

Notes on Sconing and Relators

John C. Mitchell* Andre Scedrov†

Abstract

This paper describes a semantics of typed lambda calculi based on relations. The main mathematical tool is a category-theoretic method of sconing, also called glueing or Freyd covers. Its correspondence to logical relations is also examined.

1 Introduction

Many modern programming languages feature rather sophisticated typing mechanisms. In particular, languages such as ML include *polymorphic* data types, which allow considerable programming flexibility. Several notions of polymorphism were introduced into computer science by Strachey [Str67], among them the important notion of *parametric* polymorphism. Strachey's intuitive definition is that a polymorphic function is parametric if it has a uniformly given algorithm in all types, that is, if the function's behavior is independent of the type at which the function is instantiated. Reynolds [Rey83] proposed a mathematical definition of parametric polymorphic functions by means of invariance with respect to certain relations induced by types. Unfortunately, the mathematical tools used in [Rey83] were limited to a standard set-theoretic framework, which Reynolds later showed to be vacuous in the case of so-called second order polymorphic lambda calculus (see Pitts [Pit87]).

We describe a general mathematical framework for semantics of type disciplines based on relations. This framework, which is also proposed independently by Ma and Reynolds [MaR92], may be used to give a mathematically correct notion of relational invariance proposed in [Rey83] and a rigorous framework for the considerations in Wadler [Wad89]. The relational semantic framework we describe is also useful in investigating other aspects of programming languages. Abramsky and Jensen [AbJ91] use it for strictness analysis and O'Hearn and Tennent [OHT93] use it in studying semantics of local variables.

The main mathematical tool used in this work is the category-theoretic method of *sconing* described in Freyd and Scedrov [FrS90] and also called glueing or Freyd covers, see Lambek and Scott [LS86]. To our knowledge, the first application of this method to type disciplines is given in Appendix C of Lafont [Laf88]. In the case of simple types, this method corresponds closely to so-called *logical relations*, described for instance in Plotkin [Plo80], Statman [Sta85], and Mitchell [Mit90]. This correspondence is examined in detail. In the case of polymorphic types, a central role is played by *relators*, *i.e.*, maps that take objects to objects and relations to relations. Sconing may be used to express relators as functors.

*jcm@cs.stanford.edu Department of Computer Science, Stanford University, Stanford, CA 94305. Supported in part by an NSF PYI Award, matching funds from Digital Equipment Corporation, the Powell Foundation, and Xerox Corporation; and the Wallace F. and Lucille M. Davis Faculty Scholarship.

†andre@cis.upenn.edu Department of Mathematics, University of Pennsylvania, Philadelphia, PA 19104-6395. Scedrov is an American Mathematical Society Centennial Research Fellow. He is partially supported by NSF Grant CCR-91-02753 and by ONR Grants N00014-88-K-0635 and N00014-92-J-1916.

In order to make the methods and arguments accessible to readers with minimal background in category theory, we choose a much more relaxed style of presentation than would be usual in a research paper. In this vein, here we discuss only the basic framework for simple types and for implicit polymorphism. The extension to explicit polymorphism should be routine to readers thoroughly familiar with [Pit87, See87, Gir86, Wad89]. (Note to specialists: the extension to second order polymorphism differs significantly from second order logical relations proposed by Mitchell and Meyer [MM85]; we shall take this up elsewhere.)

It is a pleasure to acknowledge the collaboration with Samson Abramsky and Philip Wadler in the initial stages of this work. We would like to thank John Reynolds for many inspiring discussions of polymorphism.

2 Simply typed lambda calculus

2.1 Types and terms

We will consider typed lambda calculus with terminator (terminal object) $\mathbf{1}$, product and function types, and any set S of type constants b_1, b_2, \dots . We will use the symbol “ \Rightarrow ” in function types, and reserve the single arrow “ \rightarrow ” for morphisms in a category (including functions between sets). The well-formed terms over any set of types are defined using the subsidiary notion of type assignment. A *type assignment* Γ is a finite set of formulas $x:\sigma$ associating types to variables, with no variable x occurring twice. We write $\Gamma, x:\sigma$ for the type assignment

$$\Gamma, x:\sigma = \Gamma \cup \{x:\sigma\},$$

where, in writing this, we assume that x does not appear in Γ . Terms will be written in the form $\Gamma \triangleright M:\sigma$, which may be read, “ M has type σ relative to Γ .” The well-formed terms are given by the following rules, where we assume $*$ is a constant symbol of type $\mathbf{1}$. For simplicity, we will assume that this is the only constant symbol.

$$\begin{array}{l}
(\text{var}) \qquad \qquad \qquad x:\sigma \triangleright x:\sigma \\
(\text{cst}) \qquad \qquad \qquad \emptyset \triangleright *:\mathbf{1} \\
(\Rightarrow E) \qquad \qquad \frac{\Gamma \triangleright M:\sigma \Rightarrow \tau, \Gamma \triangleright N:\tau}{\Gamma \triangleright MN:\tau} \\
(\Rightarrow I) \qquad \qquad \frac{\Gamma, x:\sigma \triangleright M:\tau}{\Gamma \triangleright \lambda x:\sigma. M:\sigma \Rightarrow \tau} \\
(\times E) \qquad \qquad \frac{\Gamma \triangleright M:\sigma \times \tau}{\Gamma \triangleright \mathbf{Proj}_1^{\sigma,\tau} M:\sigma, \Gamma \triangleright \mathbf{Proj}_2^{\sigma,\tau} M:\tau} \\
(\times I) \qquad \qquad \frac{\Gamma \triangleright M:\sigma, \Gamma \triangleright N:\tau}{\Gamma \triangleright \langle M, N \rangle:\sigma \times \tau} \\
(\text{add hyp}) \qquad \frac{\Gamma \triangleright M:\tau}{\Gamma, x:\sigma \triangleright M:\tau}
\end{array}$$

2.2 Equations

Given our formulation of terms, it is natural to write equations in the form

$$\Gamma \triangleright M = N : \tau$$

where we assume that $\Gamma \triangleright M : \tau$ and $\Gamma \triangleright N : \tau$. The equational axioms and inference rules are as follows, where $[N/x]M$ denotes substitution of N for x in M .

$$\begin{array}{ll}
 (\text{one}) & \Gamma \triangleright x = * : \mathbf{1} \\
 (\mathbf{Proj}_1) & \Gamma \triangleright \mathbf{Proj}_1(M, N) = M : \sigma \\
 (\mathbf{Proj}_2) & \Gamma \triangleright \mathbf{Proj}_2(M, N) = N : \tau \\
 (\mathbf{Pair}) & \Gamma \triangleright (\mathbf{Proj}_1 M, \mathbf{Proj}_2 M) = M : \sigma \times \tau \\
 (\alpha) & \Gamma \triangleright \lambda x : \sigma. M = \lambda y : \sigma. [y/x]M : \sigma \Rightarrow \tau, \text{ provided } y \notin FV(M) \\
 (\beta) & \Gamma \triangleright (\lambda x : \sigma. M)N = [N/x]M : \tau \\
 (\eta) & \Gamma \triangleright \lambda x : \sigma. (Mx) = M : \sigma \Rightarrow \tau, \text{ provided } x \notin FV(M) \\
 (\text{ref}) & \Gamma \triangleright M = M : \sigma \\
 (\text{sym}) & \frac{\Gamma \triangleright M = N : \sigma}{\Gamma \triangleright N = M : \sigma} \\
 (\text{trans}) & \frac{\Gamma \triangleright M = N : \sigma, \Gamma \triangleright N = P : \sigma}{\Gamma \triangleright M = P : \sigma} \\
 (\xi) & \frac{\Gamma, x : \sigma \triangleright M = N : \tau}{\Gamma \triangleright \lambda x : \sigma. M = \lambda x : \sigma. N : \sigma \Rightarrow \tau} \\
 (\mu) & \frac{\Gamma \triangleright M_1 = M_2 : \sigma \Rightarrow \tau, \Gamma \triangleright N_1 = N_2 : \sigma}{\Gamma \triangleright M_1 N_1 = M_2 N_2 : \tau} \\
 (\text{add hyp}) & \frac{\Gamma \triangleright M = N : \sigma}{\Gamma, x : \tau \triangleright M = N : \sigma}
 \end{array}$$

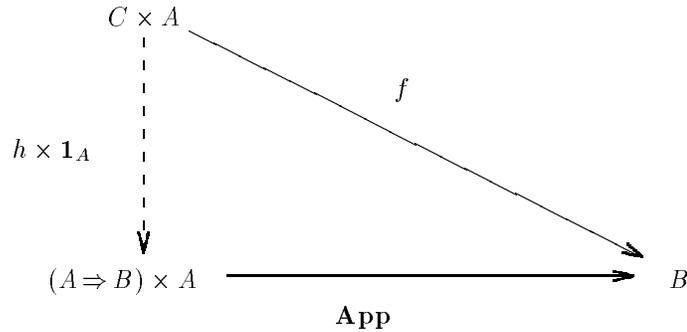
3 Cartesian closed categories

A *cartesian closed category* (or *ccc*) is a category with specified terminal object, products and exponentials. This means that a category \mathbf{C} is cartesian closed only if the following additional data are provided and regarded as part of the structure:

- An object $\mathbf{1}$ with unique arrow $\mathcal{O}^A : A \rightarrow \mathbf{1}$ for each object A ,
- A binary object map \times with, for any objects A, B and C , specified arrows $\mathbf{Proj}_1^{A,B}, \mathbf{Proj}_2^{A,B}$, and map $\langle \cdot, \cdot \rangle_{C,A,B}$ on arrows such that for every $f : C \rightarrow A$ and $g : C \rightarrow B$, the arrow $\langle f, g \rangle_{C,A,B} : C \rightarrow A \times B$ is the unique h satisfying

$$\begin{array}{ccccc}
 & & C & & \\
 & f \swarrow & \vdots & \searrow g & \\
 & & h & & \\
 & & \Downarrow & & \\
 & \swarrow & A \times B & \searrow & \\
 \mathbf{Proj}_1 \swarrow & & & & \searrow \mathbf{Proj}_2 \\
 A & & & & B
 \end{array}$$

- A binary object map \Rightarrow with, for any objects A, B and C , a specified arrow $\mathbf{App}^{A,B}$ and map $\mathbf{Curry}^{A,B,C}$ on arrows such that for every $f: C \times A \rightarrow B$, the arrow $\mathbf{Curry}^{A,B,C}(f)$ is the unique h satisfying

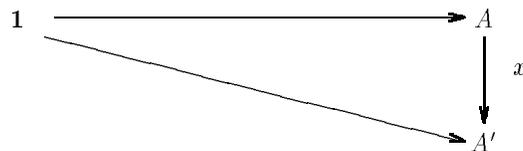


A category \mathbf{C} is *well-pointed* if, for any $f, g: A \rightarrow B$, we have $f = g$ iff for every $x: \mathbf{1} \rightarrow A$, we have $f \circ x = g \circ x$. Another phrase for “well-pointed” is *generated by $\mathbf{1}$* .

Functors that preserve cartesian closed structure will play an important role in our discussion. A *representation of cartesian closed categories* (or simply *ccc-representation*) is a functor $\mathbf{F}: \mathbf{C} \rightarrow \mathbf{D}$ from one cartesian closed category to another that preserves terminator, products and exponentials (function spaces). In the literature, ccc-representations are sometimes called cartesian closed functors, or cc-functors.

4 Scoring

Let \mathbf{C} be a category with a terminator (terminal object) $\mathbf{1}$. We will define a functor $|\cdot|$ from \mathbf{C} to \mathbf{Set} , the category of sets and functions, which associates to each object its *set of global elements*. For any object A in \mathbf{C} , let $|A|$ be the set of all morphisms $\mathbf{1} \rightarrow A$ in \mathbf{C} . For any morphism $x: A \rightarrow A'$ in \mathbf{C} let $|x|: |A| \rightarrow |A'|$ be the function defined by composition, according to the following diagram.



It is not hard to see that the functor $|\cdot|$ from \mathbf{C} to \mathbf{Set} preserves products up to isomorphism. (The functor also preserves equalizers up to isomorphism, but we shall not use this fact.)

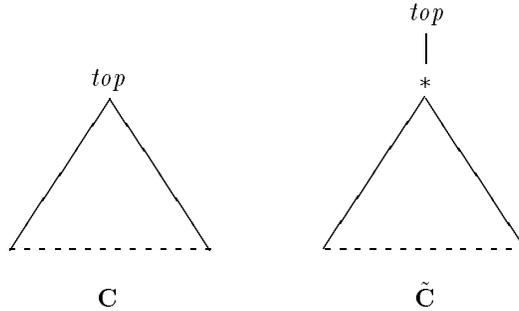
The category $\hat{\mathbf{C}}$, called the *scone of \mathbf{C}* , is defined as follows. The objects of $\hat{\mathbf{C}}$ are triples $\langle S, f, A \rangle$, where S is a set, A is an object of \mathbf{C} , and $f: S \rightarrow |A|$ is a function (in \mathbf{Set}). The morphisms $\langle S, f, A \rangle \rightarrow \langle S', f', A' \rangle$ in $\hat{\mathbf{C}}$ are pairs $\langle t, x \rangle$, where $t: S \rightarrow S'$ is a function and $x: A \rightarrow A'$ is a morphism in \mathbf{C} such that:

$$\begin{array}{ccc}
S & \xrightarrow{t} & S' \\
f \downarrow & & \downarrow f' \\
|A| & \xrightarrow{|x|} & |A'|
\end{array}$$

The reader may easily verify that $\langle |1|, id, \mathbf{1} \rangle$ is a terminator in $\hat{\mathbf{C}}$. There is a canonical “forgetful” functor $\pi: \hat{\mathbf{C}} \rightarrow \mathbf{C}$ which takes $\langle S, f, A \rangle$ to A and $\langle t, x \rangle$ to x . Category theorists will recognize the scone as a special case of a *comma category*, namely $\hat{\mathbf{C}} = (id \downarrow | \cdot |)$, where id is the identity functor on Set ; see, *e.g.*, [McL71]. It is common to overload notation and use the symbol Set for the identity functor from sets to sets (and similarly for other categories). With this convention, we may write $\hat{\mathbf{C}} = (Set \downarrow | \cdot |)$.

In order to make a connection with logical predicates and logical relations described *e.g.*, in [Mit90], we will be interested in a subcategory $\tilde{\mathbf{C}}$ of the scone which we call the *subcone* of \mathbf{C} . The objects of the subcone $\tilde{\mathbf{C}}$ are the triples $\langle S, f, A \rangle$ with $f: S \hookrightarrow |A|$ an inclusion of sets. The morphisms $\langle S, f, A \rangle \rightarrow \langle S', f', A' \rangle$ of $\tilde{\mathbf{C}}$ are the same as for $\hat{\mathbf{C}}$, so $\tilde{\mathbf{C}}$ is a *full* subcategory of $\hat{\mathbf{C}}$. Notice, however, that if $\langle t, x \rangle$ is a morphism in $\tilde{\mathbf{C}}$, then the set function $t: S \rightarrow S'$ is uniquely determined by x and S . Specifically, t is the restriction of $|x|$ to S .

An illustrative first example is the subcone of a partially ordered set \mathbf{C} with a top element top . In this case, $\tilde{\mathbf{C}}$ is again partially ordered, with a “new” element (indicated by $*$ below) just below top .



A remarkable feature of *sconing* in general is that it preserves almost any additional categorical structure that \mathbf{C} might have. For further discussion, see [FrS90], for example. The sconing construction is also discussed in [LS86] and [ScS82], where $\hat{\mathbf{C}}$ is called the *Freyd cover* of \mathbf{C} . We will be concerned with sconing and cartesian closed structure.

Proposition 4.1 *If \mathbf{C} is a cartesian closed category, then $\hat{\mathbf{C}}$ is cartesian closed and the canonical functor $\hat{\mathbf{C}} \rightarrow \mathbf{C}$ is a representation of cartesian closed categories.*

Proof. Let $X = \langle S, f, A \rangle$ and $Y = \langle S', f', A' \rangle$ be objects of $\hat{\mathbf{C}}$. Then the object

$$X \times Y = \langle S \times S', f \times f', A \times A' \rangle,$$

is a product of X and Y , where $(f \times f')\langle s, s' \rangle = \langle f(s), f'(s') \rangle_{1, A, A'}$. An exponential of X and Y (an object of functions from X to Y) is given by

$$X \Rightarrow Y = \langle M, h, A \Rightarrow A' \rangle,$$

where M is the set of morphisms $X \rightarrow Y$ in $\widehat{\mathbf{C}}$ and $h(\langle t, x \rangle): \mathbf{1} \rightarrow A \Rightarrow A'$ is the morphism in \mathbf{C} obtained from $x: A \rightarrow A'$ by currying. \blacksquare

The subcategory $\widehat{\mathbf{C}}$ inherits (an isomorphic copy of) the cartesian closed structure of $\widehat{\mathbf{C}}$. More precisely, $\widehat{\mathbf{C}}$ is isomorphic to a full sub-cartesian-closed category of $\widehat{\mathbf{C}}$. The reader may enjoy constructing exponentials in the subscone. In Section 4.1, we will work out the special case where \mathbf{C} is a product of two ccc's.

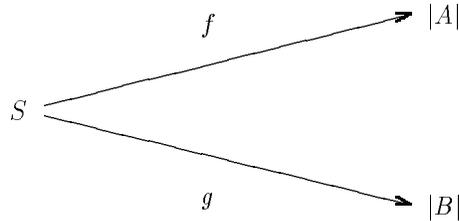
It is category-theoretic folklore that Proposition 4.1 (and its proof) extend to a more general setting with Set replaced by any cartesian closed category \mathbf{C}' with equalizers and the functor $|\cdot|: \mathbf{C} \rightarrow Set$ replaced by any functor $\mathbf{F}: \mathbf{C} \rightarrow \mathbf{C}'$ that preserves products up to isomorphism. This generalization takes us from the specific comma category $\widehat{\mathbf{C}} = (Set \downarrow |\cdot|)$ to a more general case of the form $(\mathbf{C}' \downarrow \mathbf{F})$, see [Laf88, MaR92].

Proposition 4.2 *Let \mathbf{C} and \mathbf{C}' be cartesian closed categories and assume \mathbf{C}' has equalizers. Let $\mathbf{F}: \mathbf{C} \rightarrow \mathbf{C}'$ be a functor that preserves finite products up to isomorphism. Then the comma category $(\mathbf{C}' \downarrow \mathbf{F})$ is a cartesian closed category and the canonical functor $(\mathbf{C}' \downarrow \mathbf{F}) \rightarrow \mathbf{C}$ is a representation of cartesian closed categories.*

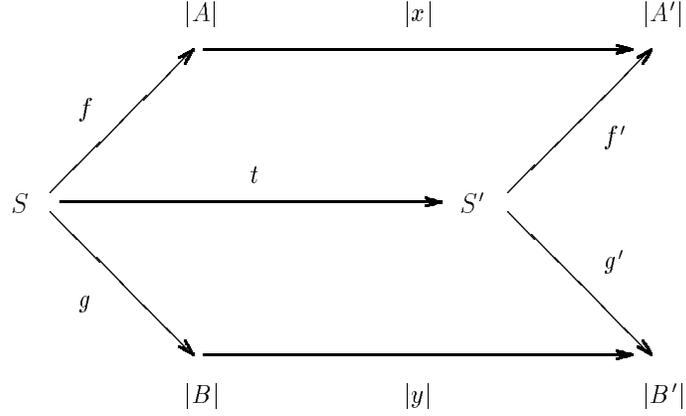
4.1 A special case: sconing with relations

An illuminating case is the scone of a product category. Let us consider a product category $\mathbf{C} = \mathbf{A} \times \mathbf{B}$, where both categories \mathbf{A} and \mathbf{B} have a terminator. We remind the reader that the objects of $\mathbf{A} \times \mathbf{B}$ are ordered pairs of objects from \mathbf{A} and \mathbf{B} and the morphisms $\langle A, B \rangle \rightarrow \langle A', B' \rangle$ are pairs $\langle x, y \rangle$, where $x: A \rightarrow A'$ is a morphism in \mathbf{A} and $y: B \rightarrow B'$ is a morphism in \mathbf{B} . When \mathbf{A} and \mathbf{B} are cartesian closed, so is $\mathbf{A} \times \mathbf{B}$, with coordinatewise cartesian closed structure. A terminator of $\mathbf{A} \times \mathbf{B}$ is $\langle \mathbf{1}, \mathbf{1} \rangle$. It is easy to see that $|\langle A, B \rangle|_{\mathbf{C}} = |A|_{\mathbf{A}} \times |B|_{\mathbf{B}}$, where the cartesian product of $|A|_{\mathbf{A}}$ and $|B|_{\mathbf{B}}$ is taken in sets. We will often omit the subscripts from various $|\cdot|$'s, since these are generally clear from context. We will focus on the scone of product categories for the rest of this section.

The objects of the scone $\widehat{\mathbf{A} \times \mathbf{B}}$ may be described as tuples $\langle S, f, g, A, B \rangle$, with S a set, A an object in \mathbf{A} , B an object in \mathbf{B} , and f and g functions that form a so-called *span*:

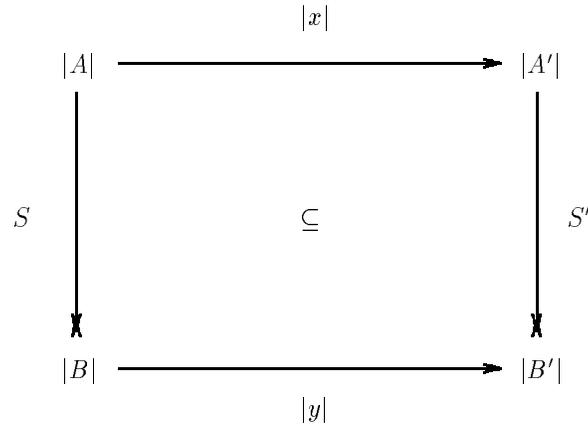


The morphisms $\langle S, f, g, A, B \rangle \rightarrow \langle S', f', g', A', B' \rangle$ in $\widehat{\mathbf{A} \times \mathbf{B}}$ may be described as tuples $\langle t, x, y \rangle$, with $x: A \rightarrow A'$ a morphism in \mathbf{A} , $y: B \rightarrow B'$ a morphism in \mathbf{B} , and $t: S \rightarrow S'$ a function such that:



Morphisms in the scene of a product category resemble *transformations of structors*, described in [Fre93].

Let us now concentrate on the subcategory $\widetilde{\mathbf{A} \times \mathbf{B}}$ of $\widehat{\mathbf{A} \times \mathbf{B}}$. Working through the definition of $\widehat{\mathbf{A} \times \mathbf{B}}$, we may regard the objects of this subcategory as tuples $\langle S, f, g, A, B \rangle$, where f and g are the coordinate functions of an inclusion $S \hookrightarrow |A| \times |B|$. In other words, since S , f and g determine a subset of $|A| \times |B|$, an object $\langle S, f, g, A, B \rangle$ is essentially an ordinary binary relation on $|A| \times |B|$. We will therefore simplify notation and suppress the coordinate functions f and g . We will write $S: |A| \multimap |B|$ to indicate that S is a binary relation on $|A| \times |B|$. This notation makes it possible to express properties of relations using diagrams instead of formulas. As noted above, for morphisms $\langle t, x, y \rangle$ between objects of $\widetilde{\mathbf{A} \times \mathbf{B}}$, the set map t is uniquely determined by \mathbf{A} and \mathbf{B} maps x and y , and the relation S . Specifically, t is the restriction of $|x| \times |y|$ to S . Omitting the redundant set map, the definition of morphism has an appealing diagrammatic presentation as a pair $\langle x, y \rangle$ satisfying:



The “ \subseteq ” in this diagram means that the relation $|y| \circ S$ is included in the relation $S' \circ |x|$. It is readily seen that this condition, expressible as a universal Horn clause, states exactly that for all morphisms $a: \mathbf{1} \rightarrow A$ in \mathbf{A} and $b: \mathbf{1} \rightarrow B$ in \mathbf{B} :

$$a S b \text{ implies } (x \circ a) S' (y \circ b),$$

where we use relational notation $a S b$ to indicate that S relates a to b , and similarly for S' .

It is instructive to write out the exponentials (function spaces) in the cartesian closed category $\widetilde{\mathbf{A} \times \mathbf{B}}$. Suppressing the coordinates of inclusions, $\langle S, A, B \rangle \Rightarrow \langle S', A', B' \rangle$ is given by

$\langle R, A \Rightarrow A', B \Rightarrow B' \rangle$, where $R: |A \Rightarrow A'| \dashv\dashv |B \Rightarrow B'|$ is the binary relation such that for all morphisms $e: \mathbf{1} \rightarrow A \Rightarrow A'$ in \mathbf{A} and $d: \mathbf{1} \rightarrow B \Rightarrow B'$ in \mathbf{B} :

$$e R d \quad \text{iff} \quad a S b \text{ implies } \mathbf{App} \circ \langle e, a \rangle S' \mathbf{App} \circ \langle d, b \rangle$$

for all $a: \mathbf{1} \rightarrow A$ in $\mathbf{A}, b: \mathbf{1} \rightarrow B$ in \mathbf{B} .

We may also write out products in a similar fashion. The subscone $\tilde{\mathbf{C}}$ is a ccc and a full subcategory of \mathbf{C} , and the restriction of the forgetful representation $\mathbf{C} \rightarrow \mathbf{C}$ to $\tilde{\mathbf{C}}$ is also a ccc-representation. The main properties of $\mathbf{A} \times \mathbf{B}$ are summarized in the following proposition, which is easily verified using the ideas presented in the above discussion.

Proposition 4.3 *Let $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ be the cartesian product of two ccc's. Then $\tilde{\mathbf{C}}$ is a cartesian closed category, with canonical functor $\tilde{\mathbf{C}} \rightarrow \mathbf{C}$ a representation of cartesian closed categories. Furthermore, products and exponentials are given by*

$$\langle S, A, B \rangle \times \langle S', A', B' \rangle = \langle S \times S', A \times A', B \times B' \rangle,$$

$$\langle S, A, B \rangle \Rightarrow \langle S', A', B' \rangle = \langle S \Rightarrow S', A \Rightarrow A', B \Rightarrow B' \rangle,$$

where relations $S \times S'$ and $S \Rightarrow S'$ have the following characterizations:

$$a (S \times S') b \quad \text{iff} \quad (\mathbf{Proj}_1 \circ a) S (\mathbf{Proj}_1 \circ b) \quad \text{and} \quad (\mathbf{Proj}_2 \circ a) S' (\mathbf{Proj}_2 \circ b)$$

$$e (S \Rightarrow S') d \quad \text{iff} \quad \forall a: \mathbf{1} \rightarrow A. \forall b: \mathbf{1} \rightarrow B. a S b \text{ implies } \mathbf{App} \circ \langle e, a \rangle S' \mathbf{App} \circ \langle d, b \rangle.$$

The descriptions of products and exponentials in this proposition correspond to the definition of logical relations given *e.g.*, in [Mit90].

Note that although much of the development above uses *Set*, we do not assume that functors $|\cdot|_{\mathbf{A}}: \mathbf{A} \rightarrow \mathbf{Set}$ and $|\cdot|_{\mathbf{B}}: \mathbf{B} \rightarrow \mathbf{Set}$ are one-to-one on morphisms. In other words, we do not assume that categories \mathbf{A} and \mathbf{B} are well-pointed (generated by $\mathbf{1}$). In the argument above, we are simply using the one-to-one correspondence between the morphisms $C \rightarrow C'$ and the morphisms $\mathbf{1} \rightarrow C \Rightarrow C'$, which holds in any cartesian closed category. In the special case of well-pointed cartesian closed categories, we may compare sconing to logical relations for Henkin models of simply typed lambda calculus (see [Sta85, Mit90]). We shall do so in the next section.

We conclude this section with a “binary version” of a general sconing framework given by means of comma categories in Proposition 4.2. This binary version (as well as the k -ary version) can be obtained from Proposition 4.2 itself.

Proposition 4.4 *Let \mathbf{A}, \mathbf{B} , and \mathbf{C}' be cartesian closed categories and assume \mathbf{C}' has equalizers. Let $\mathbf{F}: \mathbf{A} \rightarrow \mathbf{C}'$ and $\mathbf{G}: \mathbf{B} \rightarrow \mathbf{C}'$ be functors that preserve finite products up to isomorphism. Let $\mathbf{H}: \mathbf{A} \times \mathbf{B} \rightarrow \mathbf{C}'$ be the functor given by $\mathbf{H}(A, B) = \mathbf{F}(A) \times \mathbf{G}(B)$, and likewise on morphisms. Then the comma category $(\mathbf{C}' \downarrow \mathbf{H})$ is a cartesian closed category and the canonical functor $(\mathbf{C}' \downarrow \mathbf{H}) \rightarrow \mathbf{A} \times \mathbf{B}$ is a representation of cartesian closed categories.*

5 Sconing and Logical Relations

In this section, we describe the connection between the subscone and the so-called logical relations over Henkin models. We rely on the basic correspondence between well-pointed ccc's and Henkin models described, *e.g.*, in [LS86, MS89]. Logical relations are discussed *e.g.*, in [Sta85, Mit90]. If \mathbf{C} is the cartesian closed category determined by a Henkin model \mathcal{A} , then the logical predicates over \mathcal{A} have a direct correspondence with the ccc-representations from a free cartesian closed category into the subscone of \mathbf{C} . Since binary relations are often more intuitive

than predicates, we will continue to illustrate the main ideas using binary relations. We begin by reviewing the free cartesian closed category over some set of types.

For any set S of type constants, there is a *free* cartesian closed category $\mathcal{F}[S]$ generated by S . The objects of the category $\mathcal{F}[S]$ are types built from the type constants in S and the morphisms $A \rightarrow B$ are equivalence classes of lambda terms of type B with exactly one free variable of type A , where the equivalence relation is given by the equational rules [LS86, MS89]. The freeness of $\mathcal{F}[S]$ means that for any cartesian closed category \mathbf{D} and any assignment $S \rightarrow \mathbf{D}$ mapping the type constants in S to objects of \mathbf{D} there is a unique representation of cartesian closed categories such that

$$\begin{array}{ccc}
 S & \xrightarrow{\quad\quad\quad} & \mathcal{F}[S] \\
 & \searrow & \vdots \\
 & & \mathbf{D}
 \end{array}$$

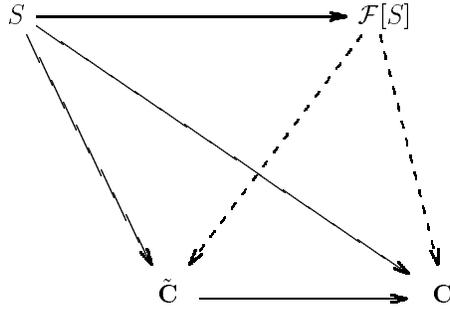
where the map $S \rightarrow \mathcal{F}[S]$ is the inclusion of S into the set of objects of $\mathcal{F}[S]$. The representation of $\mathcal{F}[S]$ into \mathbf{D} is the categorical equivalent of a meaning function $\mathcal{A}[\cdot]$ associated with a Henkin model \mathcal{A} . Because a representation is a functor that preserves cartesian closed structure, this representation gives both a map from simple types over S to objects of \mathbf{D} and a map from of simply typed lambda terms to morphisms of \mathbf{D} . Moreover, these functions together preserve all typing and equational rules of simply typed lambda calculus. One difference between the Henkin model and categorical settings is that we usually consider the mapping from type expressions to sets $\sigma \mapsto A^\sigma$ part of the Henkin model \mathcal{A} , whereas the associations of type constants to objects is not part of a cartesian closed category (hence the dependence on a function $S \rightarrow \mathbf{D}$). Another difference is that the meaning function for Henkin models is a map defined on terms, whereas a representation of $\mathcal{F}[S]$ is a map on equivalence classes. However, this is an inessential difference since either form of map may be obtained from the other in an obvious way. Additional discussion and details may be found in [LS86, MS89] and elsewhere.

Because the subscone $\tilde{\mathbf{C}}$ is a cartesian closed category, any assignment of type constants S to objects of $\tilde{\mathbf{C}}$ determines a representation from $\mathcal{F}[S]$ to $\tilde{\mathbf{C}}$. For product categories, if \mathbf{A} and \mathbf{B} are given by Henkin models \mathcal{A} and \mathcal{B} of simply typed lambda calculus, then $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ and $\tilde{\mathbf{C}}$ are again well-pointed cartesian closed categories. A mapping from simple types to objects of $\tilde{\mathbf{C}}$, as mentioned above, is a type-indexed family of relations $R^\sigma \subseteq A^\sigma \times B^\sigma$ between the Henkin models. It is easy to see from Proposition 4.3 that the family of relations determined by the object map of a representation $\mathcal{F}[S] \rightarrow \tilde{\mathbf{C}}$ is in fact a logical relation.

Proposition 5.1 *Let \mathbf{A} and \mathbf{B} be well-pointed cartesian closed categories and let \mathcal{A} and \mathcal{B} be the corresponding Henkin models of simply typed lambda calculus. Then a logical relation on $\mathbf{A} \times \mathbf{B}$ is exactly the object part of a representation of the free cartesian closed category (on a given set of generators) in the cartesian closed category $\tilde{\mathbf{A}} \times \tilde{\mathbf{B}}$.*

This correspondence and the Basic Lemma for logical relations discussed in [Mit90] are consequences of a straightforward diagram, given in the following proposition.

Proposition 5.2 *Let S be a set of type constants (symbols), let \mathbf{C} be a cartesian closed category, and consider any mapping $S \rightarrow \tilde{\mathbf{C}}$ from type constants to objects of the subscone of \mathbf{C} . Composing with the canonical functor $\tilde{\mathbf{C}} \rightarrow \mathbf{C}$, we obtain a corresponding map $S \rightarrow \mathbf{C}$. Given these two maps from S , and the injection of S into the types of the free ccc $\mathcal{F}[S]$ over S , there exist unique ccc representations commuting with the canonical functor $\tilde{\mathbf{C}} \rightarrow \mathbf{C}$ as follows:*



Proof. Let functions from S to objects of the ccc's \mathbf{C} , $\tilde{\mathbf{C}}$ and $\mathcal{F}[S]$ be given as in the statement of the proposition. It is a trivial consequence of the definitions that the triangle involving S , $\tilde{\mathbf{C}}$ and \mathbf{C} commutes. By the freeness of $\mathcal{F}[S]$, the representations $\mathcal{F}[S] \rightarrow \mathbf{C}$ and $\mathcal{F}[S] \rightarrow \tilde{\mathbf{C}}$ are uniquely determined. This implies that the two triangles with vertices S , $\mathcal{F}[S]$ and either $\tilde{\mathbf{C}}$ or \mathbf{C} commute. Finally, since the canonical functor $\tilde{\mathbf{C}} \rightarrow \mathbf{C}$ is a ccc-representation (see Proposition 4.3), the two uniquely determined representations of $\mathcal{F}[S]$ must commute with this functor. ■

The Basic Lemma follows from this diagram. Indeed, let us consider the special case that \mathbf{A} and \mathbf{B} are given by Henkin models \mathcal{A} and \mathcal{B} of simply typed lambda calculus. Then both $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ and $\tilde{\mathbf{C}}$ are well-pointed cartesian closed categories. The representation $\mathcal{F}[S] \rightarrow \mathbf{C}$ maps a typed lambda term $x: \sigma \triangleright M: \tau$ to the pair of morphisms giving the meaning of M in \mathcal{A} as a function of x and, respectively, the meaning of M in \mathcal{B} . The representation $\mathcal{F}[S] \rightarrow \tilde{\mathbf{C}}$ maps a typed lambda term $x: \sigma \triangleright M: \tau$ to a morphism from some subset $S \subseteq \sigma_{\mathbf{A}} \times \sigma_{\mathbf{B}}$ to a subset $S' \subseteq \tau_{\mathbf{A}} \times \tau_{\mathbf{B}}$, where $\sigma_{\mathbf{A}}$ is the object of \mathbf{A} given by type expression σ over type constants S , according to the chosen map $S \rightarrow \mathbf{C} (= \mathbf{A} \times \mathbf{B})$, and similarly for the other subscripted type expressions. As noted in Section 4.1, such a morphism in the subscone is a product map $f \times g$ with $f: \sigma_{\mathbf{A}} \rightarrow \tau_{\mathbf{A}}$ and $g: \sigma_{\mathbf{B}} \rightarrow \tau_{\mathbf{B}}$ which maps a pair $\langle u, v \rangle$ with $u S v$ to a pair $\langle f(u), g(v) \rangle$ with $f(u) S' g(v)$. Because the two representations of $\mathcal{F}[S]$ commute with the forgetful functor $\tilde{\mathbf{C}} \rightarrow \mathbf{C}$, f must be the meaning $f = \mathcal{A}[[x: \sigma \triangleright M: \tau]]$ of the term M in \mathcal{A} , regarded as a function of the free variable x , and similarly $g = \mathcal{B}[[x: \sigma \triangleright M: \tau]]$. Because the term $x: \sigma \triangleright M: \tau$ was arbitrary, and σ may be a cartesian product of any types, we have shown that given logically related interpretations of the free variables, the meaning of any term in \mathcal{A} is logically related to the meaning of this term in \mathcal{B} . This is precisely the Basic Lemma for logical relations over Henkin models.

6 Generalizations of scoping

The proof of Proposition 5.2 in fact establishes the general version of Proposition 5.2 where instead of $\tilde{\mathbf{C}}$ one considers any ccc \mathbf{D} that has a ccc-representation in \mathbf{C} . For instance, \mathbf{D} might be a subcategory of the comma category $(\mathbf{C}' \downarrow \mathbf{F})$ given in Proposition 4.2, so that the restriction of the canonical ccc-representation $(\mathbf{C}' \downarrow \mathbf{F}) \rightarrow \mathbf{C}$ to \mathbf{D} is a ccc-representation. We also remind the reader of the binary version given in Proposition 4.4. In the general setting that would yield the above diagram with $\mathbf{A} \times \mathbf{B}$ instead of \mathbf{C} and with \mathbf{D} instead of $\tilde{\mathbf{C}}$.

In this section we compare this categorical generalization with three forms of logical relations that have already appeared in the literature, Kripke logical relations [Plo80, MM91], cpo logical relations [MS76, Rey74, CP92], and the relational setting over PER models discussed in Section 4 of [BFS90]. Kripke logical relations over ordinary Henkin models were first used in [Plo80] in a characterization of lambda definability. Kripke logical relations were then adapted to Kripke lambda models in [MM91]. Inductive relations (as well as strict inductive relations) are widely

used in relating denotational semantics and in proofs by fixed-point induction. Relations over PER models may be used to derive parametricity properties of lambda definable functions, see Section 4 of [BFS90].

These cases of generalized logical relations are still “concrete” enough so that it makes sense to consider an analogue of a subscone, a certain subcategory of a comma category consisting of those objects for which relevant mappings are given by inclusions. Hence in these cases we will be able to assume that the objects of the subcategory \mathbf{D} of the comma category also have this form.

6.1 Scoring with Kripke models

Let \mathcal{P} be a partially ordered set. \mathcal{P} may be considered as a category whose objects are the elements of \mathcal{P} , and where for each p, p' in \mathcal{P} there is at most one morphism $p \rightarrow p'$, which exists iff $p \leq p'$ in \mathcal{P} . It is shown in Section 4 of [MM91] that each Kripke lambda model with \mathcal{P} as the set of “possible worlds” determines a cartesian closed category \mathbf{C} and a finite product preserving functor \mathbf{F} from \mathbf{C} to the functor category $Set^{\mathcal{P}}$. (The objects of \mathbf{C} are types, not interpretations of types given by the Kripke lambda model, but the morphisms $\sigma \rightarrow \tau$ in \mathbf{C} are the natural transformations $\Phi_\sigma \rightarrow \Phi_\tau$ induced by the global elements of the Kripke model of type $\sigma \Rightarrow \tau$, where Φ_σ is the interpretation of type σ . The functor \mathbf{F} , given by Φ , need not be a ccc-representation. Furthermore, any ccc is equivalent as a category to a ccc arising this way, see [MM91]. This framework covers the case when \mathbf{C} arises from an ordinary Henkin model, see our Section 5. Indeed, we let $\mathbf{F}(C)$ for each object C of \mathbf{C} be the constant functor $\mathcal{P} \rightarrow Set$ whose value is the domain set in the given Henkin model that corresponds to C .)

Let \mathbf{D} be the full subcategory of the comma category $(Set^{\mathcal{P}} \downarrow \mathbf{F})$ that consists of objects $\langle S, f, A \rangle$ with $f_p: S(p) \hookrightarrow \mathbf{F}(A)(p)$ an inclusion for each p in \mathcal{P} . We may conveniently omit f when referring to objects of \mathbf{D} . \mathbf{D} is a ccc. In describing exponentials (*i.e.*, function spaces) in \mathbf{D} we may use natural transformations $\mathbf{F}(app_{A,A'}): \mathbf{F}(A \Rightarrow A') \times \mathbf{F}(A) \rightarrow \mathbf{F}(A')$ because the functor \mathbf{F} preserves binary products. For each object C in \mathbf{C} and each $p \leq p'$ in \mathcal{P} , we also use the transition function $\mathbf{F}(C)(p \leq p'): \mathbf{F}(C)(p) \rightarrow \mathbf{F}(C)(p')$ given by the action of the functor $\mathbf{F}(C): \mathcal{P} \rightarrow Set$ on the order of \mathcal{P} . Specifically, $\langle S, A \rangle \Rightarrow \langle S', A' \rangle$ may be given as $\langle P, A \Rightarrow A' \rangle$, where $A \Rightarrow A'$ is taken in \mathbf{C} and $P: \mathcal{P} \rightarrow Set$ is the functor such that the set $P(p)$ consists of all t in $\mathbf{F}(A \Rightarrow A')(p)$ for which for all $p' \geq p$ and all a in $\mathbf{F}(A)(p')$, if a belongs to $S(p')$, then the element $\mathbf{F}(app_{A,A'})_{p'}(\mathbf{F}(A \Rightarrow A')(p \leq p')(t), a)$ in $\mathbf{F}(A')(p')$ belongs to $S'(p')$. In other words, the object parts of the free ccc-representations in \mathbf{D} in this case amount to, in the terminology of [MM91], the Kripke logical predicates on the Kripke lambda model that corresponds to \mathbf{F} .

In the binary case, let \mathbf{A} and \mathbf{B} be cartesian closed categories and $\mathbf{F}: \mathbf{A} \rightarrow Set^{\mathcal{P}}$ and $\mathbf{G}: \mathbf{A} \rightarrow Set^{\mathcal{P}}$ finite product preserving functors given by Kripke lambda models \mathcal{A} and \mathcal{B} , each with \mathcal{P} as the poset of possible worlds. Let $\mathbf{H}: \mathbf{A} \times \mathbf{B} \rightarrow Set^{\mathcal{P}}$ be the functor given by $\mathbf{H}(A, B) = \mathbf{F}(A) \times \mathbf{G}(B)$. One now continues as above, with $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ and with \mathbf{H} instead of \mathbf{F} . In this case the object parts of the free ccc-representations in \mathbf{D} are exactly the Kripke logical relations on Kripke lambda models \mathcal{A} and \mathcal{B} .

6.2 Scoring with cpo domains

Let Cpo be the category whose objects are posets with suprema of countably infinite chains, *i.e.*, cpos, and whose morphisms are monotone maps that preserve suprema of countably infinite chains, *i.e.*, continuous maps. Cpo is a cartesian closed category with equalizers. Finite products and equalizers are given as in Set , and $P \Rightarrow Q$ consists of all continuous maps $P \rightarrow Q$, with pointwise order. Let $Pcpo$ be the full subcategory of Cpo whose objects are pointed cpos, *i.e.*, cpos with the least element, usually denoted $-$. Because $P \Rightarrow Q$ as given in Cpo is pointed

if Q is pointed, with $-$ in $P \Rightarrow Q$ being the constant function with value $-$ in Q , $Pcpo$ is a sub-ccc of Cpo . We shall denote the inclusion functor by \mathbf{I} .

While in $Pcpo$ one loses equalizers, one is compensated by gaining, for each object P , a canonical morphism $fix_P: (P \Rightarrow P) \rightarrow P$ that yields fixed-points. If f is an element of $(P \Rightarrow P)$ given by a continuous map $P \rightarrow P$, then $fix_P(f)$ is the least fixed-point of f , given by $fix_P(f) = \bigvee \{f^n(-)\}$. Thus the ccc $Pcpo$ has the property that for each object P , there exists a canonical morphism $fix_P: (P \Rightarrow P) \rightarrow P$ such that for every morphism f targeted at $P \Rightarrow P$, the morphism $p = fix_P \circ f$ targeted at P satisfies $app \circ \langle f, fix_P \circ f \rangle = fix_P \circ f$.

A ccc with this property may be called a *ccc with fixed-point operators*. For any such ccc \mathbf{C} , the free ccc-representation $\mathcal{F}[S] \rightarrow \mathbf{C}$ (induced, as always, by a mapping $S \rightarrow \mathbf{C}$ that interprets type constants in \mathbf{C}) factors through the free ccc with fixed-points on the given collection S of type constants. This latter category $\mathcal{F}_{fix}[S]$ is the term category of simply typed lambda calculus (on the given collection S of type constants) with a constant fix_σ of type $(\sigma \Rightarrow \sigma) \Rightarrow \sigma$ for each type σ , for which we impose

$$\Gamma \triangleright M(fix_\sigma M) = fix_\sigma M : \sigma.$$

In addition, the ccc-representation $\mathcal{F}_{fix}[S] \rightarrow \mathbf{C}$ involved in the factorization of $\mathcal{F}[S] \rightarrow \mathbf{C}$ also preserves the canonical fixed-point operators.

Let us consider the full subcategory of the comma category $(Cpo \downarrow \mathbf{I})$ whose objects are of the form $\langle P, f, A \rangle$, where f is the inclusion $P \hookrightarrow A$. In other words, P is a sub-cpo of $pepo$ A . Yet another equivalent description is that P is an *inductive predicate* on A in the sense that P is monotone in A and for any chain Q in A , if $Q \subseteq P$, then $\bigvee Q \in P$. We omit f when referring to these objects. In this ccc the exponentials, *i.e.*, function spaces $\langle P, A \rangle \Rightarrow \langle P', A' \rangle$ may be described as $\langle R, A \Rightarrow A' \rangle$, where $A \Rightarrow A'$ is taken in Cpo , *i.e.*, it is the cpo of continuous functions $g: A \rightarrow A'$ ordered pointwise, and where R consists of all such g for which $g(a) \in P'$ whenever $a \in P$. The object parts of free ccc-representations in this ccc are exactly inductive logical predicates.

It is also useful to consider a further subcategory \mathbf{D} whose objects are of the form $\langle P, A \rangle$, where $P \hookrightarrow A$ is a sub-pepo, *i.e.*, in addition to being inductive, P is also *strict* in the sense that $-_A \in P$. The ccc structure is still the same, but in \mathbf{D} we also get canonical fixed-point operators, and the forgetful ccc-representation $\mathbf{D} \rightarrow Pcpo$ preserves them. The object parts of free ccc-representations in \mathbf{D} are exactly strict, inductive logical predicates. As we observed above, each such ccc-representation can be lifted to a free ccc-representation that preserves fixed-point operators. In the binary case, so we let $\mathbf{H}(A, B) = A \times B$ and we consider \mathbf{D} be the full subcategory of $(Cpo \downarrow \mathbf{H})$ whose objects are of the form $\langle S, A, B \rangle$, where S is a sub-pepo of $pepo$ $A \times B$, *i.e.*, S is a strict, inductive relation from A to B .

It is immediate to generalize this entire discussion from chains to directed families. Furthermore, it would be instructive to work out a treatment of sconing with cpos from the point of view of computational monads [Mgg89]. In particular, it seems useful to see objects of the form $\langle P, A \rangle$ with $P \hookrightarrow A$ a sub-pepo, as the result of lifting in $(Cpo \downarrow \mathbf{I})$.

6.3 Sconing with partial equivalence relations

Let Per be the category whose objects are partial equivalence relations (*i.e.*, symmetric, transitive relations) on natural numbers, and whose morphisms $R \rightarrow S$ are equivalence classes of partial recursive functions f that are defined on the domain of R and that preserve R in the sense that whenever $n R k$, then $f(n) S f(k)$. Two such partial recursive functions f, f' are considered equivalent iff whenever $n R k$, then $f(n) S f'(k)$. It is folklore that Per is a ccc. The “indiscriminate” relation that relates any two numbers serves as a terminal object. Given a computable pairing function, the product $R \times S$ may be described as relating $\langle n, k \rangle$ and $\langle n', k' \rangle$ iff $n R n'$ and $k S k'$. Given a gödelnumbering of partial recursive functions, $R \Rightarrow S$ may be

described as relating e and e' iff e and e' are gödelnumbers of partial recursive functions that represent per morphisms $R \rightarrow S$ and are equivalent as such.

Let \mathbf{I} be the identity functor on Per . Let \mathbf{D} be the full subcategory of the comma category $(Per \downarrow \mathbf{I})$ of objects $\langle S, f, A \rangle$, where f is named by the identity function on the natural numbers, and where S is saturated in A in the sense that $S = A \circ S \circ A$ as binary relations on natural numbers (*i.e.*, S is a restriction of A to a subset of the domain of A that is a union of A -equivalence classes). We omit f when referring to objects of \mathbf{D} . It is readily verified that $\langle S, A \rangle \Rightarrow \langle S', A' \rangle$ in \mathbf{D} is given by $\langle R, A \Rightarrow A' \rangle$, where the right component is as in Per , and where $e R e'$ iff $e A \Rightarrow A' e'$ and the partial recursive functions with gödelnumbers e and e' also represent per morphisms $S \rightarrow S'$ and are equal as such.

In the binary case the relevant comma category is $(Per \downarrow \mathbf{H})$, where \mathbf{H} is the binary product functor in Per . The objects of the relevant full subcategory may be presented in the form $\langle S, A, B \rangle$, where S is an ordinary relation from the domain of A to the domain of B subject to the saturation condition $S = A \circ S \circ B$, see Section 4 in [BFS90]. The ccc structure may be described as above.

7 Implicit polymorphism

In this section we turn our attention to a slight extension of simply typed lambda calculus. Syntactically, the language λ^{\rightarrow^t} we consider is still very close to simply typed lambda calculus. The only difference is that types may contain type variables as well as basic type symbols. However, our interest in λ^{\rightarrow^t} lies in the fact that the terms of the calculus depend on type variables, and therefore we have a simple form of polymorphism. We will refer to this as “implicit” polymorphism since the quantification over type variables is implicit, rather than explicit.

Given a cartesian closed category \mathbf{C} , let $Obj \mathbf{C}$ be the set (or class) of objects of \mathbf{C} . As with any set (or class), $Obj \mathbf{C}$ may be considered as a discrete category, the only morphisms of which are the identities. We may interpret the types of λ^{\rightarrow^t} as functors from $Obj \mathbf{C}$ to \mathbf{C} , and terms as natural transformations. Note that to interpret types with n type variables, we need functors $(Obj \mathbf{C})^n \rightarrow \mathbf{C}$, and so the interpretation naturally involves a different category for each number of type variables. The reader will note that the functors $(Obj \mathbf{C})^n \rightarrow \mathbf{C}$ are basically just n -ary functions from the set (or class) $Obj \mathbf{C}$ to $Obj \mathbf{C}$. In order to account for type substitution, these functor categories must “fit together” as n varies. A proper general categorical framework for this motivating example is based on indexed or fibred categories, see [See87, Pit87, Mog91]. This framework, discussed below, generalizes readily to explicit polymorphism. In addition, a special case of this categorical view is a construction on Henkin models described, *e.g.*, in [Mit90].

7.1 Categorical models of implicit polymorphism

If we consider the type expressions apart from lambda terms, we have an “algebraic” language with type constants (basic types), type variables, and the binary operations \times and \Rightarrow . We may regard any such language as an “algebraic theory,” in Lawvere’s terminology, or a category with a terminator, binary products, and a distinguished object representing the carrier of the algebra. For definiteness, we will assume a “base” category \mathbf{B} with distinguished object V which *generates* \mathbf{B} , in the sense that each object is a finite power V^n for some natural number n . This gives us the terminator as V^0 . Intuitively, the arrows $V^i \rightarrow V^j$ in \mathbf{B} may be regarded as j -tuples of type expressions, each having all type variables among t_1, \dots, t_i . Composition in \mathbf{B} corresponds to substitution of types for type variables. This relatively standard interpretation of an algebraic language is explained in [KR77], for example. It is helpful to note that we may

regard an arrow $V^i \rightarrow V^j$ as either a j -tuple of types, or a context $\Gamma = \{x_1: \tau_1, \dots, x_j: \tau_j\}$ of length j , with all type expressions over the same set of i type variables.

We now consider the interpretation of terms. For each finite set of type variables, $\lambda^{\rightarrow, t}$ includes all simply typed terms over these variables, regarded as type constants. Therefore, we expect a categorical interpretation to provide a ccc for each finite number of type variables. More specifically, if we regard a morphism $g: V^i \rightarrow V$ of \mathbf{B} as a context $\{x: \tau\}$, we would expect to have some kind of assignment “from” g for each term $\{x: \tau\} \triangleright M: \sigma$. We may use the type variable structure of \mathbf{B} to index categories of types and terms. We assume that for each object V^i in \mathbf{B} , there is a category \mathbf{F}_i whose objects correspond to type expressions with i type variables and morphisms represent terms of these types. This assignment is assumed to be functorial in i , *i.e.*, \mathbf{F} is a (contravariant) functor to Cat , the category of categories. The structure of \mathbf{B} is linked to \mathbf{F}_i by assuming that objects of \mathbf{F}_i are the same as morphisms $V^i \rightarrow V$ in \mathbf{B} . Similar indexed categories may be found in [See87, Pit87, Mog91].

Definition 7.1 *An iml-category is given by:*

- a base category \mathbf{B} with finite products, generated by some distinguished object V ,
- a functor $\mathbf{F}: \mathbf{B}^{op} \rightarrow Cat$ such that:
 - for each object I in \mathbf{B} , the category $\mathbf{F}(I)$ is cartesian closed,
 - for each morphism $f: I \rightarrow J$ in \mathbf{B} , the functor $\mathbf{F}(f): \mathbf{F}(J) \rightarrow \mathbf{F}(I)$ is a ccc representation,
 - for each object I in \mathbf{B} , the class of objects $Obj(\mathbf{F}(I))$ is the same as the class of morphisms $I \rightarrow V$ in \mathbf{B} ,
 - for each morphism $f: I \rightarrow J$ in \mathbf{B} , $\mathbf{F}(f)$ acts on objects by composition.

This definition may be rephrased by means of the category $Class$ of classes and mappings and the category Ccc of cartesian closed categories and representations thereof. In particular, an *iml*-category may be seen as a base category \mathbf{B} with finite products generated by some distinguished object V , together with a functor $\mathbf{F}: \mathbf{B}^{op} \rightarrow Ccc$ such that the functor $Obj \circ \mathbf{F}: \mathbf{B}^{op} \rightarrow Class$ is the representable functor $\mathbf{B}(\cdot, V)$. Here $Obj: Ccc \rightarrow Class$ is the forgetful functor that takes a ccc to its underlying set (or class) of objects and a ccc representation to the associated object map. (Probably the most expedient way of dealing with the question of a set vs. a class of objects is to regard “smallness” as relative to each example.) Let us also observe that following [Pit87, Mog91], we use a tighter definition of indexed categories than usual in category theory; we are asking for data up to equality rather than just up to canonical isomorphisms.

Because the objects of \mathbf{B} have the form V^n for some natural number n , we may identify the objects of \mathbf{B} with the natural numbers. This simplifies notation. The category $\mathbf{F}(n)$, which we also write as \mathbf{F}_n , is often called the *fiber over n* . Again, the intuition behind this definition is that n stands for a set of n distinct type variables, the objects of \mathbf{F}_n for type expressions on these type variables, the morphisms of \mathbf{F}_n for typed terms over these type variables, and $\mathbf{F}(f)$ for type substitution. This intuition may be made precise by defining a *term iml-category* for each $\lambda^{\rightarrow, t}$ language and theory, analogously to [See87, Pit87].

Example 7.2 Given any ccc \mathbf{C} , we may construct an *iml*-category \mathbf{C}_{iml} as follows. We let the base category \mathbf{B} have natural numbers as objects, and let the morphisms $i \rightarrow j$ be functors $(Obj \mathbf{C})^i \rightarrow (Obj \mathbf{C})^j$. Fiber \mathbf{F}_n over n is the functor category $(Obj \mathbf{C})^n \rightarrow \mathbf{C}$. We emphasize that these functors are basically just functions from n -tuples of objects to objects, and therefore the ccc structure of \mathbf{F}_n is in this example given pointwise (*i.e.*, objectwise) by the ccc structure of \mathbf{C} . For instance, $(F \Rightarrow G)(A) = F(A) \Rightarrow G(A)$. Finally, if $H: m \rightarrow n$ is a morphism of the base category, and therefore a functor $(Obj \mathbf{C})^m \rightarrow (Obj \mathbf{C})^n$, we define the functor $\mathbf{F}(H): \mathbf{F}_n \rightarrow \mathbf{F}_m$ by composition with H . More precisely, $\mathbf{F}(H): \mathbf{F}_n \rightarrow \mathbf{F}_m$ takes a morphism $t: F \rightarrow G$ of the fiber \mathbf{F}_n to the morphism $(t \circ H): F \circ H \rightarrow G \circ H$ of the fiber \mathbf{F}_m , where for any m -tuple A of

objects of \mathbf{C} $(t \circ H)_A = t_{H(A)}$. It is easy to check that $\mathbf{F}(H): \mathbf{F}_n \rightarrow \mathbf{F}_m$ is a ccc representation. We thus see that \mathbf{C}_{iml} is an *iml*-category for any cartesian closed category \mathbf{C} . ■

Example 7.3 A more subtle example of an *iml*-category may be found in the setting used in [Gir86]. Consider the category Qd_{emb} of qualitative domains and embedding-projection pairs. This is *not* a cartesian closed category. Let the morphisms $k \rightarrow n$ of the base category \mathbf{B} be n -tuples of functors $(Qd_{emb})^k \rightarrow Qd_{emb}$ that preserve pullbacks and directed colimits. The fiber \mathbf{F}_n is the ccc whose objects are functors $(Qd_{emb})^n \rightarrow Qd_{emb}$ that preserve pullbacks and directed colimits, *i.e.*, variable types, in the terminology of [Gir86]. The morphisms $t: F \rightarrow G$ of the fiber \mathbf{F}_n are certain “stable” families of stable maps $t_P: F(P) \rightarrow G(P)$, where P ranges over n -tuples of qualitative domains, *i.e.*, in the terminology of [Gir86], objects of variable type $F \Rightarrow G$. Here $(F \Rightarrow G)(P) = F(P) \Rightarrow G(P)$, the latter in the sense of the ccc Qd of qualitative domains and stable maps. The action of $F \Rightarrow G$ on embeddings is explained in [Gir86]. If $H: k \rightarrow n$ is a morphism in the base category \mathbf{B} , then the ccc-representation $\mathbf{F}(H): \mathbf{F}_n \rightarrow \mathbf{F}_k$ takes a morphism $t: F \rightarrow G$ of the fiber \mathbf{F}_n to the morphism $(t \circ H): F \circ H \rightarrow G \circ H$ of the fiber \mathbf{F}_k , where for any k -tuple Q of qualitative domains $(t \circ H)_Q = t_{H(Q)}$. ■

Example 7.4 Let us now describe the *iml*-category PER . We recall the category Per of pers and per morphisms discussed in Section 6.3. Let the morphisms $k \rightarrow n$ of the base category be n -tuples of maps from $(ObjPer)^k$ to $ObjPer$. The fiber \mathbf{F}_n is a ccc whose objects are maps from $(ObjPer)^n$ to $ObjPer$, and whose morphisms $t: F \rightarrow G$ are “realizable” families of per morphisms $t_P: F(P) \rightarrow G(P)$, where P ranges over n -tuples of pers, given by a single partial recursive function that names each per morphism t_P . Two partial recursive functions name the same realizable family of per morphisms $t_P: F(P) \rightarrow G(P)$ iff they name the same per morphism t_P for each n -tuple P of pers. If $H: k \rightarrow n$ is a morphism in the base category, then the ccc-representation $\mathbf{F}(H): \mathbf{F}_n \rightarrow \mathbf{F}_k$ takes a morphism $t: F \rightarrow G$ of the fiber \mathbf{F}_n to the morphism $(t \circ H): F \circ H \rightarrow G \circ H$ of the fiber \mathbf{F}_k , where for any k -tuple Q of pers $(t \circ H)_Q = t_{H(Q)}$. ■

So far we have defined *iml*-categories and discussed several examples. Recalling that the objects of base categories are identified with the natural numbers, let us now define *iml*-representations:

Definition 7.5 Let (\mathbf{B}, \mathbf{F}) , $(\mathbf{B}', \mathbf{F}')$ be *iml*-categories. An *iml*-representation $(\mathbf{B}, \mathbf{F}) \rightarrow (\mathbf{B}', \mathbf{F}')$ is given by:

- a functor $T: \mathbf{B} \rightarrow \mathbf{B}'$ that preserves finite products so that $T(n) = n$, together with
- a natural transformation $t: \mathbf{F} \rightarrow (\mathbf{F}' \circ T)$ such that for each n , the functor $t_n: \mathbf{F}_n \rightarrow \mathbf{F}'_n$ is a ccc-representation,

such that for any morphism $H: n \rightarrow \mathbf{1}$ in \mathbf{B} , the morphism $T(H): n \rightarrow \mathbf{1}$ in \mathbf{B}' is the object $t_n(H)$ in the fiber \mathbf{F}'_n when H is considered as an object in the fiber \mathbf{F}_n .

A special case of this definition is when an *iml*-category (\mathbf{B}, \mathbf{F}) is a *sub-iml*-category of an *iml*-category $(\mathbf{B}', \mathbf{F}')$ in the sense that base category \mathbf{B} is a subcategory of base category \mathbf{B}' , (the objects of either are natural numbers), for each n the fiber \mathbf{F}_n is a sub-ccc of the fiber \mathbf{F}'_n , and for each morphism $H: k \rightarrow n$ the ccc-representation $\mathbf{F}(H): \mathbf{F}_n \rightarrow \mathbf{F}_k$ is a restriction of the ccc-representation $\mathbf{F}'(H): \mathbf{F}'_n \rightarrow \mathbf{F}'_k$. For instance, the *iml*-category PER described in Example 7.4 is a *sub-iml*-category of Per_{iml} , defined in Example 7.2.

Let (\mathbf{B}, \mathbf{F}) be any *iml*-category. The intuitive motivation, mentioned above, that relates morphisms of \mathbf{B} to types of $\lambda^{\rightarrow, t}$ and morphisms of fibers to terms of $\lambda^{\rightarrow, t}$ can now be restated in a precise way. For any *iml*-category (\mathbf{B}, \mathbf{F}) there is a unique *iml*-representation from the term *iml*-category (mentioned just before Example 7.2) to (\mathbf{B}, \mathbf{F}) that extends a given assignment of objects of the fiber \mathbf{F}_0 to basic type symbols. In other words, the term *iml*-category on the given set S of basic type symbols is the free *iml*-category on S . Indeed, for each n , types and terms

with n type variables are mapped to the objects and morphisms of the fiber \mathbf{F}_n , respectively, by the unique ccc-representation of the free ccc in the ccc \mathbf{F}_n . These ccc-representations commute with type substitution.

On the other hand, while it may seem at first that for any ccc \mathbf{C} the canonical ccc-representation $\tilde{\mathbf{C}} \rightarrow \mathbf{C}$ naturally induces an *iml*-representation $(\tilde{\mathbf{C}})_{iml} \rightarrow \mathbf{C}_{iml}$, a second thought reveals there is a problem constructing a functor $t_n: (\tilde{\mathbf{C}}_{iml})_n \rightarrow (\mathbf{C}_{iml})_n$ for each n . Although the canonical ccc-representation $\tilde{\mathbf{C}} \rightarrow \mathbf{C}$ serves for the simplest case, $n = 0$, a problem already arises for $n = 1$. A functor t_1 from $(Obj \tilde{\mathbf{C}}) \rightarrow \tilde{\mathbf{C}}$ to $(Obj \mathbf{C}) \rightarrow \mathbf{C}$ is required. This only seems possible, for arbitrary \mathbf{C} , if, for each map $(Obj \tilde{\mathbf{C}}) \rightarrow (Obj \tilde{\mathbf{C}})$ sending $\langle S, A \rangle$ to $\langle S', A' \rangle$, the object A' is chosen as a function of A , independent of S . This problem is addressed by the concept of *relator*.

7.2 Relators

Two of the main concepts discussed in this paper are *relators* and *relator transformations*. These concepts and their basic properties will be derived in Section 7.3 by a combination of scoping and the construction described in Example 7.2. Nevertheless, let us first spell out a concrete presentation of relators and relator transformations, not necessarily in most general terms. As with logical relations and scoping for simply typed lambda calculus, binary relators seem to give the most intuitive and visual picture of the general case.

Definition 7.6 *Let \mathbf{C}, \mathbf{D} be categories with terminators. A (binary) relator from \mathbf{C} to \mathbf{D} consists of:*

- *an object map $F: Obj \mathbf{C} \rightarrow Obj \mathbf{D}$, together with*
- *a mapping that to any binary relation $S: |A| \multimap |B|$ associates a binary relation $S': |F(A)| \multimap |F(B)|$.*

The more general case of a binary relator of n arguments is defined similarly, using \mathbf{C}^n in place of \mathbf{C} as the domain. In other words, a binary relator of n arguments from \mathbf{C} to \mathbf{D} consists of an object map $F: (Obj \mathbf{C})^n \rightarrow Obj \mathbf{D}$, together with a mapping that to any n -tuple of binary relations $S_i: |A_i| \multimap |B_i|$, where $i = 1, \dots, n$, associates a binary relation $S': |F(A_1, \dots, A_n)| \multimap |F(B_1, \dots, B_n)|$. Even more generally, k -ary relators take k -ary relations to k -ary relations. Further generalizations will be considered in Section 7.3. When $\mathbf{C} = \mathbf{D}$ we shall speak of relators *on* \mathbf{C} .

Note. In [AbJ91] a relator is also required to map identity relations to identity relations. This stricter notion also fits in our framework, see Example 7.10.

Definition 7.7 *Let F, G be relators from \mathbf{C} to \mathbf{D} . A relator transformation from F to G is a family of morphisms $t_A: F(A) \rightarrow G(A)$ in \mathbf{D} , with A ranging over the objects of \mathbf{C} , such that for each binary relation $S: |A| \multimap |B|$*

$$\begin{array}{ccc}
|F(A)| & \xrightarrow{|t_A|} & |G(A)| \\
\downarrow S' & \subseteq & \downarrow S'' \\
|F(B)| & \xrightarrow{|t_B|} & |G(B)|
\end{array}$$

where S' and S'' are the relations assigned to S by F and G , respectively.

We remind the reader that, as in Section 4.1, the “ \subseteq ” in this diagram means that the relation $|t_B| \circ S'$ is included in the relation $S'' \circ |t_A|$. It is readily seen that this condition, expressible as a universal Horn clause, states exactly that for all morphisms $a: \mathbf{1} \rightarrow F(A)$ and $b: \mathbf{1} \rightarrow F(B)$ in \mathbf{D} :

$$a S' b \text{ implies } (t_A \circ a) S'' (t_B \circ b),$$

where we use relational notation $a S' b$ to indicate that S' relates a to b , and similarly for S'' .

These concepts were motivated by work of Wadler [Wad89]. In a modified form (see Example 7.10), they have been applied to the so-called strictness analysis [AbJ91]. We also note that although relators are slightly more general than *tabular structors* proposed in [Fre93], the Horn clause condition involved in the definition of relator transformations is basically the same as the condition defining transformations of tabular structors given in [Fre93]. Tabular structors map morphisms to relations rather than relations to relations, so the change from tabular structors to relators is required to account for type substitution.

There are three natural notions of composition involving relators and relator transformations. The first one is composition of relators, which corresponds to substitution of types for type variables in types. It is transparent that composing two relators yields a relator. (Tabular structors, on the other hand, do not compose.) Secondly, one has composition of relator transformations, which corresponds to substitution of terms for variables in terms. If $t: F \rightarrow G$ and $u: G \rightarrow H$ are relator transformations between relators from \mathbf{C} to \mathbf{D} , let $(u \circ t)$ be the family $(u \circ t)_A = u_A \circ t_A$ of morphisms in \mathbf{D} , with A ranging over the objects of \mathbf{C} . It is straightforward to check that this family is a relator transformation from F to H . The third notion of composition corresponds to substitution of types for type variables in terms. If $t: F \rightarrow G$ is a relator transformation between relators from \mathbf{C} to \mathbf{D} and if H is a relator from \mathbf{B} to \mathbf{C} , then consider the family $(t \circ H)_A = t_{H(A)}$ of morphisms in \mathbf{D} , where A ranges over the objects of \mathbf{B} . It is again readily checked that this is a relator transformation, this time from $F \circ H$ to $G \circ H$.

7.3 Sconing and *iml*-categories

Let us now return to the problem mentioned at the end of Section 7.1. An interesting application of the *iml*-category construction described in Example 7.2 arises when the ccc in question is a scone $\hat{\mathbf{C}}$, or the subscone $\tilde{\mathbf{C}}$, see Section 4. By Example 7.2, both $(\hat{\mathbf{C}})_{iml}$, and $(\tilde{\mathbf{C}})_{iml}$ are *iml*-categories. We may regard $(\hat{\mathbf{C}})_{iml}$ as a kind of “scone” of \mathbf{C}_{iml} , and $(\tilde{\mathbf{C}})_{iml}$ as a kind of “subscone” of \mathbf{C}_{iml} . However, both of these categories are in a sense too generous in the collection of morphisms of the base category. For instance, without restricting our attention

to a smaller sub-*iml*-category than $(\tilde{\mathbf{C}})_{iml}$, we would not have a straightforward forgetful *iml*-representation of this new *iml*-category in \mathbf{C}_{iml} itself. The reader will recall that the existence of a ccc-representation of $\tilde{\mathbf{C}}$ in \mathbf{C} was essential for the scoping method, and particularly in Propositions 5.1 and 5.2.

The problem may be stated more precisely: since even $(\tilde{\mathbf{C}})_{iml}$ is constructed using *all* functions from $Obj \tilde{\mathbf{C}}$ to $Obj \tilde{\mathbf{C}}$, there may be morphisms in the new base category which do not correspond to functions from $Obj \mathbf{C}$ to $Obj \mathbf{C}$. Therefore, given an arbitrary function $\Phi: Obj \tilde{\mathbf{C}} \rightarrow Obj \tilde{\mathbf{C}}$, we would like to ensure the existence of a function $F: Obj \mathbf{C} \rightarrow Obj \mathbf{C}$ with

$$\Phi(S, A) = (S', F(A))$$

for every object $\langle S, A \rangle$ of $\tilde{\mathbf{C}}$. Notice that if there is such a function on $Obj \mathbf{C}$, it is unique. The equation above can be restated as

$$\begin{array}{ccc} Obj \tilde{\mathbf{C}} & \xrightarrow{\Phi} & Obj \tilde{\mathbf{C}} \\ \downarrow & & \downarrow \\ Obj \mathbf{C} & \xrightarrow{F} & Obj \mathbf{C} \end{array}$$

where vertical maps are given by the canonical functor that forgets S in $\langle S, A \rangle$. When there is a function F with this property, we say that Φ is an *extension* of F .

A function $\Phi: Obj \tilde{\mathbf{C}} \rightarrow Obj \tilde{\mathbf{C}}$ that extends some function $F: Obj \mathbf{C} \rightarrow Obj \mathbf{C}$ is just a (unary) relator on \mathbf{C} . In other words, the unary relators on \mathbf{C} are object maps on $\tilde{\mathbf{C}}$ that extend object maps on the “underlying” category \mathbf{C} . Equivalently, a (unary) relator on \mathbf{C} consists of an object map $F: Obj \mathbf{C} \rightarrow Obj \mathbf{C}$, together with a mapping that associates a subset $S' \hookrightarrow |F(A)|$ to any subset $S \hookrightarrow |A|$. More generally, it is possible to define relators that map n -tuples of k -ary relations to k -ary relations, for each k and n . Let $\mathbf{E} = \mathbf{C}^k$. A relator given by a map from $(Obj \tilde{\mathbf{E}})^n$ to $Obj \tilde{\mathbf{E}}$ will be called a k -ary relator of n arguments. The definition of unary relator of n arguments (so $k = 1$) is therefore derived as in the diagram above, using \mathbf{C}^n in place of \mathbf{C} in the domain of F and $(\tilde{\mathbf{C}})^n$ in place of $\tilde{\mathbf{C}}$ in the domain of Φ . We thus obtain the unary case of Definition 7.6.

Rather than consider arbitrary natural transformations between (unary) relators on \mathbf{C} , we will be interested in natural transformations that “extend” natural transformations of \mathbf{C}_{iml} . More precisely, let $\Phi, \Psi: Obj \tilde{\mathbf{C}} \rightarrow \tilde{\mathbf{C}}$ be functors from the discrete category $Obj \tilde{\mathbf{C}}$ to $\tilde{\mathbf{C}}$. While these functors are essentially object maps $Obj \tilde{\mathbf{C}} \rightarrow Obj \tilde{\mathbf{C}}$, the natural transformations between them are families of *arbitrary* morphisms $\tau_{(S,A)}: \Phi(S, A) \rightarrow \Psi(S, A)$ in $\tilde{\mathbf{C}}$, with (S, A) ranging over arbitrary objects of $\tilde{\mathbf{C}}$. (These are *not* the same as natural transformations between functors $Obj \tilde{\mathbf{C}} \rightarrow Obj \tilde{\mathbf{C}}$ into the discrete category.) If Φ, Ψ are relators that extend object maps $F, G: Obj \mathbf{C} \rightarrow Obj \mathbf{C}$, a natural transformation $\tau: \Phi \rightarrow \Psi$ that extends a (necessarily unique) natural transformation t between the associated functors $F, G: Obj \mathbf{C} \rightarrow \mathbf{C}$ is precisely a relator transformation from Φ to Ψ , as in Definition 7.7. It is worth observing that if τ is a relator transformation extending t , then in fact t also determines τ uniquely. (We observed a similar fact in regard to the morphisms in subscones in Section 4.) Therefore, we often say that t is a relator transformation. In other words, if Φ, Ψ are relators on \mathbf{C} with object maps $F, G: Obj \mathbf{C} \rightarrow Obj \mathbf{C}$, a relator transformation t from Φ to Ψ is a family of morphisms

$t_A: F(A) \rightarrow G(A)$ in \mathbf{C} , with A ranging over arbitrary objects of \mathbf{C} , such that for any subset $S \hookrightarrow |A|$, t_A induces a (necessarily unique) morphism $\Phi(S, A) \rightarrow \Psi(S, A)$ in $\tilde{\mathbf{C}}$.

It can now be easily verified that relators and relator transformations form an *iml*-category.

Proposition 7.8 *Let \mathbf{C} be a ccc and let $\mathbf{D} = \tilde{\mathbf{C}}$. The category of (unary) relators of n arguments and relator transformations on a ccc \mathbf{C} is a sub-ccc of the fiber $F_n = \mathbf{D}^{(\text{Obj } \mathbf{D})^n}$ of the *iml*-category \mathbf{D}_{iml} . Furthermore, let the morphisms $k \rightarrow n$ of a new base category \mathbf{B}' be n -tuples of relators of k arguments over \mathbf{C} . Then the indexed category $(\mathbf{B}', \text{Rel } \mathbf{C})$ of relators and relator transformations on \mathbf{C} is a sub-*iml*-category of \mathbf{D}_{iml} and the canonical forgetful functor $(\mathbf{B}', \text{Rel } \mathbf{C}) \rightarrow \mathbf{C}_{iml}$ is an *iml*-representation.*

We saw in Section 4.1 that the binary case of sconing is an instance of sconing. The notion of binary relator on a ccc \mathbf{C} given in Definition 7.6 may be similarly derived from the notion of unary relator just discussed. Notice that it is not sufficient to simply let \mathbf{D} be the subscone of $\mathbf{C} \times \mathbf{C}$ and then proceed as in Proposition 7.8 because we would get too many object maps. Indeed, we would get all maps $(\text{Obj } \mathbf{C})^2 \rightarrow (\text{Obj } \mathbf{C})^2$, whereas for binary relators we need only those maps induced by pairs of maps $F', F'': \text{Obj } \mathbf{C} \rightarrow \text{Obj } \mathbf{C}$, *i.e.*, those mapping a pair of objects (A, B) to $(F'(A), F''(B))$. (In addition, we should restrict further $F' = F''$, but this further restriction can be dealt with naturally, see below.) Here the relevant *iml*-category is the product *iml*-category $\mathbf{C}_{iml} \times \mathbf{C}_{iml}$. In general, the product of two *iml*-categories is the *iml*-category whose base category morphisms $k \rightarrow n$ are pairs of such morphisms from the two given base categories, and whose fiber over n is the ordinary product of the two given fibers over n . The two projections are *iml*-representations.

With this in mind, we can revisit the commutative square above but with $\tilde{\mathbf{C}}$ replaced by \mathbf{D} , where \mathbf{D} is the subscone of $\mathbf{C} \times \mathbf{C}$, and with \mathbf{C} replaced by $\mathbf{C} \times \mathbf{C}$. Moreover, F ranges over objects of the fiber over $\mathbf{1}$ in the product *iml*-category $\mathbf{C}_{iml} \times \mathbf{C}_{iml}$. Such maps $\Phi: \mathbf{D} \rightarrow \mathbf{D}$ are given by two object maps $F', F'': \text{Obj } \mathbf{C} \rightarrow \text{Obj } \mathbf{C}$ by the requirement $\Phi(S, A, B) = (S', F'(A), F''(B))$. The restriction on transformations follows the unary case: we consider only the families of morphisms $\Phi(S, A, B) \rightarrow \Psi(S, A, B)$ in \mathbf{D} , with (S, A, B) ranging over the objects of \mathbf{D} , that extend the families of morphisms in $\mathbf{C} \times \mathbf{C}$ given by pairs of morphisms $F'(A) \rightarrow G'(A)$, $F''(B) \rightarrow G''(B)$ in \mathbf{C} . In this way we obtain a sub-ccc of the fiber over $\mathbf{1}$ of \mathbf{D}_{iml} . Binary relators on \mathbf{C} and relator transformations are clearly a further subcategory of this ccc, and as such they inherit the ccc structure.

For n arguments, we consider the analogous commutative square condition, with \mathbf{D}^n in place of \mathbf{D} in the domain of Φ , and with $(\mathbf{C} \times \mathbf{C})^n$ in place of $\mathbf{C} \times \mathbf{C}$ in the domain of F , obtaining a sub-ccc of the fiber over n of \mathbf{D}_{iml} . We have constructed an *iml*-category (with base morphisms given in the obvious way so as to have an *iml*-category) for which the canonical forgetful functor to $\mathbf{C}_{iml} \times \mathbf{C}_{iml}$ is an *iml*-representation. Furthermore, binary relators and their transformations form a sub-*iml*-category of the *iml*-category we have just constructed. Thus the image of the free *iml*-representation of the term *iml*-category in the *iml*-category we have constructed contains only binary relators, *i.e.*, the free *iml*-representation factors through the *iml*-category of binary relators and relator transformations.

Having observed a specialization of Proposition 7.8 to the binary case (k -ary case is analogous), let us also observe that a similar reasoning allows us to generalize Proposition 7.8 along the lines of Proposition 4.2 and Section 6. For this purpose let \mathbf{C} and \mathbf{C}' be cartesian closed categories, with \mathbf{C}' having equalizers, and let $\mathbf{F}: \mathbf{C} \rightarrow \mathbf{C}'$ be a functor that preserves finite products up to isomorphism. Let ccc \mathbf{D} be a full subcategory of the comma category $(\mathbf{C}' \downarrow \mathbf{F})$, so that the restriction of the canonical ccc-representation $(\mathbf{C}' \downarrow \mathbf{F}) \rightarrow \mathbf{C}$ to \mathbf{D} is a ccc-representation. For now let us also assume that this ccc-representation $\mathbf{D} \rightarrow \mathbf{C}$ is surjective on objects. (This additional surjectivity condition does hold in interesting cases for Kripke models, cpos, and pers.) Let us reconsider the commutative square given above with this new choice of \mathbf{D} in place of $\tilde{\mathbf{C}}$. We say that Φ is a (unary) \mathbf{D} -relator on \mathbf{C} that extends a (necessarily unique) object

map F . If Φ, Ψ are \mathbf{D} -relators that extend object maps $F, G: \text{Obj } \mathbf{C} \rightarrow \text{Obj } \mathbf{C}$, a natural transformation $\tau: \Phi \rightarrow \Psi$ that extends a (necessarily unique) natural transformation t between the associated functors $F, G: \text{Obj } \mathbf{C} \rightarrow \mathbf{C}$ is said to be a \mathbf{D} -relator transformation from Φ to Ψ . (While in general t does not determine τ , we have already remarked in Section 6 that Kripke models, cpos, and pers are “concrete” enough so that we may assume that in the objects of \mathbf{D} the mapping components are given by inclusions, and hence that t determines τ uniquely.) We obtain:

Proposition 7.9 *Let \mathbf{C} and \mathbf{C}' be cartesian closed categories, \mathbf{C}' with equalizers. Let $\mathbf{G}: \mathbf{C} \rightarrow \mathbf{C}'$ be a functor that preserves finite products up to isomorphism. Let ccc \mathbf{D} be a full subcategory of the comma category $(\mathbf{C}' \downarrow \mathbf{G})$, so that the restriction of the canonical ccc-representation $(\mathbf{C}' \downarrow \mathbf{G}) \rightarrow \mathbf{C}$ to \mathbf{D} is a ccc-representation. Furthermore, let this ccc-representation $\mathbf{D} \rightarrow \mathbf{C}$ be surjective on objects. Then the category of (unary) \mathbf{D} -relators of n arguments and \mathbf{D} -relator transformations on a ccc \mathbf{C} is a sub-ccc of the fiber $F_n = \mathbf{D}^{(\text{Obj } \mathbf{D})^n}$ of the *iml*-category \mathbf{D}_{iml} . Furthermore, let the morphisms $k \rightarrow n$ of a new base category \mathbf{B}' be n -tuples of \mathbf{D} -relators of k arguments over \mathbf{C} . Then the indexed category $(\mathbf{B}', \text{Rel}_{\mathbf{D}} \mathbf{C})$ of \mathbf{D} -relators and relator transformations on \mathbf{C} is a sub-*iml*-category of \mathbf{D}_{iml} and the canonical forgetful functor $(\mathbf{B}_{\mathbf{D}}, \text{Rel}_{\mathbf{D}} \mathbf{C}) \rightarrow \mathbf{C}_{iml}$ is an *iml*-representation.*

Example 7.10 Let us illustrate the notions involved in Proposition 7.9 in the setting used in [AbJ91] for the purposes of the so-called strictness analysis. In this example \mathbf{C}' is the category Cpo of cpos and continuous maps, \mathbf{C} is the full subcategory $Pcpo$ of pointed cpos, and \mathbf{G} is the inclusion functor \mathbf{I} . We concentrate on the binary case, so we let $\mathbf{H}(A, B) = A \times B$ and we let \mathbf{D} be the full subcategory of $(Cpo \downarrow \mathbf{H})$ whose objects are of the form $\langle S, A, B \rangle$, where S is a sub-pcpo of $pcpo A \times B$, i.e., S is a strict, inductive relation from A to B , see Section 6.2. \mathbf{D} is a sub-ccc of the comma category and furthermore \mathbf{D} is a ccc with fixed-point operators. A binary pcpo-relator on $Pcpo$ consists of a mapping F from pcpos to pcpos, together with a mapping that to any strict, inductive relation $S: A \rightarrow B$ associates a strict, inductive relation $S': F(A) \rightarrow F(B)$. If F, G are pcpo-relators on pcpos, a relator transformation from F to G is a family of continuous maps $t_A: F(A) \rightarrow G(A)$, with A ranging over pcpos, such that for each strict, inductive relation $S: A \rightarrow B$ and for the two strict, inductive relations $S': F(A) \rightarrow F(B)$ and $S'': G(A) \rightarrow G(B)$ assigned to S by F and by G , respectively, it is the case that the composite inductive relation $t_B \circ S'$ is included in the inductive relation $S'' \circ t_A$. This condition states exactly that for all elements a in $pcpo F(A)$ and b in $pcpo F(B)$, $a S' b$ implies $(t_A(a)) S'' (t_B(b))$. We have described the fiber over $\mathbf{1}$ in the *iml*-category of pcpo-relators over $Pcpo$ and their transformations, a special case of Proposition 7.9. This *iml*-category has fixed-point operators in the sense that each fiber is a ccc with fixed-point operators and they are preserved by the ccc-representations $\mathbf{F}_n \rightarrow \mathbf{F}_k$ induced by morphisms $k \rightarrow n$ in the base category. Pcpo-relators over $Pcpo$ that take identity relations to identity relations form a sub-*iml*-category with fixed-point operators. Thus the free *iml*-representation of the $\lambda^{\rightarrow t}$ term *iml*-category in the *iml*-category of pcpo-relators over $Pcpo$ (determined by an interpretation of type constants in \mathbf{D}) factors through the sub-*iml*-category of pcpo-relators that take identities to identities. Furthermore, since this sub-*iml*-category also has fixed-point operators, the latter *iml*-representation factors through the term *iml*-category of $\lambda^{\rightarrow t}$ with fixed-point operators, as in Section 6.2. Thus the main soundness theorem in [AbJ91] follows. \blacksquare

7.4 Relators over *iml*-categories

Proposition 7.8 may also be generalized in another way. The reader will notice that in our definition of (unary) relators on a ccc \mathbf{C} , the required object maps $F: \text{Obj } \mathbf{C} \rightarrow \text{Obj } \mathbf{C}$ are exactly the objects of the fiber over $\mathbf{1}$ in the *iml*-category \mathbf{C}_{iml} , where the fiber over $\mathbf{0}$ is the ccc

C. In other words, we have constructed an *iml*-category of relators and relator transformations, starting from a particular kind of *iml*-category, \mathbf{C}_{iml} .

However, the construction easily generalizes to arbitrary *iml*-categories. Let us recall that in any *iml*-category (\mathbf{B}, \mathbf{F}) , the objects of the fiber \mathbf{F}_n over n are exactly the morphisms $n \rightarrow \mathbf{1}$ in the given base category \mathbf{B} . In particular, an object of the fiber \mathbf{F}_1 is a morphism $\mathbf{1} \rightarrow \mathbf{1}$ in \mathbf{B} , and hence it composes in \mathbf{B} with any morphism $\mathbf{0} \rightarrow \mathbf{1}$ to produce again a morphism $\mathbf{0} \rightarrow \mathbf{1}$. In this way an object of the fiber \mathbf{F}_1 induces an object map $F: \text{Obj} \mathbf{F}_0 \rightarrow \text{Obj} \mathbf{F}_0$. Hence in the more general definition of a relator over an *iml*-category \mathbf{F} it makes sense to consider only those object maps $F: \text{Obj} \mathbf{F}_0 \rightarrow \text{Obj} \mathbf{F}_0$ induced by the objects of the fiber \mathbf{F}_1 by composition. An object of the fiber \mathbf{F}_n similarly induces an object map of n arguments on \mathbf{F}_0 by composition with n -tuples of morphisms $\mathbf{0} \rightarrow \mathbf{1}$ (*i.e.*, with morphisms $\mathbf{0} \rightarrow n$) in \mathbf{B} .

Definition 7.11 *A (unary) relator of n arguments over an *iml*-category (\mathbf{B}, \mathbf{F}) is given by an object A of the fiber \mathbf{F}_n , together with a (unary) relator of n arguments on \mathbf{F}_0 , whose object map is induced by A by composition.*

Relator transformations over an *iml*-category (\mathbf{B}, \mathbf{F}) may be defined in a similar manner. Let Φ, Ψ be relators over (\mathbf{B}, \mathbf{F}) , whose object maps F, G are induced by objects A, B of the fiber \mathbf{F}_1 . Any morphism $x: A \rightarrow B$ in the fiber \mathbf{F}_1 induces a family of morphisms $t_C: F(C) \rightarrow G(C)$ in the fiber \mathbf{F}_0 , with C ranging over the objects of the fiber \mathbf{F}_0 , as follows. Because $\mathbf{F}(C): \mathbf{F}_1 \rightarrow \mathbf{F}_0$ is a functor, it is the case that $\mathbf{F}(C)(x): \mathbf{F}(C)(A) \rightarrow \mathbf{F}(C)(B)$. Furthermore, the functor $\mathbf{F}(C)$ acts on objects by composition, so $\mathbf{F}(C)(A) = A \circ C$ and $\mathbf{F}(C)(B) = B \circ C$, and thus in fact $\mathbf{F}(C)(x): A \circ C \rightarrow B \circ C$. But $F(C) = A \circ C$ and $G(C) = B \circ C$. We shall be interested only in those relator transformations $\Phi \rightarrow \Psi$ for which the associated families t are induced by morphisms of the fiber \mathbf{F}_1 by composition, as just described. Similarly, for relators of n arguments over (\mathbf{B}, \mathbf{F}) , we shall be interested only in those relator transformations for which the associated transformations between object maps are induced by morphisms of the fiber \mathbf{F}_n .

Definition 7.12 *Let $(A, \Phi), (B, \Psi)$ be relators of n arguments over an *iml*-category (\mathbf{B}, \mathbf{F}) . Let $F, G: (\text{Obj} \mathbf{F}_0)^n \rightarrow \mathbf{F}_0$ be functors induced by A and by B , respectively. A relator transformation $(A, \Phi) \rightarrow (B, \Psi)$ over (\mathbf{B}, \mathbf{F}) is given by a morphism $x: A \rightarrow B$ of the fiber \mathbf{F}_n together with a relator transformation $\Phi \rightarrow \Psi$ whose associated transformation $F \rightarrow G$ is induced by the morphism x by composition.*

Proposition 7.8 thus extends to:

Theorem 7.13 *Let (\mathbf{B}, \mathbf{F}) be an *iml*-category. Let the morphisms $k \rightarrow n$ of a new base category \mathbf{B}' be n -tuples of relators of k arguments over (\mathbf{B}, \mathbf{F}) . Then the indexed category $(\mathbf{B}', \tilde{\mathbf{F}})$ whose fiber over n is the category of relators of n arguments and relator transformations over (\mathbf{B}, \mathbf{F}) is an *iml*-category. Furthermore, the canonical forgetful functor $(\mathbf{B}', \tilde{\mathbf{F}}) \rightarrow (\mathbf{B}, \mathbf{F})$ is an *iml*-representation.*

Proof. Let us outline the main points. We shall actually show that when dealing with relators and relator transformations over (\mathbf{B}, \mathbf{F}) , both of which are defined as certain pairs, in the left coordinate one may work in (\mathbf{B}, \mathbf{F}) and in the right coordinate one may work in the *iml*-category of relators and relator transformations on ccc \mathbf{F}_0 given in Proposition 7.8.

First, observe that Definition 7.11 is stable under composition: if (A, Φ) is a relator of n arguments over (\mathbf{B}, \mathbf{F}) and $(B_1, \Psi_1), \dots, (B_n, \Psi_n)$ relators of k arguments over (\mathbf{B}, \mathbf{F}) , then $(A \circ \langle B_1, \dots, B_n \rangle, \Phi \circ \langle \Psi_1, \dots, \Psi_n \rangle)$ is a relator of k arguments over (\mathbf{B}, \mathbf{F}) . Here the n -tuple $\langle B_1, \dots, B_n \rangle$ is the morphism $k \rightarrow n$ in the given base category \mathbf{B} with products. In this way each morphism $k \rightarrow n$ of the new base category \mathbf{B}' induces a mapping from relators of n arguments to relators of k arguments, *i.e.*, from the objects of the new fiber $(\tilde{\mathbf{F}})_n$ over n to

the objects of the new fiber $(\tilde{\mathbf{F}})_k$ over k . This mapping will be the object part of the required functor $(\tilde{\mathbf{F}})_n \rightarrow (\tilde{\mathbf{F}})_k$.

Similarly, let $(x, \tau): (A, \Phi) \rightarrow (A', \Phi')$ be a relator transformation between relators of n arguments over (\mathbf{B}, \mathbf{F}) and let $k \rightarrow n$ be a morphism in the new base category \mathbf{B}' given by an n -tuple $\langle (B_1, \Psi_1), \dots, (B_n, \Psi_n) \rangle$ of relators of k arguments over (\mathbf{B}, \mathbf{F}) . We shall define the associated relator transformation between relators of k arguments over (\mathbf{B}, \mathbf{F}) . Let $a: k \rightarrow n$ be the morphism in the starting base category \mathbf{B} given by $\langle B_1, \dots, B_n \rangle$. Let $D = \mathbf{F}(a)(A) = A \circ a$, $D' = \mathbf{F}(a)(A') = A' \circ a$, and $y = \mathbf{F}(a)(x)$. Let $\Theta = \Phi \circ \langle \Psi_1, \dots, \Psi_n \rangle$, $\Theta' = \Phi' \circ \langle \Psi_1, \dots, \Psi_n \rangle$, and $\sigma = \tau \circ \langle \Psi_1, \dots, \Psi_n \rangle$, as defined in the *iml*-category of relators on \mathbf{F}_0 . Then $(y, \sigma): (D, \Theta) \rightarrow (D', \Theta')$ is the required relator transformation between relators of k arguments over (\mathbf{B}, \mathbf{F}) . The reader will readily check that this assignment is functorial, *i.e.*, it preserves composition of relator transformations.

We also verify that each new fiber $(\tilde{\mathbf{F}})_n$ is a ccc in such way that the ccc structure is preserved under composition. Finite products in $(\tilde{\mathbf{F}})_n$ are clearly given coordinatewise and hence preserved under composition. Let us calculate $(A, \Phi) \Rightarrow (B, \Psi)$ in $(\tilde{\mathbf{F}})_n$. Here A, B are objects in \mathbf{F}_n and Φ, Ψ are relators of n arguments on the ccc \mathbf{F}_0 , whose object maps are induced by composition by A, B , respectively. Note that for any morphism $C: 0 \rightarrow n$ in the given base category \mathbf{B} , $(A \Rightarrow B) \circ C = (A \circ C) \Rightarrow (B \circ C)$ because $\mathbf{F}(C): \mathbf{F}_n \rightarrow \mathbf{F}_0$ is a ccc representation such that $\mathbf{F}(C)(D) = D \circ C$ for any object D in the ccc \mathbf{F}_n . But this means exactly that $A \Rightarrow B$ induces the object map of $\Phi \Rightarrow \Psi$ as determined in the ccc of relators of n arguments on ccc \mathbf{F}_0 , see Proposition 7.8.

Finally, erasing the right coordinate yields both a functor $L: \mathbf{B}' \rightarrow \mathbf{B}$ that preserves finite products so that $L(n) = n$, as well as a natural transformation $l: \tilde{\mathbf{F}} \rightarrow (\mathbf{F} \circ L)$ such that for each n , $l_n: (\tilde{\mathbf{F}})_n \rightarrow \mathbf{F}_n$ is a ccc representation. ■

Example 7.14 The discussion of the so-called total objects in the Appendix D of [Gir86] provides an interesting special case of Theorem 7.13. The starting *iml*-category (\mathbf{B}, \mathbf{F}) is given in Example 7.3. Object maps of the relevant relators on the ccc Qd of qualitative domains and stable maps are precisely the object maps of functors from the category Qd_{emb} of qualitative domains and embeddings to Qd_{emb} that preserve pullbacks and directed colimits. For any type σ in n type variables, the free *iml*-representation in (\mathbf{B}, \mathbf{F}) assigns to σ a functor $A: (Qd_{emb})^n \rightarrow Qd_{emb}$ that preserves pullbacks and directed colimits. By Theorem 7.13, the canonical forgetful functor from the *iml*-category of relators over (\mathbf{B}, \mathbf{F}) to (\mathbf{B}, \mathbf{F}) is an *iml*-representation. Furthermore, because free *iml*-representation (of the term *iml*-category) in any given *iml*-category is unique, the unique free *iml*-representation in (\mathbf{B}, \mathbf{F}) must factor through the free *iml*-representation to the *iml*-category of relators over (\mathbf{B}, \mathbf{F}) . Therefore this latter representation assigns to type σ a relator of n arguments over (\mathbf{B}, \mathbf{F}) that must be of the form (A, Φ) . In this particular case, this means simply that the object map of A is the object map of the relator Φ of n arguments on Qd . The total elements of A , defined in the Appendix D of [Gir86], are in our terminology those morphisms $1 \rightarrow A$ in \mathbf{F}_n that induce relator transformations $1 \rightarrow \Phi$. Thus the first order part of Theorem D.1 in [Gir86] follows from our Theorem 7.13. ■

Theorem 7.13 also specializes to the m -ary case, by working over the m -fold power of a given *iml*-category (defined below Proposition 7.8). For instance, a binary relator over an *iml*-category (\mathbf{B}, \mathbf{F}) is given by an object A of the fiber \mathbf{F}_1 , together with a unary relator on $\mathbf{F}_0 \times \mathbf{F}_0$, whose object map is induced by the object $\langle A, A \rangle$ of the fiber $(\mathbf{F} \times \mathbf{F})_1$ by composition.

Let us also mention that Theorem 7.13 may be generalized along the lines of Proposition 7.9. The reader will note that the additional condition in Proposition 7.9 that the canonical ccc-representation $\mathbf{D} \rightarrow \mathbf{C}$ is surjective on objects may now be omitted because the object map of a \mathbf{D} -relator over a given *iml*-category (\mathbf{B}, \mathbf{F}) is given explicitly by an object of the fiber \mathbf{F}_1 .

Theorem 7.15 Let (\mathbf{B}, \mathbf{F}) be an *iml*-category. Let \mathbf{C}' be a cartesian closed categories with equalizers. Let $\mathbf{G}: \mathbf{F}_0 \rightarrow \mathbf{C}'$ be a functor that preserves finite products up to isomorphism. Let *ccc* \mathbf{D} be a full subcategory of the comma category $(\mathbf{C}' \downarrow \mathbf{G})$, so that the restriction of the canonical *ccc*-representation $(\mathbf{C}' \downarrow \mathbf{G}) \rightarrow \mathbf{C}$ to \mathbf{D} is a *ccc*-representation. Let the morphisms $k \rightarrow n$ of the new base category \mathbf{B}' be n -tuples of \mathbf{D} -relators of k arguments over (\mathbf{B}, \mathbf{F}) . Then the indexed category $(\mathbf{B}', \tilde{\mathbf{F}})$ whose fiber over n is the category of \mathbf{D} -relators of n arguments and \mathbf{D} -relator transformations over (\mathbf{B}, \mathbf{F}) is an *iml*-category. Furthermore, the canonical forgetful functor $(\mathbf{B}', \tilde{\mathbf{F}}) \rightarrow (\mathbf{B}, \mathbf{F})$ is an *iml*-representation.

Example 7.16 Let us give an example of the binary case of Theorem 7.15. Recalling Section 6.3, let \mathbf{D} be the full sub-*ccc* of the comma category $(Per \downarrow \mathbf{H})$ whose objects are saturated binary relations on pers, *i.e.*, objects of the form $\langle S, P, Q \rangle$, where S is a saturated binary relation from the domain of $per\ P$ to the domain of $per\ Q$. Let the starting *iml*-category (\mathbf{B}, \mathbf{F}) be the *iml*-category PER discussed in Example 7.4, a sub-*iml*-category of Per_{iml} . In particular, the fiber over 0 in PER is the *ccc* Per of pers and per morphisms. A (binary) \mathbf{D} -relator over PER is simply a (binary) \mathbf{D} -relator on Per , *i.e.*, it consists of a mapping F from pers to pers and a mapping $\Phi: Obj\ \mathbf{D} \rightarrow Obj\ \mathbf{D}$ such that $\Phi(S, P, Q) = (S', F(P), F(Q))$. However, a \mathbf{D} -relator transformation over PER is not simply a \mathbf{D} -relator transformation on Per , because it must be given by a realizable family of per morphisms. Indeed, let $\langle F, \Phi \rangle$ and $\langle G, \Psi \rangle$ be \mathbf{D} -relators over PER . A \mathbf{D} -relator transformation $t: \langle F, \Phi \rangle \rightarrow \langle G, \Psi \rangle$ over PER is given by a partial recursive function f that names each per morphism $t_P: F(P) \rightarrow G(P)$, with P ranging over pers, and such that for every saturated relation S from the domain of P to the domain of Q and the associated saturated relations S' and S'' given by Φ and Ψ , respectively, it is the case that the composite relation $t_Q \circ S'$ is included in the relation $S'' \circ t_P$. (Each per morphism induces a saturated relation.) In other words, for all pers P and Q and all numbers n in the domain of $F(P)$ and k in the domain of $F(Q)$, $n\ S'\ k$ implies $f(n)\ S''\ f(k)$. Thus the first order part of the Soundness Theorem in Section 4.7 of [BFS90] is a special case of our Theorem 7.15. ■

Example 7.17 Another example of this more general situation is the *iml*-category of *pcpo*-relators whose associated object maps are not just arbitrary maps from *pcpos* to *pcpos*, but functors from the category $Pcpo_{emb}$ of *pcpos* and embedding-projection pairs to $Pcpo_{emb}$. Here the fiber \mathbf{F}_n of the starting *iml*-category is the *ccc* whose objects are functors $(Pcpo_{emb})^n \rightarrow Pcpo_{emb}$ and the morphisms $t: F \rightarrow G$ of the fiber \mathbf{F}_n are certain families of continuous maps $t_P: F(P) \rightarrow G(P)$, where P ranges over n -tuples of *pcpos*, see [CGW89]. ■

References

- [ACC93] M. Abadi, L. Cardelli, and P.-L. Curien. Formal parametric polymorphism. In *Proc. 20-th ACM Symposium on Principles of Programming Languages*, 1993.
- [AbJ91] S. Abramsky and T.P. Jensen. A relational approach to strictness analysis for higher-order polymorphic functions. In *Proc. 18-th ACM Symposium on Principles of Programming Languages*, 1991.
- [BFS90] E.S. Bainbridge, P.J. Freyd, A. Scedrov, and P.J. Scott. Functorial Polymorphism. *Theoretical Computer Science*, 70:35–64, 1990. Corrigendum *ibid.*, 71:431, 1990.
- [BTC88] V. Breazu-Tannen and T. Coquand. Extensional models for polymorphism. *Theoretical Computer Science*, 59:85–114, 1988.
- [BMM90] K. B. Bruce, A. R. Meyer, and J. C. Mitchell. The semantics of second-order lambda calculus. *Information and Computation*, 85(1):76–134, 1990. Reprinted in *Logical Foundations of Functional Programming*, ed. G. Huet, Addison-Wesley (1990) 213–273.

- [CMS91] L. Cardelli, J.C. Mitchell, S. Martini, and A. Scedrov. An extension of system F with Subtyping. To appear in *Information and Computation*. Extended abstract in T.Ito and A.R. Meyer (eds.), *Theoretical Aspects of Computer Software*, pages 750–770. Springer-Verlag LNCS 526, 1991.
- [CGW89] T. Coquand, C.A. Gunter, and G. Winskel. Domain theoretic models of polymorphism. *Information and Computation*, 81:123–167, 1989.
- [CP92] R.L. Crole and A.M. Pitts. New foundations for fixpoint computations: FIX-hyperdoctrines and the FIX-logic. *Information and Computation*, 98:171–210, 1992.
- [Fre93] P.J. Freyd. Structural polymorphism. *Theoretical Computer Science*, 115:107–129, 1993.
- [FrS90] P.J. Freyd and A. Scedrov. *Categories, Allegories*. Mathematical Library, North-Holland, 1990.
- [FRR92] P.J. Freyd, E.P. Robinson, and G. Rosolini. Functorial parametricity. In *Proc. 7-th Annual IEEE Symposium on Logic in Computer Science*, pages 444–452, 1992.
- [FRS92] P.J. Freyd, E.P. Robinson, and G. Rosolini. Dinaturality for free. In M.P. Fourman, P.T. Johnstone, and A.M. Pitts, eds., *Applications of Categories in Computer Science*, pages 107–118. London Math. Soc. Lecture Note Series, vol. 177, Cambridge Univ. Press, 1992.
- [Gir86] J.-Y. Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986. Reprinted in *Logical Foundations of Functional Programming*, ed. G. Huet, Addison-Wesley (1990) 87–126.
- [GLT89] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1989.
- [GSS91] J.-Y. Girard, A. Scedrov, and P.J. Scott. Normal forms and cut-free proofs as natural transformations. In: Y.N. Moschovakis, editor, *Logic from Computer Science, Proc. M.S.R.I. Workshop, Berkeley, 1989*. M.S.R.I. Series, Springer-Verlag, 1991.
- [Has90] R. Hasegawa. Categorical data types in parametric polymorphisms. Manuscript, 1990.
- [Has91] R. Hasegawa. Parametricity of extensionally collapsed term models of polymorphism and their categorical properties. In T.Ito and A.R. Meyer (eds.), *Theoretical Aspects of Computer Software*, pages 495–512. Springer-Verlag LNCS 526, 1991.
- [KR77] A. Kock, G.E. Reyes. Doctrines in categorical logic. In J. Barwise, editor, *Handbook of Mathematical Logic*, pages 283–313. North-Holland, 1977.
- [Laf88] Y. Lafont. *Logiques, Categories & Machines*. Thèse de Doctorat, Université Paris VII, 1988.
- [LS86] J. Lambek and P.J. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge studies in advanced mathematics 7, Cambridge University Press, 1986.
- [MaR92] Q. Ma and J.C. Reynolds. Types, abstraction, and parametric polymorphism, Part 2. In S. Brookes *et al.*, editors, *Mathematical Foundations of Programming Semantics, Proceedings 1991*, pages 1–40. Springer-Verlag LNCS 598, 1992.
- [McL71] S. Mac Lane. *Categories for the Working Mathematician*. Graduate Texts in Mathematics, Springer-Verlag, 1971.
- [Mai91] H. Mairson. Outline of a proof theory of parametricity. In *Proc. 5-th Intern. Symp. on Functional Programming and Computer Architecture*, 1991.
- [MS76] R.E. Milne and C. Strachey. *A theory of programming language semantics*. Chapman and Hall, London, and Wiley, New York, 1976.

- [Mit86] J.C. Mitchell. A type-inference approach to reduction properties and semantics of polymorphic expressions. In *ACM Conference on LISP and Functional Programming*, pages 308–319, August 1986. Revised version in *Logical Foundations of Functional Programming*, ed. G. Huet, Addison-Wesley (1990) 195–212.
- [Mit88] J.C. Mitchell. Polymorphic type inference and containment. *Information and Computation*, 76(2/3):211–249, 1988. Reprinted in *Logical Foundations of Functional Programming*, ed. G. Huet, Addison-Wesley (1990) 153–194.
- [Mit90] J.C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 365–458. North-Holland, 1990.
- [MM85] J.C. Mitchell and A.R. Meyer. Second-order logical relations. In *Logics of Programs*, pages 225–236. Springer-Verlag LNCS 193, June 1985.
- [MM91] J.C. Mitchell and E. Moggi. Kripke-style models for typed lambda calculus. *Annals Pure Appl. Logic*, 51:99–124, 1991.
- [MS89] J.C. Mitchell and P.J. Scott. Typed lambda calculus and cartesian closed categories. In J. W. Gray and A. Scedrov, editors, *Categories in Computer Science and Logic, Contemporary Math.*, vol. 92, pages 301–316. Amer. Math. Society, 1989.
- [Mgg89] E. Moggi. Computational lambda calculus and monads. In *Proc. 4th IEEE Symposium on Logic in Computer Science*, pages 14–23, IEEE Computer Society Press, 1989.
- [Mog91] E. Moggi. A category-theoretic account of program modules. *Math. Structures in Computer Science*, 1(1):103–139, 1991.
- [OHT93] P.W. O’Hearn and R.D. Tennent. Relational parametricity and local variables. In *Proc. 20-th ACM Symposium on Principles of Programming Languages*, 1993.
- [Pit87] A.M. Pitts. Polymorphism is set-theoretic, constructively. In *Category Theory and Computer Science, Proceedings Edinburgh, 1987*, pages 12–39. Springer-Verlag LNCS volume 283, 1987.
- [Plo80] G.D. Plotkin. Lambda definability in the full type hierarchy. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 363–373. Academic Press, 1980.
- [Rey74] J.C. Reynolds. On the relation between direct and continuation semantics. In *Second Colloq. Automata, Languages and Programming*, pages 141–156. Springer-Verlag LNCS, 1974.
- [Rey83] J.C. Reynolds. Types, abstraction, and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing ’83*, pages 513–523. North-Holland, 1983.
- [ScS82] A. Scedrov and P.J. Scott. A note on the Friedman slash and Freyd covers. In A.S. Troelstra and D. van Dalen, editors, *The L. E. J. Brouwer Symposium*, pages 443–452. North-Holland, 1982.
- [See87] R.A.G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *Journal of Symbolic Logic*, 52:969–989, 1987.
- [Sta85] R. Statman. Logical relations and the typed lambda calculus. *Information and Control*, 65:85–97, 1985.
- [Str67] C. Strachey. Fundamental concepts in programming languages. Unpublished lecture notes, International Summer School in Computer Programming, Copenhagen, August, 1967.
- [Wad89] P. Wadler. Theorems for free! In *4th Internat. Symp. on Functional Programming Languages and Computer Architecture, London*, pages 347–359, Assoc. for Comp. Machinery, 1989.