Step Indexed Realizability Semantics for a Call-by-Value Language Based on Basic Combinatorial Objects

Naohiko Hoshino
Research Institute for Mathematical Sciences, Kyoto University
Kyoto, Japan

***Abstract*—We propose a mathematical framework for step indexed realizability semantics of a call-by-value polymorphic lambda calculus with recursion, existential types and recursive types. Our framework subsumes step indexed realizability semantics by untyped call-by-value lambda calculi as well as categorical abstract machines. Starting from an extension of Hofstra's basic combinatorial objects, we construct a step indexed categorical realizability semantics. Our main result is soundness and adequacy of our step indexed realizability semantics. As an application, we show that a small step operational semantics captures the big step operational semantics of the call-by-value polymorphic lambda calculus. We also give a safe implementation of the call-by-value polymorphic lambda calculus into a categorical abstract machine.**

***Index Terms*—Semantics of Programming Languages, operational semantics, denotational semantics.**

I. Introduction

Realizability is a powerful tool for studies of syntactic systems. In this paper, we use realizability as a construction of denotational semantics from operational semantics. Such denotational semantics provides

- A correct translation (compilation) of a programming language modeled by the denotational semantics into the operational semantics.
- A translation of types in a programming language modeled by the denotational semantics into specifications of terms in the operational semantics.

There are several mathematical structures for starting points of realizability semantics such as SK algebra, partial combinatory algebra, linear combinatory algebra, ordered partial combinatory algebra and basic combinatorial object [24], [1], [16]. From these combinatory algebras, we can build categories of partial equivalence relations, categories of assemblies, realizability topoi and many variants of them. These categories model lambda calculi as well as dependent type theory and higher order predicate logic.

Recursive computation and recursive types are important features for functional programming languages. So as to model them, we need to construct a denotational semantics that has a fixed point operator and solutions of various domain equations. However, it is not obvious in which situation we can obtain a fixed point operator and solve domain equations. One candidate is to assume domain theoretic structures on combinatory algebras and consider a category of admissible partial equivalence relations [3]. Another candidate is synthetic domain theory that aims to provide models of recursive computation in a set theory [24]. Difficulties towards construction of realizability semantics for recursive computation and recursive types are that most operational semantics does not have domain theoretic structure and that synthetic domain theory may require nontrivial analysis of syntactic properties of combinatorial algebras. For example, see Section 8 in [21].

Step indexing is a powerful tool to connect operational semantics and programming languages with recursive computation and recursive types. The idea of step indexing is to stratify realizability semantics by indices that represent length of reduction sequences. We give an example of step indexing argument. We consider a program `fact` that computes the factorial $\mathtt{fact} := (\lambda \mathtt{x.g}(\lambda \mathtt{y.xxy}))(\lambda \mathtt{x.g}(\lambda \mathtt{y.xxy}))$ where `g` is $\lambda \mathtt{fn}.\ \mathtt{if\ n = 0\ then\ 1\ else\ n} * \mathtt{f}(\mathtt{n}-1)$. We show that `fact` satisfies a specification "for all integer `n`, if the evaluation of `fact n` stops, then the result is a natural number". A proof is by induction on the length of the reduction sequence from `fact n`: first, if the evaluation of `fact n` stops within 0-step reduction, then the result is a natural number. Next, since we always have

$$\begin{array}{ll} \mathtt{fact\ n} \rightsquigarrow\ \mathtt{n} * (\mathtt{fact}\ (\mathtt{n}-1)) & (\mathtt{n} \neq 0) \\ \mathtt{fact\ n} \rightsquigarrow 1 & (\mathtt{n} = 0), \end{array}$$

if the evaluation of `fact n` stops within $(i+1)$-step reductions, then `n` is 0 or the evaluation of $\mathtt{fact}\ (\mathtt{n}-1)$ stops within i-step reductions. Therefore, by the induction hypothesis, the result of `fact n` is a natural number. Hence, we see that `fact n` satisfies the specification. In this example, the step index i represents length of a reduction sequence, and we approximated the specification by stratified specifications "for all integer `n`, if the evaluation of `fact n` stops within i-step reductions, then the result is a natural number".

There are various step indexed semantics and stratifications of types/specifications. Dami indexed terms of lambda calculi and stratified types by indices of terms [10], [11]. Appel and McAllester considered a different approach, where they stratified types by indices that represent length of reduction sequences for small step operational semantics of a lambda calculus [4]. They also provided a step indexed relational model. In [2], Ahmed fixed a problem on their proof of the transitivity. Since we construct a categorical realizability

model by using partial equivalence relations, our work seems to be related to their works. Dreyer, Ahmed and Birkedal provided an abstract logical framework to reason about logical relations "indexed by length of reduction sequences" [12]. The main point is the use of the later modality [23]; although their idea is based on Appel and McAllester's work, their argument does not involve any step indexed reasoning. More abstractly, Birkedal and Møgelberg provided a step indexed categorical model for dependent type theory by a presheaf topos [8]. Benton and Hur showed correctness of an implementation of the call-by-value PCF into a SECD machine by means of step indexing [6]. Their indexing represents length of transition sequences of a SECD machine. They also employed orthogonality between indexed predicates. Melliès and Vouillon considered stratification of coinductively defined types by intermediate types [22].

The aim of this paper is to give a mathematically abstract framework for step indexed realizability semantics that does not depend on specific syntax of operational semantics so as to increase manageability and structured understanding of step indexing. Our motivation comes from the following quotation in [7]. (Benton and Hur noted the following as a reason for mechanized proof by Coq.)

> $\cdots$ the reasoning is already sufficiently complex that we really would have low confidence in (and would find it hard to manage) paper proofs, especially for nontrivial examples such as the polymorphic list module $\cdots$

Study towards abstract and flexible theory of logical relations in the presence of step indexing has already been addressed in [18]. Mathematical abstraction helps us to see essence of (complex) reasoning: it explicitly describes which properties and data are used in arguments and constructions and enables us to study various realizers and compilations in a uniform way.

In order to clarify on which mathematical structures our argument depends, we begin with an abstract mathematical object and construct categorical realizability models. We do not start with specific operational semantics like the above related works. Our starting point is a variant of Hofstra's basic combinatorial object [16], which is, roughly speaking, a set with a monoid action, and the monoid action represents a collection of computation on the underlying set. This point of view fits naturally to operational semantics of lambda calculi as well as categorical abstract machines. See examples in Section VII.

We aim to construct categorical realizability semantics as categorical semantics clarifies mathematical structures of our realizability semantics (Section V). In the above related works, [22] and [8] provide categorical models. In our construction of realizability semantics, we employ orthogonality relation stratified by step indexing. Orthogonality argument, which appears in the proof of cut elimination for linear logic [15], is one of the most powerful methodology in realizability and logical relation. In the context of step indexing, [6] and [13] employed orthogonality in their argument. The idea of our step indexed orthogonality relations is based on the orthogonality in [6]: we approximate orthogonality relation by step indices. Use of orthogonality enables us to parameterize "by which we observe" and "how we observe". For example, in Section III, we give an orthogonality that counts length of reduction sequences of lambda terms, and in Section VII-A, we consider an orthogonality that counts number of particular reduction steps, namely reduction steps for recursion and recursive types.

In the remainder of this paper, we introduce cbv combinatorial object that is an extension of Hofstra's basic combinatorial object (Section II), and we associate an indexed orthogonality to cbv combinatorial object by means of indexed predicate and evaluation (Section III). In Section IV and Section V, we construct a categorical realizability semantics parameterized by cbv combinatorial objects, evaluations and indexed predicates. In Section VI, we describe a call-by-value polymorphic lambda calculus with recursion, existential types and recursive types, and we give our main results: soundness and adequacy. In Section VII, we give two examples of cbv combinatorial objects and prove several syntactic properties.

II. CBV COMBINATORIAL OBJECT

For a reflexive transitive relation $\rightsquigarrow$ on a set A, a *homomorphism* on A is a map $f : A \to A$ such that if $a \rightsquigarrow a'$, then $f(a) \rightsquigarrow f(a')$. We write $\mathrm{Hom}(A, A)$ for the set of homomorphisms on A.

Definition II.1. A *total basic combinatorial object* is a preordered set $(A, \rightsquigarrow)$ with a set Φ of homomorphisms on A and a subset $V \subset A$ such that

- There exists $\iota \in \Phi$ such that for all $x \in V$, we have $\iota(x) \rightsquigarrow x$.
- For each $\phi, \psi \in \Phi$, there exists $\psi \cdot \phi \in \Phi$ such that for any $x \in V$, we have $\psi \cdot \phi(x) \rightsquigarrow \psi(\phi(x))$.

We can regard Φ as a set of "realizable maps" or "computable maps" and V as a set of "values". So as to save letters and space, we assume that we make a choice of ι and $\phi \cdot \psi$ for each $\phi, \psi \in \Phi$. In the following, we use $a, b, c, a', b', c' \cdots$ for elements in A and $x, y, z, x', y', z' \cdots$ for elements in V.

Hofstra's basic combinatorial objects are slightly different from total basic combinatorial objects: in the definition of Hofstra's basic combinatorial objects, Φ is a set of "partial homomorphisms" rather than homomorphisms, and V coincides with A.

Definition II.2. A map $f : V \to V$ is *computable* when there exists $\phi \in \Phi$ such that for any $x \in V$, we have $\phi(x) \rightsquigarrow f(x)$. We say that ϕ *computes* f.

Definition II.3. A *cbv combinatorial object* is a total basic combinatorial object $(A, \rightsquigarrow, \Phi, V)$ with the following data:

- $* \in V$
- $\upsilon, \pi, \pi', \varepsilon \in \Phi$
- A binary map $\langle -, - \rangle_A : V \times V \to V$
- A binary map $\langle -, - \rangle_\Phi : \Phi \times \Phi \to \Phi$
- $\gamma_{x,\phi}, \delta_x \in \mathrm{Hom}(A, A)$ for each $x \in V$ and $\phi \in \Phi$
- A computable map $\phi \star (-) : V \to V$ for each $\phi \in \Phi$

such that for any $x, y \in V$,

$\upsilon(x) \rightsquigarrow *$	discarding
$\pi\langle x, y\rangle_A \rightsquigarrow x$	first projection
$\pi'\langle x, y\rangle_A \rightsquigarrow y$	second projection
$\langle \phi, \psi\rangle_\Phi(x) \rightsquigarrow \gamma_{x,\psi}(\phi(x))$	evaluation of the left
$\gamma_{x,\psi}(y) \rightsquigarrow \delta_y(\psi(x))$	evaluation of the right
$\delta_y(x) \rightsquigarrow \langle y, x\rangle_A$	pair of values
$\varepsilon\langle \phi \star x, y\rangle_A \rightsquigarrow \phi\langle x, y\rangle_A$	β-reduction

The definition of cbv combinatorial objects is reminiscent of reduction rules for categorical combinators described in the introductory part of [9]. The value $*$, which is not present there, is to guarantee existence of values. Intuitively, $a \rightsquigarrow a'$ means that there is a (possibly 0-step) reduction sequence from a to a', and we call $a \rightsquigarrow a'$ a reduction.

In the following, we simply write A for a cbv combinatorial object without describing other data, and we omit subscripts of tupling maps $\langle -, -\rangle$, namely A and Φ, when we can infer them from context.

We give examples of cbv combinatorial objects in Section VII. So as to explain intuition of cbv combinatorial objects, we give a primal example. Let λ^{v} be an untyped call-by-value lambda calculus:

Term	$\mathsf{t} ::= \mathsf{x} \mid () \mid \mathsf{tt} \mid \lambda\mathsf{x}.\mathsf{t} \mid (\mathsf{t}, \mathsf{t}) \mid \mathsf{fst}(\mathsf{t}) \mid \mathsf{snd}(\mathsf{t})$
Value	$\mathsf{v} ::= \mathsf{x} \mid () \mid \lambda\mathsf{x}.\mathsf{t} \mid (\mathsf{v}, \mathsf{v})$
Evaluation context	$\mathsf{e} ::= [-] \mid \mathsf{et} \mid \mathsf{ve} \mid (\mathsf{e}, \mathsf{t}) \mid (\mathsf{v}, \mathsf{e})$
Reduction rule	$\mathsf{e}[(\lambda\mathsf{x}.\mathsf{t})\mathsf{v}] \mapsto \mathsf{e}[\mathsf{t}[\mathsf{v}/\mathsf{x}]]$
	$\mathsf{e}[\mathsf{fst}(\mathsf{v}, \mathsf{v}')] \mapsto \mathsf{e}[\mathsf{v}]$
	$\mathsf{e}[\mathsf{snd}(\mathsf{v}, \mathsf{v}')] \mapsto \mathsf{e}[\mathsf{v}']$

We define a total basic combinatorial object: we define $A_{\lambda^{\mathrm{v}}}$, $V_{\lambda^{\mathrm{v}}}$ and $\Phi_{\lambda^{\mathrm{v}}}$ to be the set of closed terms, the set of closed values and the set of homomorphisms of the form $\phi(-) = \mathsf{v}(-)$ for some closed value v respectively. We define $\rightsquigarrow_{\lambda^{\mathrm{v}}}$ to be the reflexive transitive closure of the reduction relation $\mapsto$. The identity ι is $\lambda\mathsf{x}.\mathsf{x}$, and the composition $\phi \cdot \psi$ is $\lambda\mathsf{x}.\phi(\psi(\mathsf{x}))$. The total basic combinatorial object $A_{\lambda^{\mathrm{v}}}$ forms a cbv combinatorial object:

$$* := () \quad \upsilon(\mathsf{t}) := (\lambda\mathsf{x}.())\mathsf{t}$$
$$\pi(\mathsf{t}) := (\lambda\mathsf{x}.\mathsf{fst}(\mathsf{x}))\mathsf{t} \quad \pi'(\mathsf{t}) := (\lambda\mathsf{x}.\mathsf{snd}(\mathsf{x}))\mathsf{t}$$
$$\langle \mathsf{v}, \mathsf{v}'\rangle := (\mathsf{v}, \mathsf{v}') \qquad \langle \phi, \phi'\rangle(\mathsf{t}) := (\lambda\mathsf{x}.(\phi(\mathsf{x}), \phi'(\mathsf{x})))\mathsf{t}$$
$$\gamma_{\mathsf{v},\phi}(\mathsf{t}) := (\mathsf{t}, \phi(\mathsf{v})) \quad \delta_{\mathsf{v}}(\mathsf{t}) := (\mathsf{v}, \mathsf{t})$$
$$\phi \star \mathsf{v} := \lambda\mathsf{y}.\phi(\mathsf{v}, \mathsf{y}) \qquad \varepsilon(\mathsf{t}) := (\lambda\mathsf{z}.(\mathsf{fst}\ \mathsf{z})(\mathsf{snd}\ \mathsf{z}))\mathsf{t}.$$

III. Evaluation and orthogonality

In this section, we introduce evaluation, indexed predicate and indexed orthogonality.

Definition III.1. An *evaluation* $\mathcal{E}$ for a cbv combinatorial object A is a set of homomorphisms on A such that

- For $\phi \in \Phi$ and $e \in \mathcal{E}$, the composition $e \circ \phi$ is in $\mathcal{E}$.
- For $x \in V$, $\phi \in \Phi$ and $e \in \mathcal{E}$, the compositions $e \circ \gamma_{x,\phi}$ and $e \circ \delta_x$ are in $\mathcal{E}$.

We use the last requirement in the definition of evaluation when pairing interacts with indexed orthogonality.

Definition III.2. An *indexed predicate* $\Omega = \{\Omega(n)\}_{n\geq 1}$ on a cbv combinatorial object A is a descending chain

$$\Omega(1) \supset \Omega(2) \supset \Omega(3) \supset \cdots$$

consisting of subsets of A such that for every $a \rightsquigarrow b$,

$$\forall n \geq 1.\ \exists k \geq 0.\ a \in \Omega(n+k) \iff b \in \Omega(n).$$

When we refer to $\Omega(0)$, we always suppose that $\Omega(0)$ is defined to be A. We say that a reduction $a \rightsquigarrow b$ is *strict* when

$$\forall e \in \mathcal{E}.\ \forall n \geq 0.\ e(b) \in \Omega(n) \implies e(a) \in \Omega(n+1),$$

and we write $a \dot{\rightsquigarrow} b$.

For an indexed predicate Ω, we define an indexed orthogonality relation $a \perp_n e$ between $a \in A$ and $e \in \mathcal{E}$ by

$$a \perp_n e \overset{\text{def}}{\iff} e(a) \in \Omega(n),$$

and we write $a \perp e$ when $a \perp_n e$ for all $n \geq 1$. We list fundamental properties on the indexed orthogonality.

Lemma III.1. *For $a, b \in A$ and $\phi \in \Phi$ and $e \in \mathcal{E}$, we have:*

1) $a \perp_{n+1} e \implies a \perp_n e$.
2) *If $a \dot{\rightsquigarrow} b$, then* $b \perp_n e \implies a \perp_{n+1} e$.
3) *If $a \rightsquigarrow b$, then* $b \perp_n e \implies a \perp_n e$.
4) *If $a \rightsquigarrow b$, then* $a \perp e \iff b \perp e$.
5) $\phi(a) \perp_n e \iff a \perp_n e \circ \phi$.

Example III.1. For the cbv combinatorial object $A_{\lambda^{\mathrm{v}}}$, we give an example of an evaluation $\mathcal{E}_{\lambda^{\mathrm{v}}}$ and an indexed predicate $\Omega_{\lambda^{\mathrm{v}}}$. We define $\mathcal{E}_{\lambda^{\mathrm{v}}}$ to be the set of evaluation contexts where we identified an evaluation context e with a homomorphism $\mathsf{t} \mapsto \mathsf{e}[\mathsf{t}]$. We define $\Omega_{\lambda^{\mathrm{v}}}$ by

$$\Omega_{\lambda^{\mathrm{v}}}(n) := \{\mathsf{t} \in A_{\lambda^{\mathrm{v}}} \mid \mathsf{t} \text{ is safe for } n\text{-steps}\}$$

where we say that t *is safe for n-steps* when t terminates to a value or there exists an n-step reduction sequence from t. For the indexed predicate $\Omega_{\lambda^{\mathrm{v}}}$, a reduction $\mathsf{t} \rightsquigarrow_{\lambda^{\mathrm{v}}} \mathsf{s}$ is strict if and only if $\mathsf{t} \rightsquigarrow_{\lambda^{\mathrm{v}}} \mathsf{s}$ consists of at least one reduction step.

With respect to the above orthogonality, $\mathsf{e} := ()[-] \in \mathcal{E}_{\lambda^{\mathrm{v}}}$ satisfies $\mathsf{v} \not\perp \mathsf{e}$ for any $\mathsf{v} \in V_{\lambda^{\mathrm{v}}}$. We generalize this situation to arbitrary evaluation.

Definition III.3. An evaluation $\mathcal{E}$ *admits error* when there exists $e \in \mathcal{E}$ such that for any $x \in V$, we have $x \not\perp e$.

IV. Partial equivalence relations

A *quasi reflexive relation on a set* X is a binary relation R on X such that if $(x, y) \in R$, then $(x, x) \in R$. For a binary relation R on a set X, we define $R^\diamond$ to be

$$\{(x, y) \in X \times X \mid (x, y), (y, x), (x, x), (y, y) \in R\}.$$

$R^\diamond$ is the largest quasi reflexive symmetric subrelation of R.

Let $\mathcal{E}$ be an evaluation for a cbv combinatorial object A, and let Ω be an indexed predicate on A. As in Section III, we define an indexed orthogonality $\perp_n$ and an orthogonality $\perp$ with respect to Ω and $\mathcal{E}$. So as to define a category of partial equivalence relations with respect to $A, \mathcal{E}$ and Ω, we introduce several notions. For an infinite descending chain consisting of binary relations on V:

$$X = \{X(1) \supset X(2) \supset X(3) \supset \cdots\},$$

we define a binary relation $X^{\perp}(n)$ on $\mathcal{E}$ to be

$$\bigcap_{i \le n} \{(e, e') \mid \forall (x, x') \in X(i).\ x \perp e \implies x' \perp_i e'\}.$$

By the definition of $X^{\perp}(n)$, we obtain an infinite descending chain $X^{\perp}(1) \supset X^{\perp}(2) \supset X^{\perp}(3) \supset \cdots$. Let $Y = \{Y(1) \supset Y(2) \supset Y(3) \supset \cdots\}$ be another infinite descending chain consisting of binary relations on V. For $\phi, \phi' \in \Phi$, we write $\phi \prec_{X,Y} \phi'$ when we have

$$\forall n \ge 1.\ \forall (x, x') \in X(n).\ \forall (e, e') \in Y^{\perp}(n).\ \phi(x) \perp e \implies \phi'(x') \perp_n e',$$

and we define $\sim_{X,Y}$ to be $(\prec_{X,Y})^{\diamond}$. We write $\approx_{X,Y}$ for the transitive closure of $\sim_{X,Y}$. Since $\sim_{X,Y}$ is quasi reflexive, $\phi \approx_{X,Y} \phi$ if and only if $\phi \sim_{X,Y} \phi$ if and only if $\phi \prec_{X,Y} \phi$.

We define a category $\mathbb{P}(A, \mathcal{E}, \Omega)$: an object is a descending chain $X = \{X(1) \supset X(2) \supset X(3) \supset \cdots\}$ consisting of binary relations on V, and a morphism $f : X \to Y$ is an equivalence class of $\approx_{X,Y}$, i.e., f is of the form $\{\phi \in \Phi \mid \phi \approx_{X,Y} \phi_0\}$ for some $\phi_0 \in \Phi$.

For legibility, we simply write $\mathbb{P}$ for $\mathbb{P}(A, \mathcal{E}, \Omega)$ when we can infer $A, \mathcal{E}$ and Ω from context. We say that $\phi \in \Phi$ *realizes* a morphism $f : X \to Y$ when ϕ is a representative of f. When ϕ realizes a morphism f from X to Y, we write $\phi \Vdash f : X \to Y$ or $\phi \Vdash X \to Y$, and we write $[\phi]$ for the morphism realized by ϕ. By the definition of $\approx_{X,Y}$, an element $\phi \in \Phi$ realizes a morphism if and only if $\phi \prec_{X,Y} \phi$. The identity on X is the morphism realized by ι. For a morphism $f : X \to Y$ realized by ϕ and a morphism $g : Y \to Z$ realized by ψ, the composition $g \circ f$ is the morphism realized by $\psi \cdot \phi$. In the following, when we refer to $X(0)$, we suppose that $X(0)$ is defined to be $V \times V$.

We say that a morphism $f : X \to Y$ is *strict* when f has a realizer ϕ such that

$$\forall n \ge 1.\ \forall (x, x') \in X(n).\ \exists (y, y') \in Y(n).\ \phi(x) \rightsquigarrow y \wedge \phi(x') \rightsquigarrow y'. \quad (1)$$

By the definition of cbv combinatorial objects, the identities are strict, and the composition preserves strict morphisms. We define $\mathbb{V}$ to be the wide subcategory of $\mathbb{P}$ determined by strict morphisms. We write $J : \mathbb{V} \to \mathbb{P}$ for the inclusion functor.

V. Categorical structures on $\mathbb{P}$

A. Unit type

We define an object $\top$ by $\top(n) := \{(*, *)\}$. Since we have $\upsilon(x) \rightsquigarrow *$ for every value x, we obtain a strict morphism from X to $\top$ realized by υ.

Proposition V.1. *$\top$ is a terminal object in $\mathbb{V}$.*

B. Pair type

For $\mathbb{P}$-objects X and Y, we define $(X \times Y)(n)$ to be

$$\{(\langle x, y\rangle, \langle x', y'\rangle) \mid (x, x') \in X(n) \wedge (y, y') \in Y(n)\}.$$

Then $X \times Y$ is a $\mathbb{P}$-object, and there are strict morphisms realized by $\pi \Vdash X \times Y \to X$ and $\pi' \Vdash X \times Y \to Y$.

Proposition V.2. *$X \times Y$ with $[\pi] : X \times Y \to X$ and $[\pi'] : X \times Y \to Y$ forms a product of X and Y in $\mathbb{V}$.*

For $\mathbb{V}$-morphisms $f : Z \to X$ and $g : Z \to Y$, the tupling $\langle f, g\rangle : Z \to X \times Y$ is the morphism realized by $\langle \phi, \psi\rangle$ where $\phi \Vdash f$ and $\psi \Vdash g$ are realizers that satisfy (1).

The inclusion functor $J : \mathbb{V} \to \mathbb{P}$ provides a categorical model of the computational lambda calculus. For the definition of Freyd category and closed Freyd category (Proposition V.4), see [14], [20].

Proposition V.3. *The unit type $\top$ and the pair type $X \times Y$ extend to a symmetric premonoidal structure on $\mathbb{P}$. The inclusion $J : \mathbb{V} \to \mathbb{P}$ forms a Freyd category with respect to the premonoidal structure.*

C. Arrow type

For objects X, Y and $i \ge 1$, we define $X \Rightarrow_i Y$ to be

$$\left\{(z, z') \;\middle|\; \begin{array}{r} \forall (x, x') \in X(i).\ \forall (e, e') \in Y^{\perp}(i). \\ \varepsilon\langle z, x\rangle \perp e \implies \varepsilon\langle z', x'\rangle \perp_i e' \end{array}\right\},$$

and we define $(X \Rightarrow Y)(n)$ to be $\bigcap_{i \le n} X \Rightarrow_i Y$.

Proposition V.4. *The inclusion functor $J : \mathbb{V} \to \mathbb{P}$ forms a closed Freyd category. Concretely, for every object X, the functor $J(-) \times X : \mathbb{V} \to \mathbb{P}$ is a left adjoint functor.*

A unit of the adjunction is the morphism from $(X \Rightarrow Y) \times X$ to Y realized by ε. For a $\mathbb{P}$-morphism $f : Y \times X \to Z$ realized by ϕ, the unique $\mathbb{V}$-morphism $f^{\wedge} : Y \to X \Rightarrow Z$ is realized by a homomorphism in Φ that computes $\phi \star (-)$.

D. Sum type

Lemma V.1. *There exists computable maps $k, k' : V \to V$ such that for any ϕ and ψ in Φ,*

$$\exists \chi \in \Phi.\ \forall x \in V.\ \chi(k(x)) \rightsquigarrow \phi(x) \wedge \chi(k'(x)) \rightsquigarrow \psi(x). \quad (2)$$

For $\mathbb{P}$-objects X and Y, we define a $\mathbb{P}$-object $X + Y$ by

$$(X + Y)(n) := \{(k(x), k(x')) \mid (x, x') \in X(n)\} \cup \{(k'(y), k'(y')) \mid (y, y') \in Y(n)\}.$$

Proposition V.5. *For $\kappa, \kappa' \in \Phi$ that compute k and k' respectively, we have $\kappa \Vdash X \to X + Y$ and $\kappa' \Vdash Y \to X + Y$. The object $X + Y$ with $[\kappa] : X \to X + Y$ and $[\kappa'] : Y \to X + Y$ forms a coproduct of X and Y in $\mathbb{V}$.*

For $f : X \to Z$ realized by ϕ and $g : Y \to Z$ realized by ψ, the cotupling $[f, g] : X + Y \to Z$ is the morphism realized by χ that is associated with ϕ and ψ in (2).

E. Fixed point operator

So as to obtain a fixed point operator, we need a "call-by-value Y-combinator" with respect to strict reductions.

Requirement V.1. For each $\phi \in \Phi$, there exists a computable map $Y_\phi : V \to V$ such that

$$\forall \phi \in \Phi.\ \forall x, y \in V.\ \varepsilon\langle Y_\phi(x), y\rangle \dot{\rightsquigarrow} \phi\langle\langle x, Y_\phi(x)\rangle, y\rangle.$$

If "β-reduction" in Definition II.3 is strict, then the cbv combinatorial object has Y_ϕ that satisfies Requirement V.1, see [17]. For example, the cbv combinatorial object $A_{\lambda^{\mathrm{v}}}$ in Section II satisfies the above requirement.

Proposition V.6. *If a cbv combinatorial object A satisfies Requirement V.1, then for a $\mathbb{P}$-morphism $f : X \times (Y \Rightarrow Z) \times Y \to Z$, there is a $\mathbb{V}$-morphism $f^\dagger : X \to Y \Rightarrow Z$ such that*

$$f^\wedge \circ \langle \mathrm{id}_X, f^\dagger\rangle = f^\dagger$$

where $f^\wedge : X \times (Y \Rightarrow Z) \to Y \Rightarrow Z$ is the transpose of f, and $\langle \mathrm{id}_X, f^\dagger\rangle : X \to X \times (Y \Rightarrow Z)$ is the tupling in $\mathbb{V}$.

For $\phi \Vdash f : X \times (Y \Rightarrow Z) \times Y \to Z$, the $\mathbb{V}$-morphism $f^\dagger$ is realized by μ_ϕ that computes Y_ϕ. Here, we assume that $\times$ is left associative, i.e., $X \times (Y \Rightarrow Z) \times Y$ is an abbreviation of $(X \times (Y \Rightarrow Z)) \times Y$. We note that $f^\dagger$ is independent of our choice of μ_ϕ that computes Y_ϕ in Requirement V.1.

F. Recursive type

Let $\mathrm{ob}(\mathbb{P})$ be the set of $\mathbb{P}$-objects. For a map F on $\mathrm{ob}(\mathbb{P})$ and $X \in \mathbb{P}$, we write $FX(n)$ for $(FX)(n)$. Terminologies in the following definition are from [4].

Definition V.1. Let F be a map on $\mathrm{ob}(\mathbb{P})$. We say that F is *well founded* when for all $\mathbb{P}$-objects X and Y, we have

$$\forall n \geq 1.\ (\forall i < n.\ X(i) = Y(i)) \Longrightarrow FX(n) = FY(n).$$

We say that F is *non-expansive* when for all $\mathbb{P}$-objects X and Y, we have

$$\forall n \geq 1.\ (\forall i \leq n.\ X(i) = Y(i)) \Longrightarrow FX(n) = FY(n).$$

More generally, we say that $F : \mathrm{Ob}(\mathbb{P})^n \to \mathrm{Ob}(\mathbb{P})$ is non-expansive when F is non-expansive for each component, i.e., $F(X_1, \cdots, X_{i-1}, -, X_{i+1}, \cdots, X_n)$ is non-expansive for each $i = 1, 2, \cdots, n$.

1) Recursive types for well founded maps: First, we give a fixed point for a well founded map F on $\mathrm{ob}(\mathbb{P})$.

Lemma V.2. *For any $\mathbb{P}$-objects X, Y and a natural number $n \geq 1$, we have $(F^n X)(n) = (F^n Y)(n)$.*

We define a $\mathbb{P}$-object μF by

$$\mu F(n) := F^n 0(n)$$

where 0 is a $\mathbb{P}$-object given by $\emptyset \supset \emptyset \supset \emptyset \supset \cdots$. The object μF is well defined by Lemma V.2:

$$\mu F(n+1) \subset F^{n+1}0(n) = F^n 0(n) = \mu F(n).$$

Theorem V.1. *Every well founded F has a unique equi-fixed point $\mu F = F(\mu F)$. Uniqueness means that if $X = FX$, then $\mu F = X$.*

2) Recursive types for non-expansive maps: Next, we consider non-expansive maps on $\mathrm{ob}(\mathbb{P})$. We need to assume a condition so as to translate non-expansive maps into well founded maps.

Requirement V.2. There exists a computable map $\ell : V \to V$ such that

$$\exists \phi \in \Phi.\ \forall x \in V.\ \phi(\ell(x)) \dot{\rightsquigarrow} x. \tag{3}$$

If "first projection" in Definition II.1 is strict, then we can find ℓ that satisfies Requirement V.2, see [17]. For example, the cbv combinatorial object $A_{\lambda^{\mathrm{v}}}$ satisfies the above requirement.

For an object X, we define an object $X^\uparrow$ by

$$X^\uparrow(n) := \{(\ell(x), \ell(x')) \mid (x, x') \in X(n-1)\}.$$

By the definition, $(-)^\uparrow$ is well founded.

Lemma V.3. *X is isomorphic to $X^\uparrow$ in $\mathbb{P}$.*

We take ψ that computes ℓ and ϕ associated with ℓ in Requirement V.2. Then $\psi \Vdash X \to X^\uparrow$ and $\phi \Vdash X^\uparrow \to X$ constitute the isomorphism. We note that ϕ is not strict. Especially, ψ and ϕ do not constitute an isomorphism in $\mathbb{V}$.

For a map F on $\mathrm{ob}(\mathbb{P})$, we define $F^\uparrow X$ to be $(FX)^\uparrow$. By the definition of $(-)^\uparrow$, if F is non-expansive, then $F^\uparrow$ is well founded.

Corollary V.1. *If a cbv combinatorial object A satisfies Requirement V.2, then every non-expansive map F on $\mathbb{P}$ has an iso-fixed point $\mu F^\uparrow = F^\uparrow(\mu F^\uparrow) \cong F(\mu F^\uparrow)$.*

All the type constructions such as $(-) \times (-)$, $(-) \Rightarrow (-)$ and $(-) + (-)$ are non-expansive maps from $\mathrm{ob}(\mathbb{P})^2$ to $\mathrm{ob}(\mathbb{P})$. Therefore, every map F on $\mathrm{ob}(\mathbb{P})$ inductively constructed from these operations has an iso-fixed point by Corollary V.1. So as to obtain equi-recursive types, we only need to compose our definition of pair type, arrow type and sum type with $(-)^\uparrow$, namely $((-) \times (-))^\uparrow$, $((-) \Rightarrow (-))^\uparrow$ and $((-) + (-))^\uparrow$. These type constructions are isomorphic to the original constructions. For any map on $\mathrm{ob}(\mathbb{P})$ inductively constructed from the well founded data types, we obtain equi-recursive types by Theorem V.1.

In this paper, we choose non-expansive data types and iso-recursive types because the definitions of non-expansive pair type, arrow type and sum type have no bureaucratic delay of indices. In Section VI, we give a call-by-value polymorphic lambda calculus with iso-recursive types, which corresponds to our choice of non-expansive data types.

VI. Semantics for Language $\Lambda_{\exists\mu}$

A. Syntax and operational semantics

We describe the syntax of a call-by-value polymorphic language with existential types and iso-recursive types. Types,

terms and values are given by the following BNF.

$$\begin{array}{lll}
\text{Type} & \mathsf{A} ::= & \mathsf{X} \mid \mathsf{Unit} \mid \mathsf{A} \times \mathsf{A} \mid \mathsf{A} + \mathsf{A} \mid \mathsf{A} \Rightarrow \mathsf{A} \\
& & \mid \mu \mathsf{X}.\mathsf{A} \mid \forall \mathsf{X}.\mathsf{A} \mid \exists \mathsf{X}.\mathsf{A} \\
\text{Term} & \mathsf{M} ::= & \mathsf{x} \mid () \mid (\mathsf{M}, \mathsf{M}) \mid \mathsf{fst}(\mathsf{M}) \mid \mathsf{snd}(\mathsf{M}) \\
& & \mid \mathsf{inl}_{\mathsf{A},\mathsf{B}}(\mathsf{M}) \mid \mathsf{inr}_{\mathsf{A},\mathsf{B}}(\mathsf{M}) \\
& & \mid \mathsf{case}\ \mathsf{M}\ \mathsf{of}\ (\mathsf{x} \mapsto \mathsf{M} \mid \mathsf{y} \mapsto \mathsf{M}) \\
& & \mid \mathsf{MM} \mid \lambda \mathsf{x} : \mathsf{A}.\mathsf{M} \mid \mathsf{rec}_{\mathsf{A},\mathsf{B}}(\mathsf{f},\mathsf{x}).\mathsf{M} \\
& & \mid \mathsf{fld}_{\mu \mathsf{X}.\mathsf{A}}(\mathsf{M}) \mid \mathsf{ufd}(\mathsf{M}) \mid \Lambda \mathsf{X}.\mathsf{M} \mid \mathsf{MA} \\
& & \mid \mathsf{pack}_{\mathsf{A},\exists \mathsf{X}.\mathsf{B}}(\mathsf{M}) \mid \mathsf{unpack}\ \mathsf{M}\ \mathsf{as}\ \mathsf{X},\mathsf{x}\ \mathsf{in}\ \mathsf{N} \\
\text{Value} & \mathsf{V} ::= & \mathsf{x} \mid () \mid (\mathsf{V}, \mathsf{V}) \mid \mathsf{inl}_{\mathsf{A},\mathsf{B}}(\mathsf{V}) \mid \mathsf{inr}_{\mathsf{A},\mathsf{B}}(\mathsf{V}) \\
& & \mid \lambda \mathsf{x} : \mathsf{A}.\mathsf{M} \mid \mathsf{rec}_{\mathsf{A},\mathsf{B}}(\mathsf{f},\mathsf{x}).\mathsf{M} \\
& & \mid \mathsf{fld}_{\mu \mathsf{X}.\mathsf{A}}(\mathsf{V}) \mid \Lambda \mathsf{X}.\mathsf{M} \mid \mathsf{pack}_{\mathsf{A},\exists \mathsf{X}.\mathsf{B}}(\mathsf{V})
\end{array}$$

A *type context* is a finite sequence of type variables in which no type variable appears more than twice. We use $\Theta, \Theta', \cdots$ for type environments. Well formed types are given by the following rules:

$$\frac{}{\mathsf{X}_1, \cdots, \mathsf{X}_n \vdash \mathsf{X}_i} \qquad \frac{}{\Theta \vdash \mathsf{Unit}}$$

$$\frac{\Theta \vdash \mathsf{A} \quad \Theta \vdash \mathsf{B}}{\Theta \vdash \mathsf{A} \times \mathsf{B}} \qquad \frac{\Theta \vdash \mathsf{A} \quad \Theta \vdash \mathsf{B}}{\Theta \vdash \mathsf{A} \Rightarrow \mathsf{B}} \qquad \frac{\Theta \vdash \mathsf{A} \quad \Theta \vdash \mathsf{B}}{\Theta \vdash \mathsf{A} + \mathsf{B}}$$

$$\frac{\Theta, \mathsf{X} \vdash \mathsf{A}}{\Theta \vdash \mu \mathsf{X}.\mathsf{A}} \qquad \frac{\Theta, \mathsf{X} \vdash \mathsf{A}}{\Theta \vdash \forall \mathsf{X}.\mathsf{A}} \qquad \frac{\Theta, \mathsf{X} \vdash \mathsf{A}}{\Theta \vdash \exists \mathsf{X}.\mathsf{A}}$$

We give typing rules in Figure 1 and the big step call-by-value operational semantics in Figure 2. Both the typing rules and the operational semantics are standard. A *term context* is a sequence of pairs consisting of a term variable and a type in which no term variable appears more than twice. We use $\Gamma, \Gamma', \cdots$ for term contexts. We write $\Theta \vdash \Gamma$ when all the free type variables in Γ are in Θ, and we write $\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A}$ when the judgment is derivable from the typing rules. For a closed term M, we write $\mathsf{M} \Downarrow$ when there is a value V such that $\mathsf{M} \Downarrow \mathsf{V}$, and we write $\mathsf{M} \Uparrow$ when there is no value V such that $\mathsf{M} \Downarrow \mathsf{V}$.

B. Interpretation of $\Lambda_{\exists\mu}$

We have shown that $J : \mathbb{V} \to \mathbb{P}$ is a closed Freyd category with coproducts. We also showed that J provides a semantics of recursion and recursive types under Requirement V.1 and Requirement V.2. So as to interpret polymorphism, we extend J to a category of "uniform families" of J like various PER-models of polymorphic lambda calculi. For categorical semantics and PER-models of polymorphic lambda calculi, see [19].

We take a cbv combinatorial object A, an evaluation $\mathcal{E}$ and an indexed predicate Ω. As in Section IV, we define a quasi reflexive relation $\sim_{X,Y}$ for each $X, Y \in \mathbb{P}$. Furthermore, we suppose that A satisfies Requirement V.1 and Requirement V.2.

Let n be a natural number. For non-expansive maps $F, G : \mathrm{ob}(\mathbb{P})^n \to \mathrm{ob}(\mathbb{P})$ and $\phi, \phi' \in \Phi$, we write $\phi \sim_{F,G} \phi'$ when $\phi \sim_{F(\vec{X}),G(\vec{X})} \phi'$ for all $\vec{X} \in \mathrm{ob}(\mathbb{P})^n$. Since $\sim_{F(\vec{X}),G(\vec{X})}$ is a quasi reflexive symmetric relation, $\sim_{F,G}$ is also quasi reflexive symmetric. We define $\approx_{F,G}$ to be the transitive closure of $\sim_{F,G}$. We write $\phi \Vdash F \to G$ when $\phi \Vdash F(\vec{X}) \to G(\vec{X})$ for all $\vec{X} \in \mathrm{ob}(\mathbb{P})^n$.

We define a category $\mathrm{Fam}(\mathbb{P})_n$: an object is a non-expansive map from $\mathrm{ob}(\mathbb{P})^n$ to $\mathrm{ob}(\mathbb{P})$, and a morphism $f : F \to G$ is an equivalence class of $\phi \Vdash F \to G$ modulo $\approx_{F,G}$. We say that ϕ *realizes* a morphism $f : F \to G$ when ϕ is a representative of f, and we write $\phi \Vdash f : F \to G$ or simply $\phi \Vdash f$. We write $[\phi]$ for the morphism realized by ϕ. We say that a $\mathrm{Fam}(\mathbb{P})_n$-morphism $f : F \to G$ is *strict* when f has a realizer ϕ such that

$$\forall \vec{X} \in \mathrm{ob}(\mathbb{P})^n.\ \forall m \geq 1.\ \forall (x, x') \in F(\vec{X})(m).\\ \exists (y, y') \in G(\vec{X})(m).\ \phi(x) \rightsquigarrow y \wedge \phi(x') \rightsquigarrow y'.$$

We define a category $\mathrm{Fam}(\mathbb{V})_n$ to be the wide subcategory of $\mathrm{Fam}(\mathbb{P})_n$ consisting of strict morphisms, and we write $J_n : \mathrm{Fam}(\mathbb{V})_n \to \mathrm{Fam}(\mathbb{P})_n$ for the inclusion functor. We can naturally extend type constructions of pair type, arrow type and sum type given in Section V to constructions on $\mathrm{Fam}(\mathbb{P})_n$-objects, and we can show that $J_n : \mathrm{Fam}(\mathbb{V})_n \to \mathrm{Fam}(\mathbb{P})_n$ is a closed Freyd category and that $\mathrm{Fam}(\mathbb{V})_n$ has coproducts by the same argument.

For $F \in \mathrm{Fam}(\mathbb{P})_{n+1}$ and $G \in \mathrm{Fam}(\mathbb{P})_n$, we define $F[G] \in \mathrm{Fam}(\mathbb{P})_n$ to be a map

$$(X_1, \cdots, X_n) \mapsto F(X_1, \cdots, X_n, G(X_1, \cdots, X_n)).$$

Since both F and G are non-expansive, $F[G]$ is also non-expansive. We note that the operation $(-)[G]$ preserves the closed Freyd structures on the nose.

In terms of fibred category theory [19], we have defined fibrations $p : \mathrm{Fam}(\mathbb{P}) \to \mathbb{B}$ and $q : \mathrm{Fam}(\mathbb{V}) \to \mathbb{B}$ with a fibred functor $J : q \to p$ where the base category $\mathbb{B}$ is: objects are natural numbers, and a morphism $F : n \to m$ is an m-tuple of non-expansive maps $\{F_i : \mathrm{ob}(\mathbb{P})^n \to \mathrm{ob}(\mathbb{P})\}_{i=1,2,\cdots,m}$. Each fibre category $J_n : \mathrm{Fam}(\mathbb{V})_n \to \mathrm{Fam}(\mathbb{P})_n$ is a closed Freyd category, and every reindexing functor preserves the structures on the nose.

We interpret a type $\mathsf{X}_1, \cdots, \mathsf{X}_n \vdash \mathsf{A}$ by an object $[\![\mathsf{A}]\!]$ in $\mathrm{Fam}(\mathbb{P})_n$. Interpretation of $\Lambda_{\exists\mu}$ is in the following form:

$$\frac{\mathsf{X}_1, \cdots, \mathsf{X}_n \mid \mathsf{x}_1 : \mathsf{A}_1, \cdots, \mathsf{x}_m : \mathsf{A}_m \vdash \mathsf{M} : \mathsf{A}}{[\![\mathsf{M}]\!] : [\![\mathsf{A}_1]\!] \times \cdots \times [\![\mathsf{A}_m]\!] \to [\![\mathsf{A}]\!] \text{ in } \mathrm{Fam}(\mathbb{P})_n}.$$

We describe interpretation of recursion, recursive types, existential types and universal types below. Interpretation of the remaining fragment is in the standard way, see [17]. For the interpretation of the call-by-value lambda calculus in a closed Freyd category, see [14] for example.

1) Recursion: For a morphism $f : F \times (G \Rightarrow H) \times G \to H$ in $\mathrm{Fam}(\mathbb{P})_n$, let ϕ be a realizer of f, and we take μ_ϕ that computes Y_ϕ. Then μ_ϕ realizes a morphism $f^\dagger : F \to (G \Rightarrow H)$ in $\mathrm{Fam}(\mathbb{V})_n$ such that

$$f^\wedge \circ \langle \mathrm{id}_F, f^\dagger \rangle = f^\dagger$$

where $f^\wedge : F \times (G \Rightarrow H) \to G \Rightarrow H$ is the transpose of f, and $\langle \mathrm{id}_F, f^\dagger \rangle : F \to F \times (G \Rightarrow H)$ is the tupling in $\mathrm{Fam}(\mathbb{V})_n$. We interpret recursion $[\![\mathsf{rec}_{\mathsf{A},\mathsf{B}}(\mathsf{f},\mathsf{x}).\mathsf{M}]\!]$ by $[\![\mathsf{M}]\!]^\dagger$.

$$\frac{\Theta \vdash \Gamma, \mathsf{x} : \mathsf{A}, \Gamma'}{\Theta \mid \Gamma, \mathsf{x} : \mathsf{A}, \Gamma' \vdash \mathsf{x} : \mathsf{A}} \quad \frac{\Theta \vdash \Gamma}{\Theta \mid \Gamma \vdash () : \mathsf{Unit}} \quad \frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A} \quad \Theta \mid \Gamma \vdash \mathsf{N} : \mathsf{B}}{\Theta \mid \Gamma \vdash (\mathsf{M}, \mathsf{N}) : \mathsf{A} \times \mathsf{B}} \quad \frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A} \times \mathsf{B}}{\Theta \mid \Gamma \vdash \mathsf{fst}(\mathsf{M}) : \mathsf{A}} \quad \frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A} \times \mathsf{B}}{\Theta \mid \Gamma \vdash \mathsf{snd}(\mathsf{M}) : \mathsf{B}}$$

$$\frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A}}{\Theta \mid \Gamma \vdash \mathsf{inl}_{\mathsf{A},\mathsf{B}}(\mathsf{M}) : \mathsf{A} + \mathsf{B}} \quad \frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{B}}{\Theta \mid \Gamma \vdash \mathsf{inr}_{\mathsf{A},\mathsf{B}}(\mathsf{M}) : \mathsf{A} + \mathsf{B}} \quad \frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A} + \mathsf{B} \quad \Theta \mid \Gamma, \mathsf{x} : \mathsf{A} \vdash \mathsf{N} : \mathsf{C} \quad \Theta \mid \Gamma, \mathsf{y} : \mathsf{B} \vdash \mathsf{N}' : \mathsf{C}}{\Theta \mid \Gamma \vdash \mathsf{case}\ \mathsf{M}\ \mathsf{of}\ (\mathsf{x} \mapsto \mathsf{N} \mid \mathsf{y} \mapsto \mathsf{N}') : \mathsf{C}}$$

$$\frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A} \Rightarrow \mathsf{B} \quad \Theta \mid \Gamma \vdash \mathsf{N} : \mathsf{A}}{\Theta \mid \Gamma \vdash \mathsf{MN} : \mathsf{B}} \quad \frac{\Theta \mid \Gamma, \mathsf{x} : \mathsf{A} \vdash \mathsf{M} : \mathsf{B}}{\Theta \mid \Gamma \vdash \lambda \mathsf{x} : \mathsf{A}.\mathsf{M} : \mathsf{A} \Rightarrow \mathsf{B}} \quad \frac{\Theta \mid \Gamma, \mathsf{f} : \mathsf{A} \Rightarrow \mathsf{B}, \mathsf{x} : \mathsf{A} \vdash \mathsf{M} : \mathsf{B}}{\Theta \mid \Gamma \vdash \mathsf{rec}_{\mathsf{A},\mathsf{B}}(\mathsf{f}, \mathsf{x}).\mathsf{M} : \mathsf{A} \Rightarrow \mathsf{B}}$$

$$\frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A}[\mu\mathsf{X}.\mathsf{A}/\mathsf{X}]}{\Theta \mid \Gamma \vdash \mathsf{fld}_{\mu\mathsf{X}.\mathsf{A}}(\mathsf{M}) : \mu\mathsf{X}.\mathsf{A}} \quad \frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mu\mathsf{X}.\mathsf{A}}{\Theta \mid \Gamma \vdash \mathsf{ufd}(\mathsf{M}) : \mathsf{A}[\mu\mathsf{X}.\mathsf{A}/\mathsf{X}]} \quad \frac{\Theta, \mathsf{X} \mid \Gamma \vdash \mathsf{M} : \mathsf{A} \quad \Theta \vdash \Gamma}{\Theta \mid \Gamma \vdash \Lambda\mathsf{X}.\mathsf{M} : \forall\mathsf{X}.\mathsf{A}} \quad \frac{\Theta \mid \Gamma \vdash \mathsf{M} : \forall\mathsf{X}.\mathsf{A} \quad \Theta \vdash \mathsf{B}}{\Theta \mid \Gamma \vdash \mathsf{MB} : \mathsf{A}[\mathsf{B}/\mathsf{X}]}$$

$$\frac{\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A}[\mathsf{B}/\mathsf{X}]}{\Theta \mid \Gamma \vdash \mathsf{pack}_{\mathsf{B},\exists\mathsf{X}.\mathsf{A}}(\mathsf{M}) : \exists\mathsf{X}.\mathsf{A}} \quad \frac{\Theta \mid \Gamma \vdash \mathsf{M} : \exists\mathsf{X}.\mathsf{A} \quad \Theta, \mathsf{X} \mid \Gamma, \mathsf{x} : \mathsf{A} \vdash \mathsf{N} : \mathsf{B} \quad \Theta \vdash \Gamma \quad \Theta \vdash \mathsf{B}}{\Theta \mid \Gamma \vdash \mathsf{unpack}\ \mathsf{M}\ \mathsf{in}\ \mathsf{X}, \mathsf{x}\ \mathsf{in}\ \mathsf{N} : \mathsf{B}}$$

Fig. 1. Typing rules of $\Lambda_{\exists\mu}$

2) Recursive type: Since every object $F \in \mathrm{Fam}(\mathbb{P})_{n+1}$ is non-expansive, we can define $\mathrm{fix}_n(F) \in \mathrm{Fam}(\mathbb{P})_n$ by

$$\mathrm{fix}_n(F)(X_1, \cdots, X_n) := \mu(F^{\uparrow}(X_1, \cdots, X_n, -)),$$

which is isomorphic to $F[\mathrm{fix}_n(F)](X_1, \cdots, X_n)$. Since the isomorphism has realizers that are irrelevant to $X_1, \cdots, X_n$, we obtain an isomorphism

$$\theta_F : \mathrm{fix}_n(F) \xrightarrow{\cong} F[\mathrm{fix}_n(F)]$$

in $\mathrm{Fam}(\mathbb{P})_n$. We interpret $\mu\mathsf{X}.\mathsf{A}$ by $\mathrm{fix}_n(\llbracket \mathsf{A} \rrbracket)$, and we interpret terms associated with recursive types as follows:

$$\llbracket \mathsf{fld}_{\mu\mathsf{X}.\mathsf{A}}(\mathsf{M}) \rrbracket := \theta^{-1}_{\llbracket \mathsf{A} \rrbracket} \circ \llbracket \mathsf{M} \rrbracket \qquad \llbracket \mathsf{ufd}(\mathsf{M}) \rrbracket := \theta_{\llbracket \mathsf{A} \rrbracket} \circ \llbracket \mathsf{M} \rrbracket.$$

3) Existential type: We define a reindexing functor $\pi_n^* : \mathrm{Fam}(\mathbb{V})_n \to \mathrm{Fam}(\mathbb{V})_{n+1}$ by $(\pi_n^* G)(X_1, \cdots, X_{n+1}) := G(X_1, \cdots, X_n)$.

Lemma VI.1. *There exists a computable map $\imath : V \to V$ such that*

$$\exists \phi \in \Phi.\ \forall x \in V.\ \phi(\imath(x)) \rightsquigarrow x.$$

For $F \in \mathrm{Fam}(\mathbb{P})_{n+1}$, we define $\exists_n F \in \mathrm{Fam}(\mathbb{P})_n$ by

$$\begin{aligned}&(y, y') \in (\exists_n F(X_1, \cdots, X_n))(m)\\ &\overset{\text{def}}{\iff} \exists X \in \mathrm{ob}(\mathbb{P}).\ \exists (x, x') \in F(X_1, \cdots, X_n, X)(m).\\ &\qquad\qquad (y, y') = (\imath(x), \imath(x')).\end{aligned}$$

Proposition VI.1. *π_n^* is a right adjoint functor, whose unit is a $\mathbb{V}$-morphism $[\eta] : F \to \pi_n^* \exists_n F$ where $\eta \in \Phi$ computes $\imath$. The adjunction $\exists_n \dashv \pi_n^*$ satisfies Beck-Chevalley condition.*

Since each $J_n : \mathrm{Fam}(\mathbb{V})_n \to \mathrm{Fam}(\mathbb{P})_n$ is a closed Freyd category, $\exists_n$ satisfies *Frobenius property*: for every $F, H \in \mathrm{Fam}(\mathbb{P})_n$ and $G \in \mathrm{Fam}(\mathbb{P})_{n+1}$, we have

$$\mathrm{Fam}(\mathbb{P})_{n+1}(\pi_n^* F \times G, \pi_n^* H) \cong \mathrm{Fam}(\mathbb{P})_n(F \times \exists_n G, H).$$

For a type $\mathsf{X}_1, \cdots, \mathsf{X}_{n+1} \vdash A$, we interpret $\exists\mathsf{X}.\mathsf{A}$ by $\exists_n \llbracket \mathsf{A} \rrbracket$. We interpret term constructions for existential types via the adjunction in Proposition VI.1 and Frobenius property.

4) Universal type:

Lemma VI.2. *There exists $\xi \in \Phi$ such that*

$$\begin{aligned}&\forall \phi \in \Phi.\ \exists f : V \to V.\ \forall x \in V.\\ &\qquad\qquad f \textit{ is computable} \wedge \xi(f(x)) \rightsquigarrow \phi(x).\end{aligned}$$

For $F \in \mathrm{Fam}(\mathbb{P})_{n+1}$, we define $\forall_n F \in \mathrm{Fam}(\mathbb{P})_n$ by

$$\begin{aligned}&(z, z') \in (\forall_n F(X_1, \cdots, X_n))(m)\\ &\overset{\text{def}}{\iff} \quad \forall i \le m.\ \forall X \in \mathrm{ob}(\mathbb{P}).\\ &\qquad \forall (e, e') \in (F(X_1, \cdots, X_n, X))^{\perp}(i).\\ &\qquad \xi(z) \perp e \implies \xi(z') \perp_i e'.\end{aligned}$$

By the definition of $\forall_n$, we have $\xi \Vdash J_{n+1}\pi_n^* \forall_n F \to F$.

Proposition VI.2. *The functor $J_{n+1}\pi_n^* : \mathrm{Fam}(\mathbb{V})_n \to \mathrm{Fam}(\mathbb{P})_{n+1}$ is a left adjoint functor, whose counit is $[\xi] : J_{n+1}\pi_n^* \forall_n F \to F$. The adjunction $J_{n+1}\pi_n^* \dashv \forall_n$ satisfies Beck-Chevalley condition.*

For a type $\mathsf{X}_1, \cdots, \mathsf{X}_{n+1} \vdash A$, we interpret $\forall\mathsf{X}.\mathsf{A}$ by $\forall_n \llbracket \mathsf{A} \rrbracket$, and we interpret term constructions for universal types via the adjunction $J_{n+1}\pi_n^* \dashv \forall_n$.

C. Soundness and adequacy

Interpretation of $\Lambda_{\exists\mu}$ has parameters: $A, \mathcal{E}, \Omega$ and

$$k \quad k' \quad \imath \quad \xi \quad Y_\phi \quad \ell$$

in Lemma V.1, Lemma VI.1, Lemma VI.2, Requirement V.1 and Requirement V.2. Each choice of $k, k', \imath, \xi, Y_\phi$ and ℓ determines a convention of data representation for sum types, existential types, universal types, recursion and recursive types. The following theorems are independent of concrete choices of $k, k', \imath, \xi, Y_\phi$ and ℓ.

Theorem VI.1 (Soundness)**.** *For any closed term $\Theta \mid - \vdash \mathsf{M} : \mathsf{A}$, if $\mathsf{M} \Downarrow \mathsf{V}$, then $\llbracket \mathsf{M} \rrbracket = \llbracket \mathsf{V} \rrbracket$.*

By means of logical relation, we can show that our interpretation adequately captures the big step operational semantics. For a proof, see [17].

$$\frac{}{\mathsf{V} \Downarrow \mathsf{V}} \quad \frac{\mathsf{M} \Downarrow \mathsf{U} \quad \mathsf{N} \Downarrow \mathsf{V}}{(\mathsf{M},\mathsf{N}) \Downarrow (\mathsf{U},\mathsf{V})} \quad \frac{\mathsf{M} \Downarrow (\mathsf{U},\mathsf{V})}{\mathsf{fst}(\mathsf{M}) \Downarrow \mathsf{U}} \quad \frac{\mathsf{M} \Downarrow (\mathsf{U},\mathsf{V})}{\mathsf{snd}(\mathsf{M}) \Downarrow \mathsf{V}} \quad \frac{\mathsf{M} \Downarrow \mathsf{V}}{\mathsf{inl}_{\mathsf{A},\mathsf{B}}(\mathsf{M}) \Downarrow \mathsf{inl}_{\mathsf{A},\mathsf{B}}(\mathsf{V})} \quad \frac{\mathsf{M} \Downarrow \mathsf{V}}{\mathsf{inr}_{\mathsf{A},\mathsf{B}}(\mathsf{M}) \Downarrow \mathsf{inr}_{\mathsf{A},\mathsf{B}}(\mathsf{V})}$$

$$\frac{\mathsf{M} \Downarrow \mathsf{inl}_{\mathsf{A},\mathsf{B}}(\mathsf{V}) \quad \mathsf{N}[\mathsf{V}/\mathsf{x}] \Downarrow \mathsf{U}}{\mathsf{case}\ \mathsf{M}\ \mathsf{of}\ (\mathsf{x} \mapsto \mathsf{N} \mid \mathsf{y} \mapsto \mathsf{N}') \Downarrow \mathsf{U}} \quad \frac{\mathsf{M} \Downarrow \mathsf{inr}_{\mathsf{A},\mathsf{B}}(\mathsf{V}) \quad \mathsf{N}'[\mathsf{V}/\mathsf{x}] \Downarrow \mathsf{U}}{\mathsf{case}\ \mathsf{M}\ \mathsf{of}\ (\mathsf{x} \mapsto \mathsf{N} \mid \mathsf{y} \mapsto \mathsf{N}') \Downarrow \mathsf{U}} \quad \frac{\mathsf{M} \Downarrow \lambda \mathsf{x} : \mathsf{A}.\mathsf{M}' \quad \mathsf{N} \Downarrow \mathsf{V} \quad \mathsf{M}'[\mathsf{V}/\mathsf{x}] \Downarrow \mathsf{U}}{\mathsf{MN} \Downarrow \mathsf{U}}$$

$$\frac{\mathsf{M} \Downarrow \mathsf{rec}_{\mathsf{A},\mathsf{B}}(\mathsf{f},\mathsf{x}).\mathsf{M}' \quad \mathsf{N} \Downarrow \mathsf{V} \quad \mathsf{M}'[\mathsf{rec}_{\mathsf{A},\mathsf{B}}(\mathsf{f},\mathsf{x}).\mathsf{M}'/\mathsf{f},\mathsf{V}/\mathsf{x}] \Downarrow \mathsf{U}}{\mathsf{MN} \Downarrow \mathsf{U}} \quad \frac{\mathsf{M} \Downarrow \mathsf{V}}{\mathsf{fld}_{\mu\mathsf{X}.\mathsf{A}}(\mathsf{M}) \Downarrow \mathsf{fld}_{\mu\mathsf{X}.\mathsf{A}}(\mathsf{V})} \quad \frac{\mathsf{M} \Downarrow \mathsf{fld}_{\mu\mathsf{X}.\mathsf{A}}(\mathsf{V})}{\mathsf{ufd}(\mathsf{M}) \Downarrow \mathsf{V}}$$

$$\frac{\mathsf{M} \Downarrow \Lambda\mathsf{X}.\mathsf{M}' \quad \mathsf{M}'[\mathsf{A}/\mathsf{X}] \Downarrow \mathsf{V}}{\mathsf{MA} \Downarrow \mathsf{V}} \quad \frac{\mathsf{M} \Downarrow \mathsf{V}}{\mathsf{pack}_{\mathsf{B},\exists\mathsf{X}.\mathsf{A}}(\mathsf{M}) \Downarrow \mathsf{pack}_{\mathsf{B},\exists\mathsf{X}.\mathsf{A}}(\mathsf{V})} \quad \frac{\mathsf{M} \Downarrow \mathsf{pack}_{\mathsf{A},\exists\mathsf{X}.\mathsf{B}}(\mathsf{V}) \quad \mathsf{N}[\mathsf{V}/\mathsf{x}] \Downarrow \mathsf{U}}{\mathsf{unpack}\ \mathsf{M}\ \mathsf{as}\ \mathsf{X},\mathsf{x}\ \mathsf{in}\ \mathsf{N} \Downarrow \mathsf{U}}$$

Fig. 2. Operational semantics of $\Lambda_{\exists\mu}$

Theorem VI.2. *For* $\Theta \mid - \vdash \mathsf{M} : \mathsf{A}$*, if* $\mathsf{M} \Uparrow$*, then there exists* $\phi \Vdash [\![\mathsf{M}]\!]$ *such that* $\phi(*) \perp e$ *for any* $e \in \mathcal{E}$*.*

Corollary VI.1. *Let* M *be a term of the form* $- \mid - \vdash \mathsf{M} : \mathsf{A}$*. If* $\mathcal{E}$ *admits error, then* $\mathsf{M} \Uparrow$ *if and only if we have* $\phi(*) \perp e$ *for any* $\phi \Vdash [\![\mathsf{M}]\!]$ *and* $e \in \mathcal{E}$*.*

VII. Examples

We give two examples of cbv combinatorial objects and derive several syntactic properties from our main results.

A. Small step operational semantics

In Section II, we gave an example of a cbv combinatorial object based on an untyped call-by-value lambda calculus. In this section, we consider a richer untyped call-by-value lambda calculus $\lambda_{\exists\mu}$ obtained by removing type annotations from $\Lambda_{\exists\mu}$. As a corollary of our main results, we prove that the small step operational semantics of $\lambda_{\exists\mu}$ "coincides" with the big step operational semantics of $\Lambda_{\exists\mu}$. Terms t, values v and evaluation contexts e of $\lambda_{\exists\mu}$ are as follows:

$$\begin{aligned}
\mathsf{t} ::= {} & \mathsf{x} \mid () \mid (\mathsf{t},\mathsf{t}) \mid \mathsf{fst}(\mathsf{t}) \mid \mathsf{snd}(\mathsf{t}) \mid \mathsf{inl}(\mathsf{t}) \mid \mathsf{inr}(\mathsf{t}) \\
& \mid \mathsf{case}\ \mathsf{t}\ \mathsf{of}\ (\mathsf{x} \mapsto \mathsf{t} \mid \mathsf{y} \mapsto \mathsf{t}) \mid \mathsf{tt} \mid \lambda\mathsf{x}.\mathsf{t} \mid \mathsf{rec}(\mathsf{f},\mathsf{x}).\mathsf{t} \\
& \mid \mathsf{fld}(\mathsf{t}) \mid \mathsf{ufd}(\mathsf{t}) \mid \Lambda\mathsf{t} \mid \mathsf{t}\bullet \mid \mathsf{pack}(\mathsf{t}) \mid \mathsf{unpack}\ \mathsf{t}\ \mathsf{as}\ \mathsf{x}\ \mathsf{in}\ \mathsf{t} \\
\mathsf{v} ::= {} & \mathsf{x} \mid () \mid (\mathsf{v},\mathsf{v}) \mid \mathsf{inl}(\mathsf{v}) \mid \mathsf{inr}(\mathsf{v}) \mid \lambda\mathsf{x}.\mathsf{t} \mid \mathsf{rec}(\mathsf{f},\mathsf{x}).\mathsf{t} \\
& \mid \mathsf{fld}(\mathsf{v}) \mid \Lambda\mathsf{t} \mid \mathsf{pack}(\mathsf{v}) \\
\mathsf{e} ::= {} & [-] \mid (\mathsf{e},\mathsf{t}) \mid (\mathsf{v},\mathsf{e}) \mid \mathsf{fst}(\mathsf{e}) \mid \mathsf{snd}(\mathsf{e}) \mid \mathsf{inl}(\mathsf{e}) \mid \mathsf{inr}(\mathsf{e}) \\
& \mid \mathsf{case}\ \mathsf{e}\ \mathsf{of}\ (\mathsf{x} \mapsto \mathsf{t} \mid \mathsf{y} \mapsto \mathsf{t}) \mid \mathsf{et} \mid \mathsf{ve} \mid \mathsf{fld}(\mathsf{e}) \mid \mathsf{ufd}(\mathsf{e}) \\
& \mid \mathsf{e}\bullet \mid \mathsf{pack}(\mathsf{e}) \mid \mathsf{unpack}\ \mathsf{e}\ \mathsf{as}\ \mathsf{x}\ \mathsf{in}\ \mathsf{t}
\end{aligned}$$

We give small step operational semantics of $\lambda_{\exists\mu}$:

$$\begin{aligned}
(\lambda) \quad & \mathsf{e}[(\lambda\mathsf{x}.\mathsf{t})\mathsf{v}] \mapsto \mathsf{e}[\mathsf{t}[\mathsf{v}/\mathsf{x}]] \\
(\mathsf{rec}) \quad & \mathsf{e}[(\mathsf{rec}(\mathsf{f},\mathsf{x}).\mathsf{t})\mathsf{v}] \mapsto \mathsf{e}[\mathsf{t}[\mathsf{rec}(\mathsf{f},\mathsf{x}).\mathsf{t}/\mathsf{f},\mathsf{v}/\mathsf{x}]] \\
(\mathsf{fst}) \quad & \mathsf{e}[\mathsf{fst}(\mathsf{v},\mathsf{v}')] \mapsto \mathsf{e}[\mathsf{v}] \\
(\mathsf{snd}) \quad & \mathsf{e}[\mathsf{snd}(\mathsf{v},\mathsf{v}')] \mapsto \mathsf{e}[\mathsf{v}'] \\
(\mathsf{inl}) \quad & \mathsf{e}[\mathsf{case}\ \mathsf{inl}(\mathsf{v})\ \mathsf{of}\ (\mathsf{x} \mapsto \mathsf{t} \mid \mathsf{y} \mapsto \mathsf{s})] \mapsto \mathsf{e}[\mathsf{t}[\mathsf{v}/\mathsf{x}]] \\
(\mathsf{inr}) \quad & \mathsf{e}[\mathsf{case}\ \mathsf{inr}(\mathsf{v})\ \mathsf{of}\ (\mathsf{x} \mapsto \mathsf{t} \mid \mathsf{y} \mapsto \mathsf{s})] \mapsto \mathsf{e}[\mathsf{s}[\mathsf{v}/\mathsf{y}]] \\
(\mathsf{pack}) \quad & \mathsf{e}[\mathsf{unpack}\ \mathsf{pack}(\mathsf{v})\ \mathsf{as}\ \mathsf{x}\ \mathsf{in}\ \mathsf{t}] \mapsto \mathsf{e}[\mathsf{t}[\mathsf{v}/\mathsf{x}]] \\
(\mathsf{fld}) \quad & \mathsf{e}[\mathsf{ufd}(\mathsf{fld}(\mathsf{v}))] \mapsto \mathsf{e}[\mathsf{v}] \\
(\Lambda) \quad & \mathsf{e}[(\Lambda\mathsf{t})\bullet] \mapsto \mathsf{e}[\mathsf{t}]
\end{aligned}$$

Definition VII.1. A $(\mathsf{rec},\mathsf{fld})$*-reduction* is a reduction that is a rec-reduction or a fld-reduction. A $\lambda_{\exists\mu}$-term t is *safe for* n*-*$(\mathsf{rec},\mathsf{fld})$ *steps* when t terminates to a value or there exists a reduction sequence from t that includes at least n-$(\mathsf{rec},\mathsf{fld})$ reduction steps. We say that t *is* $(\mathsf{rec},\mathsf{fld})$*-safe* when t is safe for n-$(\mathsf{rec},\mathsf{fld})$ steps for arbitrary large n.

For a $\Lambda_{\exists\mu}$-term M, we define a $\lambda_{\exists\mu}$-term $\mathsf{M}^\sharp$ by removing all type annotations from M. To be precise, we define $(\Lambda\mathsf{X}.\mathsf{M})^\sharp$ to be $\Lambda\mathsf{M}^\sharp$, and we define $(\mathsf{MA})^\sharp$ to be $\mathsf{M}^\sharp\bullet$.

Definition VII.2. A $\lambda_{\exists\mu}$-term t is *typable* when there exists a $\Lambda_{\exists\mu}$-term $\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A}$ such that $\mathsf{t} = \mathsf{M}^\sharp$.

The set $A_{\lambda_{\exists\mu}}$ of closed $\lambda_{\exists\mu}$-terms forms a cbv combinatorial object by the same data for λ^{v} described in Section II. Let e_ω be the constant function on $A_{\lambda_{\exists\mu}}$ that returns $(\mathsf{rec}(\mathsf{f},\mathsf{x}).\mathsf{fx})()$, whose evaluation loops with infinitely many rec-reductions. We define an evaluation $\mathcal{E}_{\lambda_{\exists\mu}}$ to be the set of homomorphisms that are equal to the constant function e_ω or of the form $e(-) = \mathsf{e}[-]$ for some evaluation context e. We define an indexed predicate $\Omega_{\lambda_{\exists\mu}}$ by

$$\Omega_{\lambda_{\exists\mu}}(n) := \{\mathsf{t} \in A_{\lambda_{\exists\mu}} \mid \mathsf{t} \text{ is safe for } n\text{-}(\mathsf{rec},\mathsf{fld}) \text{ steps}\}.$$

With respect to the evaluation $\mathcal{E}_{\lambda_{\exists\mu}}$ and the indexed predicate $\Omega_{\lambda_{\exists\mu}}$, a reduction is strict if and only if it consists of at least one $(\mathsf{rec},\mathsf{fld})$-reduction step. The cbv combinatorial object $A_{\lambda_{\exists\mu}}$ satisfies Requirement V.1 and Requirement V.2. Let $[\![-]\!]$ be the interpretation of $\Lambda_{\exists\mu}$ determined by $A_{\lambda_{\exists\mu}}$, $\mathcal{E}_{\lambda_{\exists\mu}}$, $\Omega_{\lambda_{\exists\mu}}$ and the following data:

$$\begin{aligned}
k(\mathsf{v}) &:= \mathsf{inl}(\mathsf{v}) & k'(\mathsf{v}) &:= \mathsf{inr}(\mathsf{v}) \\
\imath(\mathsf{v}) &:= \mathsf{pack}(\mathsf{v}) & \xi &:= \lambda\mathsf{x}.\mathsf{x}\bullet \\
\ell(\mathsf{v}) &:= \mathsf{fld}(\mathsf{v}) & Y_\phi(\mathsf{v}) &:= \mathsf{rec}(\mathsf{f},\mathsf{x}).\phi((\mathsf{v},\mathsf{f}),\mathsf{x}).
\end{aligned}$$

Proposition VII.1. *Let* $\Theta \mid \mathsf{x}_1 : \mathsf{A}_1, \cdots, \mathsf{x}_n : \mathsf{A}_n \vdash \mathsf{M} : \mathsf{A}$ *be a well formed* $\Lambda_{\exists\mu}$*-term. We have*

$$\lambda\mathsf{z}.\mathsf{M}^\sharp[\mathsf{p}_1\mathsf{z}/\mathsf{x}_1, \cdots \mathsf{p}_n\mathsf{z}/\mathsf{x}_n] \Vdash [\![\mathsf{M}]\!] : [\![\mathsf{A}_1]\!] \times \cdots \times [\![\mathsf{A}_n]\!] \to [\![\mathsf{A}]\!]$$

where p_i *is the* i*-th projection.*

Proposition VII.2. *Closed typable terms are* $(\mathsf{rec},\mathsf{fld})$*-safe.*

Proof: Let $\mathsf{t} = \mathsf{M}^\sharp$ be a closed typable term, and let A be the closed type of M. For every $n \geq 1$, $(e_\omega, [-])$ is in $[\![\mathsf{A}]\!]^\perp(n)$.

Therefore, by Proposition VII.1, $(\lambda \mathsf{z}.\mathsf{M}^\sharp)()$ is $(\mathsf{rec}, \mathsf{fld})$-safe. Since $(\lambda \mathsf{z}.\mathsf{M}^\sharp)() \rightsquigarrow \mathsf{M}^\sharp$, we see that $\mathsf{M}^\sharp$ is $(\mathsf{rec}, \mathsf{fld})$-safe. ■

As a corollary of our main results, we can show that the big step operational semantics of $\Lambda_{\exists\mu}$ "coincides" with the small step semantics of $\lambda_{\exists\mu}$.

Corollary VII.1. *For any* $\Theta \mid - \vdash \mathsf{M} : \mathsf{A}$*, if* $\mathsf{M} \Downarrow \mathsf{V}$*, then* $\mathsf{M}^\sharp \rightsquigarrow \mathsf{V}^\sharp$*, and if* $\mathsf{M} \Uparrow$ *then there exists an infinite* $(\mathsf{rec}, \mathsf{fld})$*-reduction sequence from* $\mathsf{M}^\sharp$*.*

Proof: By induction on the size of big step derivations, we can show that $\mathsf{M} \Downarrow \mathsf{V}$ implies $\mathsf{M}^\sharp \rightsquigarrow \mathsf{V}^\sharp$. Since $\mathcal{E}_{\lambda_{\exists\mu}}$ admits error, if $\mathsf{M} \Uparrow$, then $\mathsf{M}^\sharp$ does not reduce to a value by Corollary VI.1. Since $\mathsf{M}^\sharp$ is $(\mathsf{rec}, \mathsf{fld})$-safe, there exists an infinite $(\mathsf{rec}, \mathsf{fld})$-reduction sequence from $\mathsf{M}^\sharp$. ■

B. Categorical abstract machine

We describe a categorical abstract machine (CAM). A CAM consists of instructions, codes, terms and stacks:

$$
\begin{array}{ll}
\text{Instruction} & i ::= \mathtt{Car} \mid \mathtt{Cdr} \mid \mathtt{Qt} \mid \mathtt{Cur}(c) \mid \mathtt{RC}(c) \\
& \quad \mid \mathtt{Push} \mid \mathtt{Swap} \mid \mathtt{Cons} \mid \mathtt{App} \mid \mathtt{Inl} \mid \mathtt{Inr} \\
& \quad \mid \mathtt{Sel}(c, c) \mid \mathtt{Skip} \\
\text{Code} & c ::= [] \mid i \cdot c \\
\text{Term} & t ::= \bullet \mid c : t \mid c \overset{\mathtt{R}}{:} t \mid (t, t) \mid (0, t) \mid (1, t) \\
\text{Stack} & s ::= [] \mid t \cdot s
\end{array}
$$

The definition of CAM is based on the one in [9], which is enough to provide adequate compilation of $\Lambda_{\exists\mu}$; we add several instructions so as to provide short compiled codes. The instruction `Skip` provides us a one step reduction without effects. We use `Skip` in the interpretation of recursive types. A configuration of CAM is a triple (t, c, s) of a term t, a code c and a stack s. We give transition rules between CAM configurations in Figure 3.

Definition VII.3. We say that (t, c, s) is *safe for n-steps* when (t, c, s) reduces to $(t', [], [])$ for some term t' or there exists an n-step reduction sequence starting from (t, c, s).

We write T, C and S for the set of terms, codes and stacks respectively. The set C forms a monoid by the concatenation of instruction sequences. Each code c has an action on the set $T \times C \times S$ given by $c \ltimes (t, c', s) := (t, c' \cdot c, s)$. We define a total basic combinatorial object: we define A_{CAM} to be $T \times C \times S$ and $\rightsquigarrow_{\mathrm{CAM}}$ to be the reflexive transitive closure of $\mapsto$. We define a set of values V_{CAM} to be $T \times \{[]\} \times \{[]\}$ and a set of homomorphisms Φ_{CAM} to be $\{c \ltimes (-) \mid c \in C\}$.

In the following, we identify a code c with a homomorphism $c \ltimes (-)$, and we identify a term t with $(t, [], []) \in V_{\mathrm{CAM}}$. The identity ι is the empty code $[]$, and we define the composition $\phi \cdot \psi$ to be the concatenation of codes.

A_{CAM} forms a cbv combinatorial object by:

$$
\begin{gathered}
* := \bullet \quad \upsilon := \mathtt{Qt} \quad \pi := \mathtt{Car} \quad \pi' := \mathtt{Cdr} \\
\langle t, t' \rangle := (t, t') \quad \langle c, c' \rangle := \mathtt{Push} \cdot c \cdot \mathtt{Swap} \cdot c' \cdot \mathtt{Cons} \\
\gamma_{t,c}(t', c', s') := (t', c' \cdot \mathtt{Swap} \cdot c \cdot \mathtt{Cons}, s' \cdot t) \\
\delta_t(t', c, s) := (t', c \cdot \mathtt{Cons}, s \cdot t) \quad c \star t := c : t \quad \varepsilon := \mathtt{App}
\end{gathered}
$$

$$
\begin{array}{rcl}
((t, t'), \mathtt{Car} \cdot c, s) & \mapsto & (t, c, s) \\
((t, t'), \mathtt{Cdr} \cdot c, s) & \mapsto & (t', c, s) \\
(t, \mathtt{Qt} \cdot c, s) & \mapsto & (\bullet, c, s) \\
(t, \mathtt{Cur}(c) \cdot c', s) & \mapsto & (c : t, c', s) \\
(t, \mathtt{RC}(c) \cdot c', s) & \mapsto & (c \overset{\mathtt{R}}{:} t, c', s) \\
(t, \mathtt{Push} \cdot c, s) & \mapsto & (t, c, t \cdot s) \\
(t, \mathtt{Swap} \cdot c, t' \cdot s) & \mapsto & (t', c, t \cdot s) \\
(t, \mathtt{Cons} \cdot c, t' \cdot s) & \mapsto & ((t', t), c, s) \\
((c : t, t'), \mathtt{App} \cdot c', s) & \mapsto & ((t, t'), c \cdot c', s) \\
((c \overset{\mathtt{R}}{:} t, t'), \mathtt{App} \cdot c', s) & \mapsto & (((t, c \overset{\mathtt{R}}{:} t), t'), c \cdot c', s) \\
(t, \mathtt{Inl} \cdot c, s) & \mapsto & ((0, t), c, s) \\
(t, \mathtt{Inr} \cdot c, s) & \mapsto & ((1, t), c, s) \\
((0, t), \mathtt{Sel}(c, c') \cdot c'', s) & \mapsto & (t, c \cdot c'', s) \\
((1, t), \mathtt{Sel}(c, c') \cdot c'', s) & \mapsto & (t, c' \cdot c'', s) \\
(t, \mathtt{Skip} \cdot c, s) & \mapsto & (t, c, s)
\end{array}
$$

Fig. 3. CAM work-flow

We define e_ω to be the constant map that always returns $((\mathtt{Lp} \overset{\mathtt{R}}{:} \bullet, \bullet), \mathtt{App}, [])$ where

$$\mathtt{Lp} := \mathtt{Push} \cdot \mathtt{Car} \cdot \mathtt{Cdr} \cdot \mathtt{Swap} \cdot \mathtt{Cdr} \cdot \mathtt{Cons} \cdot \mathtt{App}.$$

We note that the evaluation of $((\mathtt{Lp} \overset{\mathtt{R}}{:} \bullet, \bullet), \mathtt{App}, [])$ loops. We identify $(c, s) \in C \times S$ with a map $(t, c', s') \mapsto (t, c' \cdot c, s' \cdot s)$, and we define an evaluation $\mathcal{E}_{\mathrm{CAM}}$ to be $\mathcal{E}_{\mathrm{CAM}} := (C \times S) \cup \{e_\omega\}$ and an indexed predicate Ω_{CAM} by

$$\Omega_{\mathrm{CAM}}(n) := \{(t, c, s) \mid (t, c, s) \text{ is safe for } n\text{-steps}\}.$$

Then the strict reduction $\dot{\rightsquigarrow}_{\mathrm{CAM}}$ coincides with the transitive closure of $\mapsto$. We consider the interpretation of $\Lambda_{\exists\mu}$ determined by the following data:

$$k(t) := (0, t) \quad k'(t) := (1, t) \quad \imath(t) := t$$

$$\xi := \mathtt{Push} \cdot \mathtt{Qt} \cdot \mathtt{Cons} \cdot \mathtt{App} \quad Y_c(t) := c \overset{\mathtt{R}}{:} t \quad \ell(t) := t$$

We give a compilation $[\![-]\!]_{\mathrm{CAM}}$ of $\Lambda_{\exists\mu}$ into the CAM in Figure 4.

Proposition VII.3. *Let* $\Theta \mid \mathsf{x}_1 : \mathsf{A}_1, \cdots \mathsf{x}_n : \mathsf{A}_n \vdash \mathsf{M} : \mathsf{A}$ *be a well formed* $\Lambda_{\exists\mu}$*-term. We have*

$$[\![\mathsf{M}]\!]_{\mathrm{CAM}} \Vdash [\![\mathsf{M}]\!] \circ \pi : \top \times [\![\mathsf{A}_1]\!] \times \cdots \times [\![\mathsf{A}_n]\!] \to [\![\mathsf{A}]\!]$$

where $\pi : \top \times [\![\mathsf{A}_1]\!] \times \cdots \times [\![\mathsf{A}_n]\!] \to [\![\mathsf{A}_1]\!] \times \cdots \times [\![\mathsf{A}_n]\!]$ *is the projection.*

Corollary VII.2. *For* $\Theta \mid - \vdash \mathsf{M} : \mathsf{Unit} + \mathsf{Unit}$*, we have*

1) *if* $\mathsf{M} \Downarrow \mathsf{inl}(())$*, then* $(\bullet, [\![\mathsf{M}]\!]_{\mathrm{CAM}}, []) \rightsquigarrow ((0, \bullet), [], [])$*,*
2) *if* $\mathsf{M} \Downarrow \mathsf{inr}(())$*, then* $(\bullet, [\![\mathsf{M}]\!]_{\mathrm{CAM}}, []) \rightsquigarrow ((1, \bullet), [], [])$*,*
3) *if* $\mathsf{M} \Uparrow$*, then there is an infinite transition sequence from* $(\bullet, [\![\mathsf{M}]\!]_{\mathrm{CAM}}, [])$*.*

Remark VII.1. Since the instruction `Skip` does not affect termination and divergence of configurations, we can remove `Skip` from CAM compilations in Figure 4.

$$[\![\Theta \mid x_1 : A_1, \cdots, x_n : A_n \vdash x_i : A_i]\!]_{\mathrm{CAM}} := \mathtt{Car}^{n-i} \cdot \mathtt{Cdr}$$
$$[\![()]\!]_{\mathrm{CAM}} := \mathtt{Qt}$$
$$[\![(M, N)]\!]_{\mathrm{CAM}} := \mathtt{Push} \cdot [\![M]\!]_{\mathrm{CAM}} \cdot \mathtt{Swap} \cdot [\![N]\!]_{\mathrm{CAM}} \cdot \mathtt{Cons}$$
$$[\![\mathsf{inl}_{A,B}(M)]\!]_{\mathrm{CAM}} := [\![M]\!]_{\mathrm{CAM}} \cdot \mathtt{Inl}$$
$$[\![\mathsf{inr}_{A,B}(M)]\!]_{\mathrm{CAM}} := [\![M]\!]_{\mathrm{CAM}} \cdot \mathtt{Inr}$$
$$[\![\mathsf{case}\ M\ \mathsf{of}\ (x \mapsto N \mid y \mapsto N')]\!]_{\mathrm{CAM}} := \mathtt{Push} \cdot [\![M]\!]_{\mathrm{CAM}} \cdot \mathtt{Sel}(\mathtt{Cons} \cdot [\![N]\!]_{\mathrm{CAM}}, \mathtt{Cons} \cdot [\![N']\!]_{\mathrm{CAM}})$$
$$[\![MN]\!]_{\mathrm{CAM}} := \mathtt{Push} \cdot [\![M]\!]_{\mathrm{CAM}} \cdot \mathtt{Swap} \cdot [\![N]\!]_{\mathrm{CAM}} \cdot \mathtt{Cons} \cdot \mathtt{App}$$
$$[\![\lambda x.M]\!]_{\mathrm{CAM}} := \mathtt{Cur}([\![M]\!]_{\mathrm{CAM}})$$
$$[\![\mathsf{rec}_{A,B}(f, x).M]\!]_{\mathrm{CAM}} := \mathtt{RC}([\![M]\!]_{\mathrm{CAM}})$$
$$[\![\mathsf{fld}_{\mu X.A}(M)]\!]_{\mathrm{CAM}} := [\![M]\!]_{\mathrm{CAM}}$$
$$[\![\mathsf{ufd}(M)]\!]_{\mathrm{CAM}} := [\![M]\!]_{\mathrm{CAM}} \cdot \mathtt{Skip}$$
$$[\![\Lambda X.M]\!]_{\mathrm{CAM}} := \mathtt{Cur}(\mathtt{Car} \cdot [\![M]\!]_{\mathrm{CAM}})$$
$$[\![MA]\!]_{\mathrm{CAM}} := [\![M]\!]_{\mathrm{CAM}} \cdot \mathtt{Push} \cdot \mathtt{Qt} \cdot \mathtt{Cons} \cdot \mathtt{App}$$
$$[\![\mathsf{pack}_{B,\exists X.A}(M)]\!]_{\mathrm{CAM}} := [\![M]\!]_{\mathrm{CAM}}$$
$$[\![\mathsf{unpack}\ M\ \mathsf{as}\ X, x\ \mathsf{in}\ N]\!]_{\mathrm{CAM}} := \mathtt{Push} \cdot [\![M]\!]_{\mathrm{CAM}} \cdot \mathtt{Cons} \cdot [\![N]\!]_{\mathrm{CAM}}$$

Fig. 4. CAM compilation of $\Lambda_{\exists\mu}$

VIII. Conclusion

We introduced cbv combinatorial objects, from which we constructed a categorical realizability model of the call-by-value polymorphic lambda calculus $\Lambda_{\exists\mu}$. We proved soundness and adequacy of the realizability model. We gave two examples of cbv combinatorial objects based on an untyped call-by-value lambda calculus and a categorical abstract machine.

So as to make our construction more clear, it is promising to reformulate our work by means of the later modality like [12], [5]. Our $(-)^{\uparrow}$ operation is very similar to the later modality. Furthermore, $(-)^{\uparrow}$ satisfies that $X^{\uparrow} \cong X$ in $\mathbb{P}$. At this point, we do not know categorical formulations of the operator. Another direction is a study of a construction of categorical models over a "step-indexed cbv combinatorial object", not a step-indexed construction of categorical models over a cbv combinatorial object. For example, it would be possible to formulate step-indexed cbv combinatorial object in the topos of trees [8].

Acknowledgment

The author would like to thank Shin-ya Katsumata and Masahito Hasegawa for valuable advice. The author also thank reviewers for their helpful comments.

References

[1] S. Abramsky, E. Haghverdi, and P. Scott, "Geometry of interaction and linear combinatory algebras," *Mathematical Structures in Computer Science*, vol. 12, pp. 625–665, 2002.

[2] A. J. Ahmed, "Step-indexed syntactic logical relations for recursive and quantified types," in *ESOP*, ser. Lecture Notes in Computer Science, P. Sestoft, Ed., vol. 3924. Springer, 2006, pp. 69–83.

[3] R. Amadio, "On the adequacy of per models," in *Mathematical Foundations of Computer Science*, ser. LNCS, A. Borzyszkowski and S. Sokolowski, Eds. Springer Berlin / Heidelberg, 1993, vol. 711, pp. 222–231.

[4] A. W. Appel and D. McAllester, "An indexed model of recursive types for foundational proof-carrying code," *ACM Transactions on Programming Languages and Systems*, vol. 23, pp. 657–683, 2001.

[5] A. W. Appel, P.-A. Melliès, C. D. Richards, and J. Vouillon, "A very modal model of a modern, major, general type system," in *Proceedings of the 34th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. New York, NY, USA: ACM, 2007, pp. 109–122.

[6] N. Benton and C.-K. Hur, "Biorthogonality, step-indexing and compiler correctness," in *Proceedings of the 14th ACM SIGPLAN International Conference on Functional Programming*. New York, NY, USA: ACM, 2009, pp. 97–108.

[7] ——, "Realizability and compositional compiler correctness for a polymorphic language," Tech. Rep. MSR-TR-2010-62, 2010.

[8] L. Birkedal, R. E. Møgelberg, J. Schwinghammer, and K. Støvring, "First steps in synthetic guarded domain theory: Step-indexing in the topos of trees," in *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 2011, pp. 55–64.

[9] P.-L. Curien, *Categorical Combinators, Sequential Algorithms, and Functional Programming*, R. V. Book, Ed. Birkhäuser Boston, 1993.

[10] L. Dami, "Labelled reductions, runtime errors and operational subsumption," in *Proceedings of the 24th International Colloquium on Automata, Languages and Programming*. London, UK: Springer-Verlag, 1997, pp. 782–793.

[11] ——, "Operational subsumption, an ideal model of subtyping," *Electronic Notes in Theoretical Computer Science*, vol. 10, no. 0, pp. 28–49, 1998, second Workshop on Higher-Order Operational Techniques in Semantics.

[12] D. Dreyer, A. Ahmed, and L. Birkedal, "Logical step-indexed logical relations," in *Proceedings of the 24th Annual IEEE Symposium on Logic in Computer Science*, 2009, pp. 71–80.

[13] D. Dreyer, G. Neis, and L. Birkedal, "The impact of higher-order state and control effects on local relational reasoning," in *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*. New York, NY, USA: ACM, 2010, pp. 143–156.

[14] C. Führmann, "Direct models for the computational lambda calculus," *Electronic Notes in Theoretical Computer Science*, vol. 20, pp. 245–292, 1999.

[15] J.-Y. Girard, "Linear logic," *Theoretical Computer Science*, vol. 50, no. 1, pp. 1–101, 1987.

[16] P. J. Hofstra, "All realizability is relative," *Mathematical Proceedings of the Cambridge Philosophical Society*, pp. 239–264, 2006.

[17] N. Hoshino, "Step indexed realizability semantics for a call-by-value language based on basic combinatorial objects," unpublished. Available at http://www.kurims.kyoto-u.ac.jp/~naophiko/paper/sind.pdf.

[18] C.-K. Hur and D. Dreyer, "A Kripke logical relation between ML and assembly," in *Proceedings of the 38th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. New York, NY, USA: ACM, 2011, pp. 133–146.

[19] B. Jacobs, *Categorical Logic and Type Theory*, ser. Studies in Logic and the Foundations of Mathematics. Amsterdam: North Holland, 1999, no. 141.

[20] P. B. Levy, J. Power, and H. Thielecke, "Modelling environments in call-by-value programming languages," *Information and Computation*, vol. 185, no. 2, pp. 182–210, 2003.

[21] J. R. Longley and A. K. Simpson, "A uniform approach to domain theory in realizability models," *Mathematical Structures in Computer Science*, vol. 7, no. 5, pp. 469–505, 1997.

[22] P.-A. Melliès and J. Vouillon, "Recursive polymorphic types and parametricity in an operational framework," in *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 2005, pp. 82–91.

[23] H. Nakano, "A modality for recursion," in *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 2000, pp. 255–266.

[24] J. V. Oosten, *Realizability: an introduction to its categorical side*, ser. Studies in Logic and the Foundations of Mathematics. Elsevier, 2008, no. 152.

APPENDIX

For each $\phi \in \Phi$, we choose $\overline{\phi} \in \Phi$ that computes $\phi \star (-)$. We assume that $\cdot$ is left associative.

Proposition A.1. *If β-reduction in Definition II.1 is strict, then we have Y_ϕ that satisfies Requirement V.1.*

Proof: We define Y_ϕ to be

$$Y_\phi(x) := \omega_\phi \star \left\langle \left(\overline{\overline{\omega_\phi}} \cdot \pi'\right) \star *, x \right\rangle$$

where $\omega_\phi := \varepsilon \cdot \left\langle \varepsilon \cdot \left\langle \overline{\overline{\phi}} \cdot \pi', \varepsilon \cdot \langle \varepsilon \cdot \langle \iota, \iota \rangle \cdot \pi, \pi' \rangle \right\rangle \cdot \pi, \pi' \right\rangle$. Y_ϕ is computed by

$$\mu_\phi := \varepsilon \cdot \left\langle \overline{\overline{\omega_\phi}} \cdot \overline{\overline{\omega_\phi} \cdot \pi'} \cdot \upsilon, \iota \right\rangle.$$

■

Proposition A.2. *If the first projection in Definition II.1 is strict, then we have ℓ that satisfies Requirement V.1*

Proof: $\zeta := \langle \iota, \upsilon \rangle$ computes $\ell(x) := \langle x, * \rangle$, and π satisfies $\pi(\ell(x)) \rightsquigarrow x$. ■

A. Interpretation of $\Lambda_{\exists\mu}$

We interpret types as follows.

$$\begin{aligned}
[\![\mathsf{X}_1, \cdots, \mathsf{X}_n \vdash \mathsf{X}_i]\!](X_1, \cdots, X_n) &:= X_i \\
[\![\Theta \vdash \mathsf{Unit}]\!](\vec{X}) &:= \top \\
[\![\Theta \vdash \mathsf{A} \times \mathsf{B}]\!](\vec{X}) &:= [\![\Theta \vdash \mathsf{A}]\!](\vec{X}) \times [\![\Theta \vdash \mathsf{B}]\!](\vec{X}) \\
[\![\Theta \vdash \mathsf{A} + \mathsf{B}]\!](\vec{X}) &:= [\![\Theta \vdash \mathsf{A}]\!](\vec{X}) + [\![\Theta \vdash \mathsf{B}]\!](\vec{X}) \\
[\![\Theta \vdash \mathsf{A} \Rightarrow \mathsf{B}]\!](\vec{X}) &:= [\![\Theta \vdash \mathsf{A}]\!](\vec{X}) \Rightarrow [\![\Theta \vdash \mathsf{B}]\!](\vec{X}) \\
[\![\Theta \vdash \mu\mathsf{X}.\mathsf{A}]\!] &:= \mathrm{fix}[\![\Theta, \mathsf{X} \vdash \mathsf{A}]\!] \\
[\![\Theta \vdash \forall\mathsf{X}.\mathsf{A}]\!] &:= \forall_n [\![\Theta, \mathsf{X} \vdash \mathsf{A}]\!] \\
[\![\Theta \vdash \exists\mathsf{X}.\mathsf{A}]\!] &:= \exists_n [\![\Theta, \mathsf{X} \vdash \mathsf{A}]\!]
\end{aligned}$$

where n is the length of Θ. We give interpretation of terms in Figure 5.

B. Proof of adequacy

We write Type for the set of closed types and TVar for the set of type variables. We write $\mathrm{Value}(\mathsf{A})$ for the set of closed values of type A. A *type environment* is a map from TVar to Type.

For a type A and a type environment $\varrho : \mathrm{TVar} \to \mathrm{Type}$, we define A_ϱ to be the closed type obtained by replacing each free type variable X in A by $\varrho(\mathsf{X})$.

Let Γ be a term context of the form $\mathsf{x}_1 : \mathsf{A}_1, \cdots, \mathsf{x}_n : \mathsf{A}_n$. For a term $\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A}$, a type environment $\varrho : \mathrm{TVar} \to \mathrm{Type}$ and a finite sequence $\vec{\mathsf{V}} = \mathsf{V}_1, \cdots, \mathsf{V}_n$, we define $\mathsf{M}[\varrho/\Theta, \vec{\mathsf{V}}/\Gamma]$ to be the term obtained by replacing each free type variable X in M by $\varrho(\mathsf{X})$ and each free variable x_i in M by V_i.

For a closed type A, a *predicate* R *on* A is a family of subsets $R(n) \subset \mathrm{Value}(\mathsf{A}) \times V$ such that

$$R(1) \supset R(2) \supset \cdots.$$

We write $\mathrm{Pred}(\mathsf{A})$ for the set of predicates on A. When we refer to $R(0)$, we assume that $R(n)$ is defined to be $\mathrm{Value}(\mathsf{A}) \times V$.

For $R \in \mathrm{Pred}(\mathsf{A})$, we define

$$R^\perp(n) \subset \bigcup_{\mathsf{B} \in \mathrm{Type}} \mathrm{Value}(\mathsf{A} \Rightarrow \mathsf{B}) \times \mathcal{E}$$

to be

$$\bigcap_{k \le n} \{(\mathsf{V}, e) \mid \forall(\mathsf{U}, x) \in R(k).\ \mathsf{VU} \Uparrow \implies x \perp_k e\}.$$

We give some constructions of predicates. For each closed type A, we define the empty predicate $0 \in \mathrm{Pred}(\mathsf{A})$ by

$$0(n) := \emptyset.$$

For $R \in \mathrm{Pred}(\mathsf{A})$ and $S \in \mathrm{Pred}(\mathsf{B})$, we define $R \square S \in \mathrm{Pred}(\mathsf{A} \square \mathsf{B})$ for $\square = \times, \Rightarrow, +$ by

$$\begin{aligned}
(R \times S)(n) &:= \left\{ ((\mathsf{U}, \mathsf{V}), \langle x, y \rangle) \;\middle|\; \begin{array}{l} (\mathsf{U}, x) \in R(n) \\ (\mathsf{V}, y) \in S(n) \end{array} \right\}, \\
(R \Rightarrow S)(n) &:= \\
&\bigcap_{k \le n} \left\{ (\mathsf{V}, z) \;\middle|\; \begin{array}{r} \forall(\mathsf{U}, x) \in R(k).\ \forall(\mathsf{W}, e) \in S^\perp(k). \\ \mathsf{W}(\mathsf{VU}) \Uparrow \implies \varepsilon \langle z, x \rangle \perp_k e \end{array} \right\}, \\
(R + S)(n) &:= \{(\mathsf{inl}_{\mathsf{A},\mathsf{B}}(\mathsf{V}), k(x)) \mid (\mathsf{V}, x) \in R(n)\} \\
&\quad \cup \{(\mathsf{inr}_{\mathsf{A},\mathsf{B}}(\mathsf{V}), k'(x)) \mid (\mathsf{V}, x) \in S(n)\}.
\end{aligned}$$

A *predicate environment for a type environment* ϱ is a family of predicates of the form

$$\{\varsigma(\mathsf{X}) \in \mathrm{Pred}(\varrho(\mathsf{X}))\}_{\mathsf{X} \in \mathrm{TVar}}.$$

Let ϱ be a type environment and ς be a predicate environment for ϱ. For a closed type A and a predicate $R \in \mathrm{Pred}(\mathsf{A})$, we define a type environment $\varrho[\mathsf{X} \mapsto \mathsf{A}]$ and a predicate environment $\varsigma[\mathsf{X} \mapsto R]$ by

$$\begin{aligned}
\varrho[\mathsf{X} \mapsto \mathsf{A}](\mathsf{Y}) &:= \begin{cases} \varrho(\mathsf{Y}) & (\mathsf{X} \neq \mathsf{Y}) \\ \mathsf{A} & (\mathsf{X} = \mathsf{Y}) \end{cases} \\
\varsigma[\mathsf{X} \mapsto R](\mathsf{Y}) &:= \begin{cases} \varsigma(\mathsf{Y}) & (\mathsf{X} \neq \mathsf{Y}) \\ R & (\mathsf{X} = \mathsf{Y}) \end{cases}.
\end{aligned}$$

We give an inductive definition of a predicate $[\![\mathsf{A}]\!](\varrho, \varsigma) \in \mathrm{Pred}(\mathsf{A}_\varrho)$ in Figure 6. For a term environment $\Gamma = \mathsf{x}_1 : \mathsf{A}_1, \cdots, \mathsf{x}_n : \mathsf{A}_n$, we define $[\![\Gamma]\!](\varrho, \varsigma)$ to be the finite product $\prod_{i=1,2,\cdots,n} [\![\mathsf{A}_i]\!](\varrho, \varsigma)$.

Lemma A.1. *For a term $\Theta \mid \Gamma \vdash \mathsf{M} : \mathsf{A}$, a type environment $\varrho : \mathrm{TVar} \to \mathrm{Type}$ and a predicate environment ς for ϱ, there exists $\phi \Vdash [\![\mathsf{M}]\!]$ such that for any $n \ge 1$,*

$$\forall(\vec{\mathsf{V}}, \vec{x}) \in [\![\Gamma]\!](\varrho, \varsigma)(n).\ \forall(\mathsf{U}, e) \in ([\![\mathsf{A}]\!](\varrho, \varsigma))^\perp(n).\ \mathsf{UM}[\varrho/\Theta, \vec{\mathsf{V}}/\Gamma] \Uparrow \implies \phi \langle \vec{x} \rangle \perp_n e$$

where

$$\langle \vec{x} \rangle := \begin{cases} \langle \cdots \langle x_1, x_2 \rangle \cdots, x_n \rangle & (\vec{x} = x_1, x_2, \cdots, x_n) \\ * & (\vec{x} \text{ is empty}) \end{cases}.$$

$$[\![\Theta \mid x_1 : A_1, \cdots x_n : A_n \vdash x_i : A_i]\!] := i\text{-th projection}$$

$$[\![\Theta \mid \Gamma \vdash () : \mathsf{Unit}]\!] := \text{the unique strict morphism to } \top$$

$$[\![\Theta \mid \Gamma \vdash (M, N) : A \times B]\!] := [\![\Gamma]\!] \xrightarrow{\Delta} [\![\Gamma]\!] \otimes [\![\Gamma]\!] \xrightarrow{[\![M]\!] \times \mathrm{id}} [\![A]\!] \otimes [\![\Gamma]\!] \xrightarrow{\mathrm{id} \times [\![N]\!]} [\![A]\!] \otimes [\![B]\!]$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{fst}(M) : A]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![A]\!] \otimes [\![B]\!] \xrightarrow{[\pi]} [\![A]\!]$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{snd}(M) : A]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![A]\!] \otimes [\![B]\!] \xrightarrow{[\pi']} [\![A]\!]$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{inl}_{A,B}(M) : A + B]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![A]\!] \xrightarrow{[\kappa]} [\![A]\!] + [\![B]\!]$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{inr}_{A,B}(M) : A + B]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![B]\!] \xrightarrow{[\kappa']} [\![A]\!] + [\![B]\!]$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{case}\ M\ \mathsf{of}\ (x \mapsto N \mid y \mapsto N')]\!] := [\![\Gamma]\!] \xrightarrow{\Delta} [\![\Gamma]\!] \otimes [\![\Gamma]\!] \xrightarrow{\mathrm{id} \otimes [\![M]\!]} [\![\Gamma]\!] \otimes ([\![A]\!] + [\![B]\!])$$
$$\xrightarrow{\text{distributivity}} [\![\Gamma]\!] \otimes [\![A]\!] + [\![\Gamma]\!] \otimes [\![B]\!] \xrightarrow{[[\![N]\!], [\![N']\!]]} [\![C]\!]$$

$$[\![\Theta \mid \Gamma \vdash MN : A]\!] := [\![\Gamma]\!] \xrightarrow{\Delta} [\![\Gamma]\!] \otimes [\![\Gamma]\!] \xrightarrow{[\![M]\!] \times \mathrm{id}} [\![B \Rightarrow A]\!] \otimes [\![\Gamma]\!] \xrightarrow{\mathrm{id} \times [\![N]\!]} [\![B \Rightarrow A]\!] \otimes [\![B]\!] \xrightarrow{[\varepsilon]} [\![A]\!]$$

$$[\![\Theta \mid \Gamma \vdash \lambda x : A.M : A \Rightarrow B]\!] := \text{the transpose of } [\![M]\!] \qquad \text{(Proposition V.4)}$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{rec}_{A,B}(f, x).M : A \Rightarrow B]\!] := [\![M]\!]^{\dagger} \qquad \text{(Proposition V.6)}$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{fld}_{\mu X.A}(M) : \mu X.A]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![A[\mu X.A/X]]\!] \xrightarrow{\theta^{-1}_{[\![A]\!]}} [\![\mu X.A]\!]$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{ufd}(M) : A[\mu X.A/X]]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![\mu X.A]\!] \xrightarrow{\theta_{[\![A]\!]}} [\![A[\mu X.A/X]]\!]$$

$$[\![\Theta \mid \Gamma \vdash \Lambda X.M : \forall X.A]\!] := \text{the transpose of } [\![M]\!] \qquad \text{(Section VI-B2)}$$

$$[\![\Theta \mid \Gamma \vdash MB : A[B/X]]\!] := \text{the reindexing of the transpose of } [\![M]\!] : [\![\Gamma]\!] \to \forall_n [\![A]\!]$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{pack}_{B,\exists X.A}(M) : \exists X.A]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![A[B/X]]\!] \xrightarrow{(\#)} \exists_n [\![A]\!]$$
$$\text{where } (\#) \text{ is the reindexing of the unit of } \exists_n \vdash \pi_n^*$$

$$[\![\Theta \mid \Gamma \vdash \mathsf{unpack}\ M\ \mathsf{as}\ X, x\ \mathsf{in}\ N : B]\!] := [\![\Gamma]\!] \xrightarrow{\Delta} [\![\Gamma]\!] \otimes [\![\Gamma]\!] \xrightarrow{\mathrm{id} \otimes [\![M]\!]} [\![\Gamma]\!] \otimes \exists_n [\![A]\!] \xrightarrow{(@)} [\![B]\!]$$
$$\text{where } (@) \text{ is obtained from } [\![N]\!] : \pi_n^* [\![\Gamma]\!] \otimes [\![A]\!] \to \pi_n^* [\![B]\!] \text{ via Frobenius property}$$

Fig. 5. Interpretation of terms

Proof: By induction on term construction. ∎

Theorem A.1 (Theorem VI.2)**.** *For* $\Theta \mid - \vdash M : A$*, if* $M \Uparrow$*, then there exists* $\phi \Vdash [\![M]\!]$ *such that* $\phi(*) \perp e$ *for any* $e \in \mathcal{E}$*.*

Proof: $(\lambda x.x, e)$ is in $R^{\perp}(n)$ for any predicate R and $n \geq 1$. Since $(\lambda x.x)M \Uparrow$, we have $\phi(*) \perp_n e$ for any n. ∎

Corollary A.1 (Corollary VI.1)**.** *Let* M *be a term of the form* $- \mid - \vdash M : A$*. If* $\mathcal{E}$ *admits error, then* $M \Uparrow$ *if and only if we have* $\phi(*) \perp e$ *for any* $\phi \Vdash [\![M]\!]$ *and* $e \in \mathcal{E}$*.*

Proof: Let e_0 be a witness of error. (Only if) By Theorem VI.2, there exists $\phi_0 \Vdash [\![M]\!]$ such that $\phi_0(*) \perp e_0$. Since (e_0, e) is in $[\![A]\!]$ for any $e \in \mathcal{E}$, if $\phi \sim \phi_0$, then $\phi(*) \perp e$ for any $e \in \mathcal{E}$, and generally, we have $\phi(*) \perp e$ for any $e \in \mathcal{E}$ and any $\phi \approx \phi_0 \Vdash M$. (If) We suppose that $M \Downarrow V$. Since $[\![M]\!] = [\![V]\!]$, there exists $\phi \Vdash M$ such that $\phi(*) \rightsquigarrow x \in V$, which contradicts to $\phi(*) \perp e_0$. ∎

$$
\begin{aligned}
[\![\mathsf{X}]\!](\varrho,\varsigma) &:= \varsigma(\mathsf{X}) \\
[\![\mathsf{Unit}]\!](\varrho,\varsigma)(n) &:= \{((),*)\} \\
[\![\mathsf{A}\square\mathsf{B}]\!](\varrho,\varsigma) &:= [\![\mathsf{A}]\!](\varrho,\varsigma)\square[\![\mathsf{B}]\!](\varrho,\varsigma) \qquad (\square = \times, \Rightarrow, +) \\
[\![\exists\mathsf{X}.\mathsf{A}]\!](\varrho,\varsigma)(n) &:= \bigcup_{\mathsf{B}\in\mathrm{Type}} \bigcup_{S\in\mathrm{Pred}(\mathsf{B})} \{(\mathsf{pack}_{\mathsf{B},(\exists\mathsf{X}.\mathsf{A})_\varrho}(\mathsf{V}), \imath(x)) \mid (\mathsf{V},x) \in [\![\mathsf{A}]\!](\varrho[\mathsf{X}\mapsto\mathsf{B}],\varsigma[\mathsf{X}\mapsto S])(n)\} \\
[\![\forall\mathsf{X}.\mathsf{A}]\!](\varrho,\varsigma)(n) &:= \bigcap_{k\le n} \bigcap_{\mathsf{B}\in\mathrm{Type}} \bigcap_{S\in\mathrm{Pred}(\mathsf{B})} \left\{ (\Lambda\mathsf{X}.\mathsf{M}, z) \;\middle|\; \begin{array}{l} \forall(\mathsf{V},e) \in [\![\mathsf{A}]\!](\varrho[\mathsf{X}\mapsto\mathsf{B}],\varsigma[\mathsf{X}\mapsto S])^{\perp}(k). \\ \mathsf{V}(\mathsf{M}[\mathsf{B}/\mathsf{X}]) \Uparrow \Longrightarrow\ \xi(z) \perp_k e \end{array} \right\} \\
[\![\mu^1\mathsf{X}.\mathsf{A}]\!](\varrho,\varsigma)(n) &:= \{(\mathsf{fld}_{(\mu\mathsf{X}.\mathsf{A})_\varrho}(\mathsf{V}), \ell(x)) \mid (\mathsf{V},x) \in [\![\mathsf{A}]\!](\varrho[\mathsf{X}\mapsto(\mu\mathsf{X}.\mathsf{A})_\varrho],\varsigma[\mathsf{X}\mapsto \mathrm{Value}((\mu\mathsf{X}.\mathsf{A})_\varrho)])(n-1)\} \\
[\![\mu^{i+1}\mathsf{X}.\mathsf{A}]\!](\varrho,\varsigma)(n) &:= \{(\mathsf{fld}_{(\mu\mathsf{X}.\mathsf{A})_\varrho}(\mathsf{V}), \ell(x)) \mid (\mathsf{V},x) \in [\![\mathsf{A}]\!](\varrho[\mathsf{X}\mapsto(\mu\mathsf{X}.\mathsf{A})_\varrho],\varsigma[\mathsf{X}\mapsto [\![\mu^i\mathsf{X}.\mathsf{A}]\!](\varrho,\varsigma)])(n-1)\} \\
[\![\mu\mathsf{X}.\mathsf{A}]\!](\varrho,\varsigma)(n) &:= [\![\mu^n\mathsf{X}.\mathsf{A}]\!](\varrho,\varsigma)(n)
\end{aligned}
$$

Fig. 6. Interpretation of types