Relating Computational Effects by $\top\top$-Lifting

Shin-ya Katsumata

Research Institute for Mathematical Sciences
Kyoto University, Kyoto, 606-8502, Japan
sinya@kurims.kyoto-u.ac.jp

Abstract. We consider the problem of establishing a relationship between two interpretations of base type terms of a λ_c-calculus with algebraic operations. We show that the given relationship holds if it satisfies a set of natural conditions. We apply this result to comparing interpretations of new name creation by two monads: Stark's new name creation monad [25] and a global counter monad.

1 Introduction

Suppose that two monadic semantics $\mathcal{A}_1, \mathcal{A}_2$ are given to a call-by-value functional language, and each semantics $\mathcal{A}_i$ interprets a base type b by a set $A_i b$ and computational effects by a monad $\mathcal{T}_i$. After comparing these semantics, you find a relationship $Vb \subseteq A_1 b \times A_2 b$ between base type values, and also a relationship $Cb \subseteq T_1 A_1 b \times T_2 A_2 b$ between base type computations. We then consider the following problem:

> For any well-typed term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$ and $(v_i, w_i) \in Vb_i$, do we have $(\mathcal{A}_1[\![M]\!](v_1, \cdots, v_n), \mathcal{A}_2[\![M]\!](w_1, \cdots, w_n)) \in Cb$?

We name this problem *effect simulation problem* and tackle it under the situation where 1) the call-by-value functional language is the simply typed λ_c-calculus with products, coproducts, effect-free constants and *algebraic operations* [22], and 2) the underlying category of a semantics is a bi-CCC with a strong monad. We show that the answer of the effect simulation problem is "yes" if I) monad units η_1, η_2 map pairs in V to pairs in C, II) V is closed under effect-free constants and III) C is closed under algebraic operations in the λ_c-calculus. We prove this by extending Mitchell's representation independence proof [17] with logical relations for monads constructed by *categorical* $\top\top$*-lifting [11]*, which is a semantic formulation of the *leapfrog method* introduced by Lindley and Stark [14, 15]. The point of this result is the generality: it holds with any monad, algebraic operation and relation V and C. We demonstrate the flexibility of our solution by showing a general comparison theorem of two monadic semantics related by strong monad morphisms (Section 4), and comparing two interpretations of new name creation by Stark's new name creation monad [25] and a global state monad (Section 5). In Section 6 we consider the effect simulation problem under the presence of recursive functions.

Preliminary A bold letter, such as $\boldsymbol{x}$, abbreviates a sequence $x_1 \cdots x_n$. The length of the sequence is written by $|\boldsymbol{x}|$. We regard every set as a discrete category. We write $\Rightarrow$ and λ for exponentials and currying operators in a CCC. A *bi-CCC* is a CCC with finite coproducts. For a monad (T, η, μ) and a morphism $f : I \to TJ$, we write $f^{\#}$ for $\mu_J \circ Tf$.

2 The λ_c-Calculus with Algebraic Operations

We adopt the simply-typed λ_c-calculus with effect-free constants and *algebraic operations* [22] as an idealised call-by-value functional language. In Section 6 we add recursive functions.

Let B be the set of base types. We use b (and its variants) to range over B. An effect-free constant in the calculus takes a value of type $b_1 \times \ldots \times b_n$ and returns a value of type $\sum_{i=1}^{m} \prod_{j=1}^{l_i} b'_{ij}$. We encode this type information by an element in $B^* \times (B^*)^*$. Examples of effect-free constants are the equality predicate: $\mathbf{eq}^b : b \times b \to 1+1$ and the division function with a reminder and error: $\mathbf{div} : \mathit{nat} \times \mathit{nat} \to \mathit{nat} \times \mathit{nat} + 1$. An algebraic operation in the calculus has the form $o(\boldsymbol{x}_1^{\boldsymbol{b}_1}.M_1, \ldots, \boldsymbol{x}_n^{\boldsymbol{b}_n}.M_n)$. The variables $\boldsymbol{x}_i$ are bound in each subterm M_i. The number of subterms of the algebraic operation and the types of the bound variables in each subterm are specified by an element in $(B^*)^*$ called *arity*. For instance, an algebraic operation of arity $(\epsilon, b_1 b_2)$ looks like $o(M_1, x_1^{b_1} x_2^{b_2}.M_2)$. The symbols of effect-free constants and algebraic operations added to the calculus are specified by a *signature* Σ over B. It assigns a set of symbols to each element in $B^* \times (B^*)^* + (B^*)^*$. We assume that $\Sigma(x)$ is disjoint with each other. We write $\Sigma^{\boldsymbol{b} \to a}$ and Σ^a for the sets $\Sigma(\iota_1(\boldsymbol{b}, a))$ and $\Sigma(\iota_2(a))$, respectively.

We define the computational lambda calculus $\lambda_c(B, \Sigma)$ over the set B of base types and a signature Σ over B. The types and terms of $\lambda_c(B, \Sigma)$ are defined as follows:

$$\begin{aligned}
\rho &::= b \mid \rho \Rightarrow \rho \mid 1 \mid \rho \times \rho \mid 0 \mid \rho + \rho \\
M &::= x \mid \lambda x^{\rho}\,.\,M \mid MM \mid * \mid (M, M) \mid \pi_i(M) \mid \bot_\rho \mid \\
&\qquad \iota_i(M) \mid \delta(M, x.M, x.M) \mid \mathsf{let}\ x = M\ \mathsf{in}\ M \mid c\ M \mid o_\rho(\boldsymbol{x}^{\boldsymbol{b}}.M, \cdots, \boldsymbol{x}^{\boldsymbol{b}}.M)
\end{aligned}$$

where c, o ranges over the set of symbols for effect-free constants and algebraic operations specified by Σ. The type system of $\lambda_c(B, \Sigma)$ extends the one for the simply typed lambda calculus with products and sums (see e.g. [18]). The term $\bot_\rho$ denotes the unique term of type $0 \Rightarrow \rho$, and the δ-term denotes the sum elimination. The typing rules for the last three terms are the following:

$$\frac{\Gamma \vdash M : \rho \quad \Gamma, x : \rho \vdash N : \sigma}{\Gamma \vdash \mathsf{let}\ x = M\ \mathsf{in}\ N : \sigma} \quad \frac{\Gamma \vdash M : b_1 \times \cdots \times b_n \quad c \in \Sigma^{b_1 \cdots b_n \to (\boldsymbol{b}'_1, \cdots, \boldsymbol{b}'_m)}}{\Gamma \vdash c\ M : \sum_{i=1}^{m} \prod_{j=1}^{|\boldsymbol{b}'_i|} b'_{ij}}$$

$$\frac{\Gamma, x_{i1} : b_{i1}, \cdots, x_{i|\boldsymbol{b}_i|} : b_{i|\boldsymbol{b}_i|} \vdash M_i : \rho \quad 1 \le i \le n \quad o \in \Sigma^{(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)}}{\Gamma \vdash o_\rho(\boldsymbol{x}_1^{\boldsymbol{b}_1}.M_1, \cdots, \boldsymbol{x}_n^{\boldsymbol{b}_n}.M_n) : \rho} \quad .$$

We move to the semantics of the $\lambda_c(B, \Sigma)$-calculus.

Definition 1 ([22]). *Let $\mathcal{T} = (T, \eta, \mu, \theta)$ be a strong monad over a CCC $\mathbb{C}$ and $Z \in \mathbb{C}$. We write $st^Z_{I,J} : I \times (Z \Rightarrow J) \to Z \Rightarrow (I \times J)$ for the strength of $Z \Rightarrow -$. A* Z-ary algebraic operation *for $\mathcal{T}$ is a natural transformation $\alpha_I : Z \Rightarrow TI \to TI$ such that*

$$\alpha_{I \times J} \circ Z \Rightarrow \theta_{I,J} \circ st^Z_{I,J} = \theta_{I,J} \circ I \times \alpha_J, \qquad \mu_I \circ \alpha_{TI} = \alpha_I \circ Z \Rightarrow \mu_I.$$

Let $A : B \to \mathbb{C}$ be a functor to a bi-CCC $\mathbb{C}$. We extend A to a functor $A' : (B^*)^* \to \mathbb{C}$ by $A'(\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n) = \sum_{i=1}^{n} \prod_{j=1}^{|\boldsymbol{b}_i|} Ab_{ij}$. Below we simply write A for A'.

Definition 2. *A* $\lambda_c(B,\Sigma)$-structure *is a tuple* $\mathcal{A} = (\mathbb{C}, \mathcal{T}, A, k, \alpha)$ *where* $\mathbb{C}$ *is a bi-CCC,* $\mathcal{T}$ *is a strong monad on* $\mathbb{C}$*,* A *is a functor of type* $B \to \mathbb{C}$*,* k *assigns to each* $c \in \Sigma^{\boldsymbol{b} \to a}$ *a morphism* $kc : \prod_{i=1}^{|\boldsymbol{b}|} Ab_i \to Aa$*, and* α *assigns to each* $o \in \Sigma^{a}$ *an* Aa*-ary algebraic operation* αo *for* $\mathcal{T}$*.*

We write $\mathcal{A}_1 \times \mathcal{A}_2$ to mean the evident product of two $\lambda_c(B,\Sigma)$-structures $\mathcal{A}_1, \mathcal{A}_2$. Each $\lambda_c(B,\Sigma)$-structure $\mathcal{A}$ determines a natural interpretation $\mathcal{A}[\![-]\!]$ of types and well-typed terms of $\lambda_c(B,\Sigma)$ in the category of $\mathcal{A}$ (see e.g. CBV Translation in [1]). Effect-free constants and algebraic operations are interpreted as follows:

$$\mathcal{A}[\![c\ M]\!] = T(kc) \circ \mathcal{A}[\![M]\!]$$
$$\mathcal{A}[\![o_\rho(\boldsymbol{x}_1.M_1, \cdots, \boldsymbol{x}_n.M_n)]\!] = \alpha o_{\mathcal{A}[\![\rho]\!]} \circ \langle \lambda^{|\boldsymbol{x}_1|}(\mathcal{A}[\![M_1]\!]), \cdots, \lambda^{|\boldsymbol{x}_n|}(\mathcal{A}[\![M_n]\!])\rangle.$$

3 Effect Simulation Problem

The main problem we consider is the *effect simulation problem*. We first introduce a set-theoretic version of it. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be $\lambda_c(B,\Sigma)$-structures over **Set**. A *simulation* between $\mathcal{A}_1$ and $\mathcal{A}_2$ is a pair (V, C) where V and C are B-indexed families of binary relations such that $Vb \subseteq \mathcal{A}_1[\![b]\!] \times \mathcal{A}_2[\![b]\!]$ and $Cb \subseteq T_1\mathcal{A}_1[\![b]\!] \times T_2\mathcal{A}_2[\![b]\!]$; we call V and C *value simulation* and *computation simulation*, respectively. The effect simulation problem is the following:

> Suppose that a simulation (V, C) between $\mathcal{A}_1$ and $\mathcal{A}_2$ is given. Then for any well-typed term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$ and $(v_i, w_i) \in Vb_i$, do we have $(\mathcal{A}_1[\![M]\!](\boldsymbol{v}), \mathcal{A}_2[\![M]\!](\boldsymbol{w})) \in Cb$?

Example 1. Let B be the set of base types and Σ be the signature that specifies only two algebraic operation symbols: $\mathsf{null} \in \Sigma^{\epsilon}$ and $\mathsf{join} \in \Sigma^{(\epsilon,\epsilon)}$. We regard $\lambda_c(B,\Sigma)$ as a call-by-value functional language with constructors for nondeterministic computation. The standard semantics of $\lambda_c(B,\Sigma)$ is given by the $\lambda_c(B,\Sigma)$-structure $\mathcal{A}_1 = (\mathbf{Set}, \mathcal{T}_p, A, k, \alpha_1)$, where $\mathcal{T}_p$ is the finite powerset monad, and α_1 assigns algebraic operations by $\alpha_1(\mathsf{null}) = \emptyset$ and $\alpha_1(\mathsf{join})(x, y) = x \cup y$. On the other hand, one may represent nondeterministic choices by finite lists instead of finite sets. This representation corresponds to the semantics of $\lambda_c(B,\Sigma)$ by the $\lambda_c(B,\Sigma)$-structure $\mathcal{A}_2 = (\mathbf{Set}, \mathcal{T}_m, A, k, \alpha_2)$ where $\mathcal{T}_m$ is the free monoid monad, and α_2 assigns algebraic operations by $\alpha_2(\mathsf{null}) = \epsilon$ (the empty list) and $\alpha_2(\mathsf{join})(x, y) = x \cdot y$ (the concatenation of two lists).

We expect that for any well-typed term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$ and $v_i \in Ab_i$, the denotation $\mathcal{A}_1[\![M]\!](\boldsymbol{v})$ gives the set of possible choices listed up in $\mathcal{A}_2[\![M]\!](\boldsymbol{v})$. That is, we expect that the answer to the effect simulation problem with value simulation $Vb = \{(v, v) \mid v \in Ab\}$ and computation simulation $Cb = \{(X, l) \in T_pAb \times T_mAb \mid X = \text{the set of elements in } l\}$ is yes.

We move on to the general situation where the underlying categories are other than **Set**. To formulate the concept of relation between two objects from different categories, we first formulate the concept of predicate over objects in arbitrary category in terms of *fibrational category theory*, then derive the concept of relation as predicates over product categories. Formulating logical relations in fibrational category theory is advocated by Hermida [9], which subsumes *subscone* [19]; see Example 3-1.

Here we give brief definitions of fibration and related concepts; see [10] for the complete detail. Let $p : \mathbb{E} \to \mathbb{B}$ be a functor. We say that $X \in \mathbb{E}$ is *above* $I \in \mathbb{B}$ if $pX = I$. We use the same word for morphisms in $\mathbb{E}$ and $\mathbb{B}$. A *fibre category* over $I \in \mathbb{B}$ is the subcategory of $\mathbb{E}$ consisting of objects above $I \in \mathbb{B}$ and morphisms above id_I. We next assume that p is faithful. One easily sees that $\mathbb{E}_I$ is a preorder. In this situation, we regard $\mathbb{E}_I$ as the preorder of predicates on I. For $X, Y \in \mathbb{E}$, by $f : X \dot{\to} Y$ we mean that $f \in \mathbb{B}(pX, pY)$ and there exists a (necessarily unique) morphism $\dot{f} : X \to Y$ above f. We call $\dot{f}$ the *witness* of $f : X \dot{\to} Y$. The statement $f : X \dot{\to} Y$ means that f is a morphism that sends elements satisfying X to those satisfying Y.

Definition 3. *A* partial order bifibration with fibrewise small products *is a faithful functor $p : \mathbb{E} \to \mathbb{B}$ such that:*

(Partial Order) *Each fibre category is a partial order.*
(Fibration) [1] *For any $I \in \mathbb{B}$, $Y \in \mathbb{E}$ and $f \in \mathbb{B}(I, pY)$, there exists $X \in \mathbb{E}$ above I such that $f : X \dot{\to} Y$ and the following property holds: for any $Z \in \mathbb{E}$ and $h : pZ \to I$, $f \circ h : Z \dot{\to} Y$ implies $h : Z \dot{\to} X$. This property and $\mathbb{E}_I$ being a partial order imply that X is unique; hence we write it by f^*Y, and the witness of $f : f^*Y \dot{\to} Y$ by $\overline{f}Y$. Furthermore, for any $f \in \mathbb{B}(I, J)$, the mapping $Y \in \mathbb{E}_J \mapsto f^*Y \in \mathbb{E}_I$ extends to a functor $f^* : \mathbb{E}_J \to \mathbb{E}_I$. We call it* inverse image functor. *Intuitively, f^*Y corresponds to the predicate $\{i \in I \mid f(i) \in Y\}$ on I.*
(Bi–) *Each inverse image functor has a left adjoint called* direct image functor.
(Fibrewise Small Products) *Each fibre category has small products and inverse image functors (necessarily) preserve them.*

Definition 4. *A* category for logical relations *over a bi-CCC $\mathbb{C}$ is a partial order bifibration $p : \mathbb{E} \to \mathbb{C}$ with fibrewise small products*[2] *such that $\mathbb{E}$ is a bi-CCC and p strictly preserves the bi-cartesian closed structure. Notational convention: we write the bi-cartesian closed structure on $\mathbb{E}$ with dots on the top, like $\dot{\Rightarrow}, \dot{1}, \dot{\times}, \dot{0}, \dot{+}, \dot{ev}, \dot{\lambda}, \dot{\langle} -, - \dot{\rangle}, \cdots$.*

In [10, Section 9.2], it is discussed when a fibration becomes a category for logical relations. Particularly, the subobject fibration of any presheaf category is a category for logical relations. Below we see a special case: the subobject fibration of **Set**.

Example 2. [10, Chapter 0] We define the category **Pred** by the following data: an object in **Pred** is a pair (X, I) where X is a subset of I and a morphism from (X, I) to (Y, J) is a function $f : I \to J$ such that for any $i \in X$, $f(i) \in Y$. This category is equivalent to the category of subobjects of **Set**. The evident forgetful functor π : **Pred** $\to$ **Set** is a bifibration; the inverse image functor for a function $f : I \to J$ is given by $f^*(Y, J) = (\{x \mid f(x) \in Y\}, I)$, and it has a left adjoint given by $f_*(X, I) = (\{f(x) \mid x \in X\}, J)$. The fibre category $\mathbf{Pred}_I$ is the poset $(2^I, \subseteq)$, and the intersection gives small products. Therefore π is a partial order bifibration with fibrewise small products.

The category **Pred** has a bi-cartesian closed structure that is strictly preserved by π [10, Exercise 9.2.1]. The exponential is given by $(X, I) \dot{\Rightarrow} (Y, J) = (\{f \mid \forall x \in X \,.\, f(x) \in Y\}, I \Rightarrow J)$. To summarise, π is a category for logical relations.

[1] This definition of fibration exploits the assumption that p is faithful. The concept of fibration is actually defined on arbitrary functor [10, Definition 1.1.3].
[2] We actually only use fibrewise products up to the cardinality of the set B of base types.

Proposition 1. *Let $p : \mathbb{E} \to \mathbb{B}$ be a category for logical relations, $\mathbb{C}$ be a bi-CCC and $F : \mathbb{C} \to \mathbb{B}$ be a finite-product preserving functor. Then the pullback $F^*(p) : F^*(\mathbb{E}) \to \mathbb{C}$ of p along F is a category for logical relations.*

Proof. This is a straightforward generalisation of the proof that any subscone is a CCC [19]. We use direct image functors to construct finite coproducts in $F^*(\mathbb{E})$.

Example 3.
1. A *subscone* [19] over a bi-CCC $\mathbb{C}$ is the category obtained by pulling back $\pi : \mathbf{Pred} \to \mathbf{Set}$ along the global element functor $\mathbb{C}(1, -) : \mathbb{C} \to \mathbf{Set}$. The leg from the subscone to $\mathbb{C}$ is a category for logical relations.
2. We pull-back π along the product functor $- \times - : \mathbf{Set}^2 \to \mathbf{Set}$. This yields the category $\mathbf{Rel}$ of binary relations, and the leg $q : \mathbf{Rel} \to \mathbf{Set}^2$ is a category for logical relations.

Having abstracted the concept of predicates in terms of fibrational category theory, we now generalise the effect simulation problem to the following *effect property problem*. Let $\mathcal{A} = (\mathbb{C}, \mathcal{T}, A, k, \alpha)$ be a $\lambda_c(B, \Sigma)$-structure and $p : \mathbb{E} \to \mathbb{C}$ be a category for logical relations. A *property* over $\mathcal{A}$ is a pair (V, C) of functors $V, C : B \to \mathbb{E}$ such that for all base types $b \in B$, Vb is above Ab and Cb is above TAb. The problem is:

> Given a property (V, C) over $\mathcal{A}$, does any well-typed term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$ satisfy $\mathcal{A}[\![M]\!] : Vb_1 \dot{\times} \cdots \dot{\times} Vb_n \dot{\to} Cb$?

An effect simulation problem between two $\lambda_c(B, \Sigma)$-structures $\mathcal{A}_1$ and $\mathcal{A}_2$ is nothing but an effect property problem over $\mathcal{A}_1 \times \mathcal{A}_2$; particularly the set-theoretic one in the beginning of this section uses $q : \mathbf{Rel} \to \mathbf{Set}^2$ as a category for logical relations.

We say that a property (V, C) over a $\lambda_c(B, \Sigma)$-structure $\mathcal{A} = (\mathbb{C}, \mathcal{T}, A, k, \alpha)$ satisfies:

(I) if for all base types $b \in B$, we have $\eta_{Ab} : Vb \dot{\to} Cb$.
(C1) if for all $(\boldsymbol{b}, a) \in B^* \times (B^*)^*$ and effect-free constant symbols $c \in \Sigma^{\boldsymbol{b} \to a}$, we have $kc : \dot{\prod}_{i=1}^{|\boldsymbol{b}|} Vb_i \dot{\to} Va$.
(C2) if for all arities $a \in (B^*)^*$, algebraic operation symbols $o \in \Sigma^a$ and base types $b \in B$, we have $\alpha o_{Ab} : Va \dot{\Rightarrow} Cb \dot{\to} Cb$.

Theorem 1. *Let $\mathcal{A}$ be a $\lambda_c(B, \Sigma)$-structure and (V, C) be a property over $\mathcal{A}$ that satisfies (I), (C1) and (C2). Then for any well-typed term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$, we have $\mathcal{A}[\![M]\!] : Vb_1 \dot{\times} \cdots \dot{\times} Vb_n \dot{\to} Cb$.*

The rest of this section is the proof of the above theorem. The proof extends Mitchell's representation independence [17] using a logical relation with a special care on monads. Let $\mathcal{A} = (\mathbb{C}, \mathcal{T}, A, k, \alpha)$ be a $\lambda_c(B, \Sigma)$-structure and (V, C) be a property over $\mathcal{A}$ that satisfies (I), (C1) and (C2). We aim to construct a $\lambda_c(B, \Sigma)$-structure $\mathcal{D} = (\mathbb{E}, \dot{\mathcal{T}}, V, \dot{k}, \dot{\alpha})$ such that 1) $\dot{T}f, \dot{\eta}_X, \dot{\mu}_X, \dot{\theta}_{X,Y}$ are respectively above $T(pf), \eta_{pX}, \mu_{pX}, \theta_{pX,pY}$, 2) $\dot{k}c$ is above kc, 3) $\dot{\alpha}o_X$ is above αo_{pX} and 4) $\dot{T}Vb \leq Cb$ holds in $\mathbb{E}_{TAb}$.

We construct the strong monad $\dot{\mathcal{T}}$ on $\mathbb{E}$ by *categorical* $\top\top$*-lifting* [11], which is a semantic formulation of Lindley and Stark's $\top\top$-lifting [15, 14]. Let $X \in \mathbb{E}$ be above $I \in \mathbb{C}$. We first define the object $X^{\top\top(Cb)}$ above TI to be the inverse image of $(X \dot{\Rightarrow} Cb) \dot{\Rightarrow} Cb$ along the following morphism $\sigma_I^{\mathcal{T}, Ab} : TI \to (I \Rightarrow TAb) \Rightarrow TAb$:

$$\sigma_I^{\mathcal{T}, Ab} = \lambda(ev^\# \circ \theta_{I \Rightarrow TAb, I} \circ \langle \pi_2, \pi_1 \rangle).$$

This is the strong monad morphism (see Section 4) from $\mathcal{T}$ to the continuation monad $(- \Rightarrow TAb) \Rightarrow TAb$. We then define $\dot{T}X$ by the following fibrewise product:

$$\dot{T}X = \bigwedge_{b \in B} X^{\top\top(Cb)} \left(= \bigwedge_{b \in B} (\sigma_I^{\mathcal{T},Ab})^*((X \mathrel{\dot{\Rightarrow}} Cb) \mathrel{\dot{\Rightarrow}} Cb) \right). \tag{1}$$

Proposition 2 ([11]). *Let $X, Y \in \mathbb{E}$.*

1. *For any morphism f in $\mathbb{C}$, $f : X \mathrel{\dot{\to}} Y$ implies $Tf : \dot{T}X \mathrel{\dot{\to}} \dot{T}Y$.*
2. *We have $\eta_{pX} : X \mathrel{\dot{\to}} \dot{T}X$, $\mu_{pX} : \dot{T}\dot{T}X \mathrel{\dot{\to}} \dot{T}X$ and $\theta_{pX,pY} : X \mathbin{\dot{\times}} \dot{T}Y \mathrel{\dot{\to}} \dot{T}(X \mathbin{\dot{\times}} Y)$.*

From this, for any morphism $\dot{f} : X \to Y$ in $\mathbb{E}$, we define $\dot{T}\dot{f}$ to be the witness of $T(p\dot{f}) : \dot{T}X \mathrel{\dot{\to}} \dot{T}Y$. We similarly define morphisms $\dot{\eta}_X, \dot{\mu}_X, \dot{\theta}_{X,Y}$ in $\mathbb{E}$ to be the witnesses of the statements in Proposition 2-2. Then the tuple $\dot{\mathcal{T}} = (\dot{T}, \dot{\eta}, \dot{\mu}, \dot{\theta})$ forms a strong monad over $\mathbb{E}$ satisfying the condition 1.

From (C1), for each $(\boldsymbol{b}, a) \in B^* \times (B^*)^*$ and $c \in \Sigma^{\boldsymbol{b} \to a}$, we define $\dot{k}c$ to be the witness of $kc : \dot{\prod}_{i=1}^{|\boldsymbol{b}|} Vb_i \mathrel{\dot{\to}} Va$. This defines the component $\dot{k}$ of $\mathcal{D}$ satisfying the condition 2.

We construct the component $\dot{\alpha}$ of $\mathcal{D}$ satisfying the condition 3. We first prove a general fact about algebraic operations for the $\top\top$-lifted monads $\dot{T}$:

Proposition 3. *Let $Z \in \mathbb{C}$, α be an Z-ary algebraic operation for $\mathcal{T}$ and $\dot{Z} \in \mathbb{E}$ be above Z. If $\alpha_{Ab} : \dot{Z} \mathrel{\dot{\Rightarrow}} Cb \mathrel{\dot{\to}} Cb$ holds for all base types $b \in B$, then for any object $X \in \mathbb{E}$, we have $\alpha_{pX} : \dot{Z} \mathrel{\dot{\Rightarrow}} \dot{T}X \mathrel{\dot{\to}} \dot{T}X$.* [3]

For any $a \in (B^*)^*$, Va is above Aa. Therefore from (C2) and Proposition 3, for any arity $a \in (B^*)^*$, algebraic operation symbol $o \in \Sigma^a$ and $X \in \mathbb{E}$, we have $\alpha o_{pX} : Va \mathrel{\dot{\Rightarrow}} \dot{T}X \mathrel{\dot{\to}} \dot{T}X$. We define $\dot{\alpha}o_X$ to be its witness. Then $\dot{\alpha}o$ is a Va-ary algebraic operation for $\dot{\mathcal{T}}$.

We have obtained the $\lambda_c(B, \Sigma)$-structure $\mathcal{D}$ satisfying the conditions 1–3. One can easily show the basic lemma of logical relations:

Proposition 4. *For any well-typed $\lambda_c(B, \Sigma)$-term $x_1 : \rho_1, \cdots, x_n : \rho_n \vdash M : \rho$, we have $\mathcal{A}[\![M]\!] : \mathcal{D}[\![\rho_1]\!] \mathbin{\dot{\times}} \cdots \mathbin{\dot{\times}} \mathcal{D}[\![\rho_n]\!] \mathrel{\dot{\to}} \dot{T}\mathcal{D}[\![\rho]\!]$; its witness is given by $\mathcal{D}[\![M]\!]$.*

We finally show the condition 4.

Proposition 5. *For any base type $b \in B$, we have $\dot{T}Vb \leq Cb$ in $\mathbb{E}_{TAb}$.*

Proof. As $\dot{T}Vb = \bigwedge_{b' \in B} T^{\top\top(Cb')}Vb$, it is sufficient to show $T^{\top\top(Cb)}Vb \leq Cb$. Let us write $\dot{\eta}_b : Vb \to Cb$ for the witness of $\eta_{Ab} : Vb \mathrel{\dot{\to}} Cb$. Then in $\mathbb{E}$ we obtain a morphism

$$\dot{ev} \circ \dot{\langle \mathrm{id}, \dot{\lambda}(\dot{\eta}_b \circ \dot{\pi}_2) \rangle} \circ \overline{\sigma_{Ab}^{\mathcal{T},Ab}}((Vb \mathrel{\dot{\Rightarrow}} Cb) \mathrel{\dot{\Rightarrow}} Cb) : T^{\top\top(Cb)}Vb \to Cb$$

which is above $ev \circ \langle \mathrm{id}, \lambda(\eta_{Ab} \circ \pi_2) \rangle \circ \sigma_{Ab}^{\mathcal{T},Ab} = \mathrm{id}_{TAb}$. Therefore $T^{\top\top(Cb)}Vb \leq Cb$.

Theorem 1 is an immediate corollary of Proposition 4 and 5. This ends the proof.

[3] Though we do not use it, the converse of this statement holds: if $\alpha_{pX} : \dot{Z} \mathrel{\dot{\Rightarrow}} \dot{T}X \mathrel{\dot{\to}} \dot{T}X$ holds for all $X \in \mathbb{E}$, then $\alpha_{Ab} : \dot{Z} \mathrel{\dot{\Rightarrow}} Cb \mathrel{\dot{\to}} Cb$ holds for any base type $b \in B$.

4 Effect Simulation by Monad Morphism

Monadic semantics are often related by *strong monad morphisms*. Let $\mathcal{T}_i = (T_i, \eta_i, \mu_i, \theta_i)$ be strong monads ($i = 1, 2$) over a cartesian category $\mathbb{C}$. A strong monad morphism from $\mathcal{T}_1$ to $\mathcal{T}_2$ is a natural transformation $\sigma : T_1 \to T_2$ such that

$$\sigma_I \circ (\eta_1)_I = (\eta_2)_I \quad \sigma_I \circ (\mu_1)_I = (\mu_2)_I \circ \sigma_{T_2 I} \circ T_1 \sigma_I \quad (\theta_2)_{I,J} \circ I \times \sigma_J = \sigma_{I \times J} \circ (\theta_1)_{I,J}.$$

It transfers each Z-ary algebraic operation α for $\mathcal{T}_1$ to the following Z-ary algebraic operation $\sigma\alpha$ for $\mathcal{T}_2$:

$$(\sigma\alpha)_I = \ Z \Rightarrow T_2 I \xrightarrow{Z \Rightarrow (\eta_1)_{T_2 I}} Z \Rightarrow T_1 T_2 I \xrightarrow{\alpha_{T_2 I}} T_1 T_2 I \xrightarrow{\sigma_{T_2 I}} T_2 T_2 I \xrightarrow{(\mu_2)_I} T_2 I.$$

We define the *image* of a $\lambda_c(B, \Sigma)$-structure $\mathcal{A}_1 = (\mathbb{C}, \mathcal{T}_1, A, k, \alpha)$ along σ to be the $\lambda_c(B, \Sigma)$-structure $\sigma\mathcal{A}_1 = (\mathbb{C}, \mathcal{T}_2, A, k, \sigma\alpha)$, where $\sigma\alpha$ assigns the algebraic operation $\sigma(\alpha o)$ to each algebraic operation symbol $o \in \Sigma^a$ of arity $a \in (B^*)^*$.

Theorem 2. *Let $\mathcal{A} = (\mathbb{C}, \mathcal{T}_1, A, k, \alpha)$ be a $\lambda_c(B, \Sigma)$-structure such that $\mathbb{C}$ is small, $\mathcal{T}_2$ be a strong monad over $\mathbb{C}$ and $\sigma : \mathcal{T}_1 \to \mathcal{T}_2$ be a strong monad morphism. Then for any well-typed term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$, we have $\sigma \circ \mathcal{A}[\![M]\!] = (\sigma\mathcal{A})[\![M]\!]$.*

Proof. We pull-back the subobject fibration $\mathbf{Sub}([\mathbb{C}^{op}, \mathbf{Set}]) \to [\mathbb{C}^{op}, \mathbf{Set}]$ along the finite-product preserving functor $D : \mathbb{C}^2 \to [\mathbb{C}^{op}, \mathbf{Set}]$ defined by $D(I, J) = yI \times yJ$; here y is yoneda embedding. From Proposition 1, the leg of the pullback, say $q : \mathbb{K} \to \mathbb{C}^2$, is a category for logical relations. Now the following simulation (V, C) between $\mathcal{A}$ and $\sigma\mathcal{A}$ satisfies (I), (C1) and (C2):

$$VbH = \{(f, f) \mid f \in \mathbb{C}(H, Ab)\} \quad CbH = \{(f, \sigma_{Ab} \circ f) \mid f \in \mathbb{C}(H, T_1 Ab)\} \qquad (H \in \mathbb{C})$$

The goal is a corollary of Theorem 1 with the above simulation.

Example 4. (Continued from Example 1) There is a monad morphism σ from $\mathcal{T}_m$ to $\mathcal{T}_p$ mapping a list $l \in T_m I$ to the set $\sigma_I(l) \in T_p I$ of elements occurring in l. Thus from Theorem 2, for any well-typed term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$ and value $v_i \in Ab_i$, $\mathcal{A}_1[\![M]\!](\boldsymbol{v})$ is the set of elements occurring in $\mathcal{A}_2[\![M]\!](\boldsymbol{v})$.

Example 5. Let $\mathcal{A} = (\mathbb{C}, \mathcal{T}, A, k, \alpha)$ be a $\lambda_c(B, \Sigma)$-structure such that $\mathbb{C}$ is small. We write $C^{\mathcal{T},\perp}$ for the *continuation monad* with respect to the monad $\mathcal{T}$ and a result type $\perp$ (which is just an object in $\mathbb{C}$). The functor part of $C^{\mathcal{T},\perp}$ is given by $C^{\mathcal{T},\perp} I = (I \Rightarrow T\perp) \Rightarrow T\perp$. As we have seen in the proof of Theorem 1, there is a strong monad morphism $\sigma^{\mathcal{T},\perp} : \mathcal{T} \to C^{\mathcal{T},\perp}$. We instantiate Theorem 2 with it, and obtain an equation $\sigma^{\mathcal{T},\perp} \circ \mathcal{A}[\![M]\!] = \sigma^{\mathcal{T},\perp}\mathcal{A}[\![M]\!]$, particularly for any closed M. The r.h.s. of this equation is the *CPS semantics* [2] of $\lambda_c(B, \Sigma)$, while the l.h.s. roughly corresponds to $\lambda k \,.\, k^\#(\mathcal{A}[\![M]\!])$. This equation is indeed the *monadic congruence result* [2] restricted to base types.

	$\mathcal{A}_1$	$\mathcal{A}_2$
Category	$[\mathcal{I}, \mathbf{Set}]$	$\mathbf{Set}$
Monad	$T_1F = \mathrm{colim}_{Q\in\mathcal{I}} F(- + Q)$	$T_2I = \mathbf{N} \Rightarrow I \times \mathbf{N}$
Name type object	$A_1\mathbf{n} = N : \mathcal{I} \hookrightarrow \mathbf{Set}$	$A_2\mathbf{n} = \mathbf{N}$
Name equality predicate	$(k_1\mathbf{eq})_P(i,j) = \begin{cases} \iota_1(*) & (i = j) \\ \iota_2(*) & (i \neq j) \end{cases}$	$(k_2\mathbf{eq})(i,j) = \begin{cases} \iota_1(*) & (i = j) \\ \iota_2(*) & (i \neq j) \end{cases}$
Name creation	$\alpha_1\nu$: see (2) below	$\alpha_2\nu = \lambda f x \,.\, f(x)(x+1)$

Table 1. Definition of two ν-calculus structures

5 Comparing Two Monadic Semantics of ν-Calculus

Dynamic name creation, such as the one in π-calculus, is often categorically modelled in the presheaf category over the category $\mathcal{I}$ of finite sets and injections between them [25, 26]. On the other hand, in practical programming names are represented by natural numbers and dynamic name creation is implemented by a hidden global counter that keeps track of the next fresh name.

In this section, we consider Stark's *ν-calculus* [25] and discuss an effect simulation problem between presheaf semantics and global counter semantics of name creation. The ν-calculus has only one base type $\mathbf{n}$ for *names*, one effect-free constant $\mathbf{eq} : \mathbf{n}\times\mathbf{n} \to 1 + 1$ for checking name equality, and one algebraic operation $\nu(x^{\mathbf{n}}.M)$ whose intended meaning is to allocate a fresh name and bind it to x, like the one in π-calculus. We write Σ_ν for the signature specifying only these symbols. The ν-calculus is then defined to be $\lambda_c(\{\mathbf{n}\}, \Sigma_\nu)$. Below we call a $\lambda_c(\{\mathbf{n}\}, \Sigma_\nu)$-structure *$\nu$-calculus structure*.

In Table 1 we present two ν-calculus structures with which we consider an effect simulation problem. The ν-calculus structure $\mathcal{A}_1$ extracts the ingredients that are used in the categorical semantics of the ν-calculus in [25]. The monad $\mathcal{T}_1$ is Stark's *dynamic name creation monad*:

$$T_1FP = \mathrm{colim}_{Q\in\mathcal{I}} F(P + Q) = \{(Q, x) \mid Q \in \mathcal{I}, x \in F(P + Q)\}/\sim$$

where $(Q, x) \sim (R, y)$ if there are $S \in \mathcal{I}$ and two injections $l : Q \rightarrowtail S, m : R \rightarrowtail S$ such that $F(P + l)(x) = F(P + m)(y)$. We note $TFP \simeq F(P + 1)$. The object for the name type is the inclusion functor $N : \mathcal{I} \hookrightarrow \mathbf{Set}$; this is the standard choice for representing names. The behaviour of the name equality predicate at a finite set P is given in Table 1; there i, j are elements in P. The algebraic operation $\alpha_1\nu$ for name creation is defined by

$$((\alpha_1\nu)_F)_P(\alpha) = [1 + Q, F(i)(y)]_\sim \quad (\text{where } [Q, y]_\sim = \alpha_{P+1}(\iota_1, \iota_2)) \tag{2}$$

where $i : (P + 1) + Q \to P + (1 + Q)$ is the coherence isomorphism.

The ν-calculus structure $\mathcal{A}_2$ is a semantic analogue of dynamic name creation by a global state. We note that the interpretation $\mathcal{A}_2[\![-]\!]$ is not sound with respect to the ν-calculus axioms in [25].

We compare the denotation of a well-typed term $x_1 : \mathbf{n}, \ldots, x_n : \mathbf{n} \vdash M : \mathbf{n}$ in each ν-calculus structure. Suppose that p names have been allocated, and some of them are supplied to the free variables of M. Then M returns either one of the allocated names supplied to its free variables, or M allocates a new name and returns it. This behaviour is expressed differently in each ν-calculus structure:

- (in $\mathcal{A}_1$) Let P be the finite set consisting of p allocated names. We feed $i_1, \ldots, i_n \in P$ to the free variables of M. When M returns an allocated name, the denotation $\mathcal{A}_1 [\![M]\!]_P (\boldsymbol{i}) \in TNP \simeq P + 1$ is $\iota_1(i)$ with some $i \in P$. Otherwise, M returns a new name and the denotation is $\iota_2(*)$.
- (in $\mathcal{A}_2$) Natural numbers $0, \ldots, p-1$ correspond to the allocated names. We thus feed $0 \leq i_1 \ldots i_n < p$ to the free variables of M. The global counter pointing to the next fresh name is now p, so the name that M returns is given by $i = \pi_1(\mathcal{A}_2 [\![M]\!] (\boldsymbol{i})(p))$. When M returns an allocated name, $0 \leq i < p$; otherwise $i \geqslant p$. In fact, this behaviour of M remains the same even when the counter is increased from p. Therefore when M returns an allocated name i, for any $k \geq p$ we have $\pi_1(\mathcal{A}_2 [\![M]\!] (\boldsymbol{i})(k)) = i$; otherwise for any $k \geq p$ we have $\pi_1(\mathcal{A}_2 [\![M]\!] (\boldsymbol{i})(k)) \geq k$.

Based on this analysis, we establish a correspondence between the denotation of M in each ν-calculus structure. As names are represented differently, this relationship is parametrised by bijective correspondences between allocated names and natural numbers. Below for a finite set P, by $|P|$ we mean its cardinality. For a natural number p, we write $\overline{p}$ for the finite set $\{0, \cdots, p-1\}$. A *name enumeration* is a bijection $\sigma : P \to \overline{|P|}$.

Theorem 3. *Let $x_1 : \mathbf{n}, \ldots, x_n : \mathbf{n} \vdash M : \mathbf{n}$ be a ν-calculus term. For any finite set P, elements $i_1 \ldots i_n \in P$ and a name enumeration $\sigma : P \to \overline{|P|}$, either*

- *there is $i \in P$ such that $\mathcal{A}_1 [\![M]\!]_P (\boldsymbol{i}) = \iota_1(i)$ and for all $k \geq |P|$, we have $\pi_1 \circ \mathcal{A}_2 [\![M]\!] (\sigma(\boldsymbol{i}))(k) = \sigma(i)$, or*
- *$\mathcal{A}_1 [\![M]\!]_P (\boldsymbol{i}) = \iota_2(*)$ and for all $k \geq |P|$, we have $\pi_1 \circ \mathcal{A}_2 [\![M]\!] (\sigma(\boldsymbol{i}))(k) \geqslant k$.*

The rest of this section is the proof of this theorem. We construct a suitable category for logical relations over $[\mathcal{I}, \mathbf{Set}] \times \mathbf{Set}$, and give a simulation (V, C) between $\mathcal{A}_1$ and $\mathcal{A}_2$ that implies the goal of the theorem. We then check that it satisfies (I), (C1) and (C2).

We observe that the theorem is parametrised by name enumerations, so we first introduce the category $\mathcal{E}$ of name enumerations. As defined above, a name enumeration is a bijection $\sigma : P \to \overline{|P|}$. A morphism h from σ to $\tau : Q \to \overline{|Q|}$ is a (necessarily unique) injection $h : P \to Q$ such that $\sigma = \tau \circ h$. We note that $\mathcal{E}$ is actually equivalent to $(\mathbf{N}, \leq)$. There is an evident projection functor $\pi : \mathcal{E} \to \mathcal{I}$.

We next pull-back the subobject fibration $\mathbf{Sub}([\mathcal{E}, \mathbf{Set}]) \to [\mathcal{E}, \mathbf{Set}]$ along the finite-product preserving functor $D : [\mathcal{I}, \mathbf{Set}] \times \mathbf{Set} \to [\mathcal{E}, \mathbf{Set}]$ defined by $D(F, I) = (F \circ \pi) \times \Delta I$. We obtain the category $q : \mathbf{ERel} \to [\mathcal{I}, \mathbf{Set}] \times \mathbf{Set}$ for logical relations by Proposition 1. An object in $\mathbf{ERel}$ is a triple (X, F, I) where $F \in [\mathcal{I}, \mathbf{Set}]$, $I \in \mathbf{Set}$ and X assigns a binary relation $X\sigma \subseteq FP \times I$ to each name enumeration $\sigma : P \to \overline{|P|}$. Moreover, X should satisfy the *monotonicity condition*: for any $h \in \mathcal{E}(\sigma, \tau)$ and $(x, y) \in X\sigma$, we have $(Fhx, y) \in X\tau$.

We give the simulation (V, C) between $\mathcal{A}_1$ and $\mathcal{A}_2$ that entails Theorem 3. For each name enumeration $\sigma : P \to \overline{|P|}$, we define $V\mathbf{n}\sigma = \{(i, \sigma(i)) \mid i \in P\}$ and

$$\begin{aligned} C\mathbf{n}\sigma = {} & \{(\iota_1(i), f) \mid i \in P \wedge \forall k \geq |P| \,.\, \pi_1 \circ f(k) = \sigma(i)\} \; \cup \\ & \{(\iota_2(*), f) \mid \forall k \geq |P| \,.\, \pi_1 \circ f(k) \geq k\}. \end{aligned}$$

Proposition 6. *The above simulation (V, C) satisfies (I), (C1) and (C2).*

Theorem 3 is an immediate corollary of Theorem 1 with the above simulation.

$$\frac{\Gamma, f : \rho \to \sigma, x : \rho \vdash M : \sigma}{\Gamma \vdash \mu f x.M : \rho \to \sigma} \qquad \begin{aligned} &\mathcal{A}[\![\Gamma \vdash \mu f x.M : \rho \Rightarrow \sigma]\!] \\ &= \mathbf{fix}^{\mathcal{A}[\![\Gamma]\!]}_{\mathcal{A}[\![\rho \Rightarrow \sigma]\!]}(\mathcal{A}[\![\lambda x\,.\,M]\!]^{\#} \circ \theta_{\mathcal{A}[\![\Gamma]\!],\mathcal{A}[\![\rho \Rightarrow \sigma]\!]}). \end{aligned}$$

Fig. 1. Recursion: Typing Rule and Interpretation

6 Extending $\lambda_c(B, \Sigma)$ with Recursive Functions

We next add the recursive function constructor $\mu f x.M$ to $\lambda_c(B, \Sigma)$. This term creates a closure that may recursively call itself inside M; see Figure 1 for its typing rule. We call the extended calculus $\lambda_c^{\mathbf{fix}}(B, \Sigma)$. To interpret the recursion under the presence of computation, we employ *uniform T-fixpoint operator* [24]. An equivalent, direct formulation of recursion in call-by-value is also studied in [8]. Let $\mathcal{T} = (T, \eta, \mu, \theta)$ be a strong monad over a cartesian category $\mathbb{C}$. A *uniform T-fixpoint operator* for $\mathcal{T}$ is a family of mappings $\mathbf{fix}_I : \mathbb{C}(TI, TI) \to \mathbb{C}(1, TI)$ such that $\mathbf{fix}_I(f) = f \circ \mathbf{fix}_I(f)$, and it satisfies the *uniformity principle*: for any $f : TI \to TI, g : TJ \to TJ$ and $h : I \to TJ$, $h^{\#} \circ f = g \circ h^{\#}$ implies $g = h^{\#} \circ \mathbf{fix}_I(f)$. When $\mathbb{C}$ is a CCC, we can parametrise it as $\mathbf{fix}_I^X : \mathbb{C}(X \times TI, TI) \to \mathbb{C}(X, TI)$; see [24] for the detail.

Definition 5. *A $\lambda_c^{\mathbf{fix}}(B, \Sigma)$-structure is a pair of a $\lambda_c(B, \Sigma)$-structure $\mathcal{A}$ and a uniform T-fixpoint operator* **fix** *for the strong monad of $\mathcal{A}$.*

The interpretation of a recursive function constructor is given in Figure 1.

We aim to extend Theorem 1 to $\lambda_c^{\mathbf{fix}}(B, \Sigma)$ and a $\lambda_c^{\mathbf{fix}}(B, \Sigma)$ structure $\mathcal{A}$ where 1) the category of $\mathcal{A}$ is ω**CPO**-*enriched*[4], 2) the strong monad of $\mathcal{A}$ *lifts* the given domain and 3) the uniform T-fixpoint operator is given by the least fixpoint. That $\mathcal{T}$ lifts a given domain (as defined below) is expressed by the fact that $\mathcal{T}$ admits an algebraic operation denoting the least element of TI.

Definition 6. *We call a strong monad $\mathcal{T}$ over a* **Pos**-*enriched bi-CCC $\mathbb{C}$* pseudo-lifting *if it has an 0-ary algebraic operation $\perp$ such that for any $I \in \mathbb{C}$, $\perp_I$ is the least element in $\mathbb{C}(0 \Rightarrow TI, TI) \simeq \mathbb{C}(1, TI)$.*

We note that for any pseudo-lifting monad $(\mathcal{T}_1, \perp)$ over a **Pos**-enriched bi-CCC $\mathbb{C}$, $\mathcal{T}_2$ be another strong monad over $\mathbb{C}$ and strong monad morphism $\sigma : \mathcal{T}_1 \to \mathcal{T}_2$, the pair $(\mathcal{T}_2, \sigma\perp)$ is pseudo-lifting and σ is strict, that is, $\sigma_I \circ \perp_I = (\sigma\perp)_I$.

Definition 7. *An ω***CPO**-*enriched $\lambda_c(B, \Sigma)$-structure is a tuple $(\mathbb{C}, \mathcal{T}, A, k, \alpha, \perp)$ such that the first five components form a $\lambda_c(B, \Sigma)$-structure, $\mathbb{C}$ is an ω***CPO**-*enriched bi-CCC and $(\mathcal{T}, \perp)$ is a pseudo-lifting monad.*

We can turn every ω**CPO**-enriched $\lambda_c(B, \Sigma)$-structure into a $\lambda_c^{\mathbf{fix}}(B, \Sigma)$-structure by paring it with the uniform T-fixpoint operator given by $\mathbf{fix}_I(f) = \bigsqcup_{n \in \mathbf{N}} f^{(n)} \circ \perp_I$.

Let $\mathcal{A} = (\mathbb{C}, \mathcal{T}, A, k, \alpha, \perp)$ be an ω**CPO**-enriched $\lambda_c(B, \Sigma)$-structure, and $p : \mathbb{E} \to \mathbb{C}$ be a category for logical relations. Since p is faithful, we can restrict the partial order

[4] We write **Pos** for the category of posets and monotone functions with finite products as the symmetric monoidal structure. The category ω**CPO** is a subcategory of **Pos** where objects are ω-complete partial orders and and morphisms are continuous functions.

on $\mathbb{C}(pX, pY)$ to $\mathbb{E}(X, Y)$. Moreover, as p strictly preserves bi-cartesian closed structure, $\mathbb{E}$ becomes a **Pos**-enriched bi-CCC. We call $X \in \mathbb{E}$ above TI *admissible* if 1) $\perp_I : \dot{1} \dot{\rightarrow} X$ and 2) for any $Y \in \mathbb{E}$ and ω-chain $\dot{f}_i$ in $\mathbb{E}(Y, X)$, we have $\bigsqcup_{i=0}^{\infty}(p\dot{f}_i) : Y \dot{\rightarrow} X$.

Theorem 4. *Let $\mathcal{A} = (\mathbb{C}, \mathcal{T}, A, k, \alpha, \perp)$ be an ω**CPO**-enriched $\lambda_c(B, \Sigma)$-structure, $p : \mathbb{E} \to \mathbb{C}$ be a category for logical relations, (V, C) be a property over $(\mathbb{C}, \mathcal{T}, A, k, \alpha)$ such that it satisfies (I), (C1) and (C2) and Cb is admissible for all base types $b \in B$. Then for any well-typed $\lambda_c^{\mathbf{fix}}(B, \Sigma)$-term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$, we have $\mathcal{A}[\![M]\!] : Vb_1 \dot{\times} \cdots \dot{\times} Vb_n \dot{\rightarrow} Ob$.*

Proof. As done in the proof of Theorem 1, we obtain a $\lambda_c(B, \Sigma)$-structure $(\mathbb{E}, \dot{\mathcal{T}}, V, \dot{\alpha})$.

We next show that for any object X in $\mathbb{E}$ above an object I in $\mathbb{C}$ and morphism $f : \dot{T}X \to \dot{T}X$ in $\mathbb{E}$, we have $\mathbf{fix}(pf) : \dot{1} \dot{\rightarrow} \dot{T}X$. From $\perp_{Ab} : \dot{1} \dot{\rightarrow} Ob$ for each b, we obtain $\perp_I : \dot{1} \dot{\rightarrow} \dot{T}X$ for any object X in $\mathbb{E}$ by Proposition 2. We write $\dot{\perp}_X : \dot{1} \to \dot{T}X$ for its witness. We then have an ω-chain $f^{(n)} \circ \dot{\perp}_X$ in $\mathbb{E}(\dot{1}, \dot{T}X)$. From the definition of $\dot{T}X$, for any b we have another ω-chain: $\dot{\lambda}^{-1}(\dot{\sigma}_I^b \circ f^{(n)} \circ \dot{\perp}_I)$ in $\mathbb{E}(\dot{1} \dot{\times} (X \dot{\Rightarrow} Ob), Ob)$. Since Ob is admissible, we have $\lambda^{-1}(\sigma_I^b \circ \mathbf{fix}(pf)) : \dot{1} \dot{\times} (X \dot{\Rightarrow} Ob) \dot{\rightarrow} Ob$. Thus by currying it, we have $\sigma_I^b \circ \mathbf{fix}(pf) : \dot{1} \dot{\rightarrow} (X \dot{\Rightarrow} Ob) \dot{\Rightarrow} Ob$. As $\mathbb{E}$ is a fibration, we have $\mathbf{fix}(pf) : \dot{1} \dot{\rightarrow} \dot{T}X$. We name the witness of this $\dot{\mathbf{fix}} f : \dot{1} \to \dot{T}X$.

One can easily check that the mapping $f \mapsto \dot{\mathbf{fix}} f$ is indeed a uniform T-fixpoint operator for $\dot{\mathcal{T}}$. Therefore the tuple $\mathcal{D} = (\mathbb{E}, \dot{\mathcal{T}}, V, \dot{\alpha}, \dot{\mathbf{fix}})$ is a $\lambda_c^{\mathbf{fix}}(B, \Sigma)$-structure such that for any well-typed $\lambda_c^{\mathbf{fix}}(B, \Sigma)$-term $x_1 : \rho_1, \cdots, x_n : \rho_n \vdash M : \rho$, we have $p\mathcal{D}[\![M]\!] = \mathcal{A}[\![M]\!]$ (the basic lemma of logical relation). Theorem 4 then follows from this.

Theorem 5. *Let $\mathcal{A} = (\mathbb{C}, \mathcal{T}_1, k, A, \alpha, \perp)$ be an ω**CPO**-enriched $\lambda_c(B, \Sigma)$-structure such that $\mathbb{C}$ is small, $\mathcal{T}_2$ be a strong monad over $\mathbb{C}$ and $\sigma : \mathcal{T}_1 \to \mathcal{T}_2$ be a strong monad morphism. Then for any well-typed $\lambda_c^{\mathbf{fix}}(B, \Sigma)$-term $x_1 : b_1, \cdots, x_n : b_n \vdash M : b$, we have $\sigma \circ \mathcal{A}[\![M]\!] = (\sigma\mathcal{A})[\![M]\!]$.*

Proof. The proof is the same as that of Theorem 2. To apply Theorem 4, it suffices to check that the simulation Ob satisfies 1) $(\perp_{Ab}, \sigma\perp_{Ab}) : \dot{1} \dot{\rightarrow} Ob$ and 2) for any object X in $\mathbb{K}$ and ω-chain f_i in $\mathbb{K}(X, Ob)$, we have $\bigsqcup_{i=0}^{\infty} pf_i : X \dot{\rightarrow} Ob$. Below we write (g_i, h_i) for pf_i.

1) is immediate from the strictness of σ.

2) Let H be an object in $\mathbb{C}$ and $(a, b) \in XH$. We show $\left(\bigsqcup_{i=0}^{\infty} h_i\right) \circ b = \sigma_{Ab} \circ \left(\bigsqcup_{i=0}^{\infty} g_i\right) \circ a$. First, from the assumption $f_i : X \to Ob$, we obtain $h_i \circ b = \sigma_{Ab} \circ g_i \circ a$ for any natural number i. Next, as g_i and h_i are ω-chains in $\mathbb{C}(pX, TAb)$, their l.u.b. $\bigsqcup_{i=0}^{\infty} g_i, \bigsqcup_{i=0}^{\infty} h_i$ exist. Then from the continuity of composition of $\mathbb{C}$, we obtain

$$\left(\bigsqcup_{i=0}^{\infty} h_i\right) \circ b = \bigsqcup_{i=0}^{\infty} (h_i \circ b) = \bigsqcup_{i=0}^{\infty} (\sigma_{Ab} \circ g_i \circ a) = \sigma_{Ab} \circ \left(\bigsqcup_{i=0}^{\infty} g_i\right) \circ a.$$

7 Related Work

Filinski is one of the pioneers in logical relations for monads [3], and developed various techniques to establish relationships between semantics of higher-order languages with effects [2–6]. He pointed out at least three methods to obtain logical relations for monads: 1) When T is a syntactically constructed monad, such as state monad and continuation monad, then define a logical relation $\dot{T}$ for T in the same way as T is constructed. 2) For a logical relation $\dot{T}$ for a monad T and a strong monad morphism $\sigma : S \to T$, the inverse image $\sigma^* \dot{T}$ is a logical relation for S. 3) For a family of logical relations $\dot{T}_i$ for a monad T, the intersection $\bigwedge_i \dot{T}_i$ is again a logical relation for T. A method based on factorisation systems is also proposed by Larrecq et al [13].

The categorical $\top\top$-lifting is technically a particular combination of the methods 1–3 (in fibrational category theory). However, what is new about $\top\top$-lifting is that we *use* the simulation / property we would like to establish on computational effects to define the logical relation $\dot{T}$; see definition (1). This idea is a secret recipe in the proofs of various results by the precursors of categorical $\top\top$-lifting, such as biorthogonality [7, 16, 20], $\top\top$-closure [21] and leapfrog method [14, 15]; see also [12].

The advantage of logical relations for monads by $\top\top$-lifting is that it does not limit the form of simulation / property we would like to establish on computational effects. Furthermore, Proposition 3 gives a good characterisation of when algebraic operations are related by the logical relations given by $\top\top$-lifting. On the other hand, it is rather difficult to check whether non-algebraic operations that manipulate computational effects, such as Felleisen's $\mathcal{C}$- and $\mathcal{A}$-operators, are related by the logical relations given by $\top\top$-lifting; this shall be discussed in a separate paper. Extending our results with recursive types and handlers for algebraic effects [23] is also a future work.

Acknowledgement The author is grateful to Masahito Hasegawa, Naohiko Hoshino and Susumu Nishimura for fruitful discussions. He thanks anonymous reviewers for valuable comments. This research was partly supported by Grant-in-Aid for Young Scientists (B) 20700012.

References

1. N. Benton, J. Hughes, and E. Moggi. Monads and effects. In *Proc. APPSEM 2000*, volume 2395 of *LNCS*, pages 42–122. Springer, 2002.
2. Andrzej Filinski. Representing monads. In *Proc. POPL 1994*, pages 446–457, 1994.
3. Andrzej Filinski. *Controlling Effects*. PhD thesis, Carnegie Mellon University, 1996.
4. Andrzej Filinski. Representing layered monads. In *Proc. POPL 1999*, pages 175–188, 1999.
5. Andrzej Filinski. On the relations between monadic semantics. *Theor. Comput. Sci.*, 375(1-3):41–75, 2007.
6. Andrzej Filinski and K. Støvring. Inductive reasoning about effectful data types. In R. Hinze and N. Ramsey, editors, *ICFP*, pages 97–110. ACM, 2007.
7. Jean.-Yves. Girard. Linear logic. *Theor. Comp. Sci.*, 50:1–102, 1987.
8. Masahito Hasegawa and Yoshihiko Kakutani. Axioms for recursion in call-by-value. *Higher-Order and Symbolic Computation*, 15(2-3):235–264, 2002.
9. Claudio Hermida. *Fibrations, Logical Predicates and Indeterminants*. PhD thesis, University of Edinburgh, 1993.
10. Bart Jacobs. *Categorical Logic and Type Theory*. Elsevier, 1999.
11. Shin-ya Katsumata. A semantic formulation of $\top\top$-lifting and logical predicates for computational metalanguage. In *Proc. CSL 2005*, volume 3634 of *LNCS*, pages 87–102. Springer, 2005.

12. Shin-ya Katsumata. A characterisation of lambda definability with sums via $\top\top$-closure operators. In *Proc. CSL 2008*, volume 5213 of *LNCS*, pages 278–292. Springer, 2008.
13. J.-G. Larrecq, S. Lasota, and D. Nowak. Logical relations for monadic types. *Math. Struct. in Comp. Science*, 18:1169–1217, 2008.
14. Samuel Lindley. *Normalisation by Evaluation in the Compilation of Typed Functional Programming Languages*. PhD thesis, University of Edinburgh, 2004.
15. Samuel Lindley and Ian Stark. Reducibility and $\top\top$-lifting for computation types. In *Proc. TLCA*, pages 262–277, 2005.
16. Paul-André Melliès and Jerome Vouillon. Recursive polymorphic types and parametricity in an operational framework. In *LICS*, pages 82–91. IEEE Computer Society, 2005.
17. John Mitchell. Representation independence and data abstraction. In *Proc. POPL 1986*, pages 263–276, 1986.
18. John Mitchell. *Foundations for Programming Languages*. MIT Press, 1996.
19. John Mitchell and Andre Scedrov. Notes on sconing and relators. In *CSL*, volume 702 of *Lecture Notes in Computer Science*, pages 352–378. Springer, 1992.
20. Michel Parigot. Proofs of strong normalisation for second order classical natural deduction. *Journal of Symbolic Logic*, 62(4):1461–1479, 1997.
21. Andrew Pitts. Parametric polymorphism and operational equivalence. *Mathematical Structures in Computer Science*, 10(3):321–359, 2000.
22. Gordon Plotkin and John Power. Semantics for algebraic operations. *Electr. Notes Theor. Comput. Sci.*, 45, 2001.
23. Gordon D. Plotkin and Matija Pretnar. Handlers of algebraic effects. In Giuseppe Castagna, editor, *ESOP*, volume 5502 of *LNCS*, pages 80–94. Springer, 2009.
24. Alex Simpson and Gordon Plotkin. Complete axioms for categorical fixed-point operators. In *LICS*, pages 30–41, 2000.
25. Ian Stark. Categorical models for local names. *Lisp and Symbolic Computation*, 9(1):77–107, February 1996.
26. Ian Stark. A fully abstract domain model for the π-calculus. In *Proc. LICS 1996*, pages 36–42. IEEE, 1996.