substitutions and ground types $\mathcal{T}$ to elements of the fibres and substitutions to morphisms in the base.*

Proof. The proof is similar to that of proposition 5. 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 $\mathtt{let}\ u\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ t$ 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 two `let`-expressions. The verification that the map $[\![t]\!]$ respects equality judgements relies on a substitution lemmata similar to that of proposition 5. □

5 The Exponentials

The standard categorical model of the exponentials of linear logic is via a co-Kleisli construction [See89] [Bie94]. Benton [Ben95] proposes an equivalent construction, namely a monoidal adjunction between a cartesian closed category (CCC) and a symmetric monoidal closed category (SMCC). The CCC models the non-linear aspects of the calculus while the SMCC models the linear aspects. The functor from the SMCC to the CCC is a forgetful functor which treats linear morphisms as arbitrary morphisms, while its adjoint maps every morphism, including the non-linear ones, into a linear morphism of exponential type. The adaptation of this approach to our framework is more succinct than the co-Kleisli construction and hence is followed here. [6]

Definition 9. *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* $GB \in \mathcal{S}$*.*

As usual we wish to be able to move from LNL-categories to $!L$-categories and back so as to reflect the syntactic inclusion of DILL in xDILL and the embedding of xDILL into DILL. In fact, the relationship between extensional $!L$-categories and linear-non-linear categories is as expected:

Theorem 10. *(i) Let* $(L\colon \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}, E\colon \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{S}})$ *be a* $!L$*-category. Then the full subcategory of* $\mathcal{B}$ *generated by* $\mathcal{T}$ *and of* $\mathcal{C}$ *generated by* $\mathcal{S}$ *is a linear-non-linear 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* $E\colon \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{S}}$ *by* $E(\_)_\Delta = \mathsf{Hom}_{\mathcal{C}}(\_, \Delta)$ *and* $L\colon \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$ *by* $L(\_)_\Delta = \mathsf{Hom}_{\mathcal{B}}(\_, \Delta)$ *then* (L, E) *is an* $!L$*-category.*

(iii) If $(L\colon \mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}, E\colon \mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{S}})$ *be a* $!L$*-category, then the* $!L$*-category constructed in* (ii) *from the monoidal adjunction constructed in* (i) *is isomorphic* [7] *to the original* L*-category.*

Proof. The same proof as for Theorem 4 works. □

The interpretation of the linear λ-calculus without exponentials in an L-category can be extended to the interpretation of the full linear λ-calculus with explicit substitutions in a $!L$-category. We use as an underlying linear λ-calculus a calculus developed by Barber called DILL [BP97] because it incorporates the semantic separation of linear and non-linear contexts directly into the syntax — we therefore call our calculus xDILL [GdPR98].

[6] creation of comonoids

[7] in the obvious componentwise sense

Of course, this choice is merely a matter of convenience and the connection between the syntax and its categorical semantics could also be established for Bierman's version of the linear λ-calculus. We present the typing judgements of xDILL in Table 3 but refer the reader to [BP97] for a comprehensive treatment of the calculus. As in the previous section, we start by proving soundness by giving an interpretation of xDILL in a $!L$-category.

xDILL is a strict extension of the calculus of section 4 containing 3 new term constructs — an introduction term, an elimination term and an associated `let`-substitution for the !-type constructor. The other new feature of xDILL is that, like DILL, it contains both linear and intuitionistic variables and hence has zoned contexts of the form $\Gamma|\Delta$. Intuitively 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.

Formally, the types of xDILL are base types, unit, function, tensor and !-types. The raw expressions of xDILL are

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

where x is a variable and p a pattern, ie of the form $\bullet, x \otimes y$ or $!x$. 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 3. xDILL can be interpreted in any $!L$-category:

Proposition 11. *There is a canonical interpretation $[\![\_]\!]$ of xDILL with base types in $\mathcal{T}$ in a $!L$-category $(L{:}\,\mathcal{B}^{op}{\to}\mathbf{Set}^{\mathcal{T}}, E{:}\,\mathcal{C}^{op}{\to}\mathbf{Set}^{\mathcal{S}})$ with monoidal adjunction $F \dashv G : \mathcal{C}{\to}\mathcal{B}$*

Proof. Firstly one can interpret the types in $\mathcal{T}$ — for !-types set $[\![!A]\!] = FG[\![A]\!]$. This then 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 $\Gamma = x_1{:}\,A_1, \ldots, x_n{:}\,A_n$ and $\Delta = y_1{:}\,B_1, \ldots, y_m{:}\,B_m$. 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 by induction over the structure of expressions with most cases being similar to their analogues in section 4. For the new expressions we have

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

where $\epsilon_\Gamma : (!X_1, \ldots, !X_n) \to (!!X_1, \ldots, !!X_n)$ 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 $!X, !Y \to !(X, Y)$.

□

6 The Term Model

In this section we prove completeness of $!L$-categories as models of xDILL by using the standard construction of a term model. It is worth repeating that all known examples of categorical models for intuitionistic linear logic will provide us with a $!L$-category, but these models will not make the distinction between context concatenation and the tensor connective that we are aiming at with $!L$-categories. We proceed in the same sequence as the definition of the categorical model. This is fairly traditional in categorical type theory, for a textbook example see Crole [Cro94]. We only define the structure involved and (mostly) omit the (lengthy, but routine) verification that the structure has the required properties.

We start by defining the functor $L{:}\,\mathcal{B}^{op} \to \mathbf{Set}^{\mathcal{T}}$. The objects of $\mathcal{B}$ are contexts $\Gamma|\Delta$ and the morphisms of $\mathcal{B}$ are given by substitution judgements $\Gamma|\Delta \vdash f : \Gamma'|\Delta'$. Certainly $\mathcal{B}$ is a category and supports a monoidal structure. For $\mathcal{T}$ we take the set of types of xDILL, define $L(\Gamma|\Delta)_A$ to be the set of judgements $\Gamma \vdash |\Delta \vdash t : A$, set $L(f)(t)$ to be $f * t$, choose $\mathtt{Var}_A$ 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.

Now we extend the structure to cover tensor products and linear implication. The isomorphism defining tensor products is given by

$$(\_|x{:}\,A, y{:}\,B) \vdash (x \otimes y/z){:}\,(\_|z{:}\,A \otimes B)$$

and

$$(\_|z{:}\,A \otimes B) \vdash \mathtt{let}\ z\ \mathtt{be}\ x \otimes y\ \mathtt{in}\ (x/x, y/y){:}\,(\_|x{:}\,A, y{:}\,B)$$

Recall that the whole purpose of introducing `let`-substitutions was to guarantee the existence of such isomorphisms in $\mathcal{B}$. The isomorphism for the unit type is modelled similarly. The natural isomorphisms required for linear implication are as usual by λ-abstraction and application.

Now we turn to the exponentials. 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 to term substitutions that arise in the meta-theory of xDILL []. Composition in $\mathcal{C}$ is given by substitution with tuples of variables forming the identities. Again we take $\mathcal{S}$ to e the set of types and define $E{:}\,\mathcal{C}^{op} \to \mathbf{Set}^{\mathcal{T}}$ 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 Girards decomposition of intuitionistic function spaces $A \to B$ into linear function spaces $A \multimap B$.

We now finish by constructing 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 of the form $\_|x : A$ — again these isomorphisms are derived by using the `let`-substitutions of xDILL. Hence we construct a monoidal adjunction $F \dashv G : \mathcal{C} \to \mathcal{B}_0$ which then automatically lifts to a monoidal adjunction using $\mathcal{B}$. 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 12. *The term model is a $!L$-category.*

Completeness follows by the standard argument.

Theorem 13. *The term model is the initial $!L$-category over the set of ground types $\mathcal{T}$ and $\mathcal{S}$ in the sense that for any other $!L$-category over the same set of ground types there exists exactly one structure-preserving functor F from the term model to this category.*

Proof. The standard argument applies because all equalities which were used to show that the term model is a $!L$-category are judgemental equalities. □

7 Conclusions

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 accomodate linear calculi in the same framework as cartesian calculi. We have also shown that these models are appropriate, by relating them to the well-established categorical models for their underlying λ-calculi and found semantic counterparts to key elements of the meta-theory of these calculi of explicit substitution.

Recapitulating from the introduction, the reason for describing these new

models is our goal of designing an abstract machine based on the linear lambda-calculus that is conceptually clean (and easy to prove correct!). This paper corresponds to the first step of this ongoing research project. These models have already been used to derive a linear lambda-calculus with explicit substitutions abstract machine, which has been implemented by Alberti [Alb97].

Acknowlegments

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

References

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

[Alb97] F.J Alberti. An abstract machine based on linear logic and explicit substitutions. Master's thesis, School of Computer Science, University of Birmingham, 1997.

[BBdPH93] 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.

[Ben95] 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.

[Bie94] Gavin Bierman. *On Intuitionistic Linear Logic*. Phd-thesis, University of Cambridge, 1994. Also available as Technical Report No. 346.

[BP97] A. Barber and G. Plotkin. Dual intuitionistic linear logic. Technical report, LFCS, University of Edinburgh, 1997.

[CCM87] Guy Cousineau, Pierre-Louis Curien, and Michel Mauny. The categorical abstract machine. *Science of Computer Programming*, 8:173–202, 1987.

[Cro94] Roy L. Crole. *Categories for Types*. Cambridge University Press, 1994.

[Ehr88] Thomas Ehrhard. A categorical semantics of constructions. In *Third Annual Symposium on Logic in Computer Science*, pages 264–273. IEEE, 1988.

[GdPR98] N. Ghani, V. de Paiva, and E. Ritter. Linear explicit substitutions. In *This reader*. 1998.

[HP89] J. Martin E. Hyland and Andrew M. Pitts. The theory of constructions: Categorical semantics and topos theoretic models. *Contemporary Mathematics*, 92:137–198, 1989.

[Jac92] 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. Cambridge University Press, 1992.

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

[Mac94] Ian Mackie. Lilac: A functional programming language based on linear logic. *Journal of Functional Programming*, 4(4):395–433, 1994.

[NdPR94] Monica Nesi, Valeria de Paiva, and Eike Ritter. Rewriting properties of combinators for intuitionistic linear logic. In *Proceedings of the Workshop on Higher Order Algebra, Logic and Term Rewriting*, pages 256–275. Lecture Notes in Computer Science No. 816, Berlin, Heidelberg, New York, 1994.

[RdP97] E. Ritter and V. de Paiva. On explicit substitution and names (extended abstract). In *Proc. of ICALP'97*, LNCS 1256, pages 248–258, 1997.

[Rit94] Eike Ritter. Categorical abstract machines for higher-order lambda calculi. *Theoretical Computer Science*, 136(1):125–162, 1994.

[See89] R. A. G. Seely. Linear logic, *-autonomous categories and cofree algebras. *Contemporary Mathematics*, 92, 1989.

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

Table 3: xDILL Typing Judgements

The term judgements of xDILL are

$$\Gamma, x\colon A, \Gamma'|\_ \vdash x\colon A \qquad \Gamma|x\colon A \vdash x\colon A$$

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

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

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

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

$$\frac{\Gamma_1|\Delta_1 \vdash f\colon \Gamma_2|\Delta_2 \quad \Gamma_2|\Delta_2 \vdash t\colon A}{\Gamma_1|\Delta_1 \vdash f * t\colon A}$$

The substitution judgements of xDILL are

$$\frac{}{\Gamma|\Delta \vdash \langle\rangle_X : \Gamma \setminus X|\Delta}$$

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

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

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

$$\frac{\Gamma|\Delta_1 \vdash t\colon I \quad \Gamma|\Delta_2 \vdash f\colon \Gamma'|\Delta'}{\Gamma|\Delta \vdash \texttt{let}\ t\ \texttt{be}\ \bullet\ \texttt{in}\ f\colon \Gamma'|\Delta'}$$

$$\frac{\Gamma_1|\Delta_1 \vdash f\colon \Gamma_2|\Delta_2 \quad \Gamma_2|\Delta_2 \vdash g\colon \Gamma_3|\Delta_3}{\Gamma_1|\Delta_1 \vdash f; g\colon \Gamma_3|\Delta_3}$$

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.