Modal type theory based on the intuitionistic epistemic logic

Abstract

Modal intuitionistic epistemic logic IEL$^-$ was proposed by S.Artemov and T. Protopopescu as the formal foundation for the intuitionistic theory of knowledge. We construct a modal simply typed lambda-calculus which is Curry-Howard isomorphic to IEL$^-$ as formal theory of calculations with applicative functors in functional programming languages like Haskell or Idris.

1 Introduction

Modal intutionistic epistemic logic IEL was proposed by S. Artemov and T. Proropopescu [1]. IEL provides the epistimology and the theory of knowledge as based on BHK-semantics of intuitionistic logic. IEL$^-$ is a variant of IEL, that corresponds to intuitionistic belief. Informally, $\mathbf{K}A$ denotes that A is verified intuitionistically.

Intuitionistic epistemic logic IEL$^-$ is defined with by following axioms and derivation rules:

Definition 1. *Intuitionistic epistemic logic IEL:*
1) IPC axioms;
2) $\mathbf{K}(A \to B) \to (\mathbf{K}A \to \mathbf{K}B)$ *(normality);*
3) $A \to \mathbf{K}A$ *(co-reflection);*
Rule: MP.

We have the deduction theorem and necessitation rule which is derivable.

V. Krupski and A. Yatmanov provided the sequential calculus for IEL and proved that this calculus is PSPACE-complete [2].

Functional programming languages such as Haskell [3], Idris [4], Purescript [5] Elm [6] or Scala [?] have special type classes[1] for calculations with container types like `Functor` and `Applicative` [2]:

```
class Functor f where
    fmap :: (a -> b) -> f a -> f b
```

```
class Functor f => Applicative f where
    pure :: a -> f a
    (<*>) :: f (a -> b) -> f a -> f b
```

---

[1]Type class in Haskell is a general interface for special group of datatypes.

[2]Reader may read more about container types in the Haskell standard library documentation[7] or in the next one textbook [8]

By *container* (or *computational context*) type we mean some type-operator f, where f is a "function" from $*$ to $*$: type operator takes a simple type (which has kind $*$) and returns another simple type type with kind $*$. For more detailed description of the type system with kinds used in Haskell see [12].

The motivation for using an applicative functor is quite natural. Applicative functor allows to generalize the action of a functor for functions with arbitrary number of arguments, for instance:

```
liftA2 :: Applicative f => (a -> b -> c) -> f a -> f b -> f c
liftA2 f x y = pure f <*> x <*> y
```

It's not difficult to see that modal axioms in IEL^- and types of the methods of Applicative class in Haskell-like languages (which is described below) are syntactically similar and we are going to show that this coincidence has a non-trivial computational meaning.

The main goal of our research is a relationship between intuitionistic epistemic logic IEL^- and functional programming with effects. We show that relationship by building the type system (which is called $\lambda_{\mathbf{K}}$) which is Curry-Howard isomorphic to IEL^-. So we will consider $\mathbf{K}$-modality as an arbitrary applicative functor.

$\lambda_{\mathbf{K}}$ consists of the rules for simply typed lambda-calculus and special typing rules for lifting types into the applicative functor $\mathbf{K}$. We assume that our type system will axiomatize the simplest case of computation with effects with one container. We provide proof-theoretical view on this kind of computations in functional programming and prove strong normalization and confluence.

2 Typed lambda-calculus based on IEL$^-$

At first we define the natural deduction for IEL$^-$:

Definition 2. *Natural deduction NIEL for IEL$^-$ is an extenstion of intuitionistic natural deduction with additional derivation rules for modality:*

$$\frac{\Gamma \vdash A}{\Gamma \vdash \mathbf{K}A}\ \mathbf{K}_I \qquad\qquad \frac{\Gamma \vdash \mathbf{K}\vec{A} \qquad \vec{A} \vdash B}{\Gamma \vdash \mathbf{K}B}$$

Where $\Gamma \vdash \mathbf{K}\vec{A}$ is a syntax sugar for $\Gamma \vdash \mathbf{K}A_1, \ldots, \Gamma \vdash \mathbf{K}A_n$.

Lemma 1. $\Gamma \vdash_{NIEL^-_{\wedge,\to}} A \Rightarrow IEL^- \vdash \bigwedge \Gamma \to A$.

Proof. Induction on the derivation.

Let us consider cases with modality.

1) If $\Gamma \vdash_{NIEL^-_{\wedge,\to}} A$, then $IEL^- \vdash \bigwedge \Gamma \to \mathbf{K}A$.

(1)	$\bigwedge \Gamma \to A$	assumption
(2)	$A \to \mathbf{K}A$	co-reflection
(3)	$(\bigwedge \Gamma \to A) \to ((A \to \mathbf{K}A) \to (\bigwedge \Gamma \to \mathbf{K}A))$	IPC theorem
(4)	$(A \to \mathbf{K}A) \to (\bigwedge \Gamma \to \mathbf{K}A)$	from (1), (3) and MP
(5)	$\bigwedge \Gamma \to \mathbf{K}A$	from (2), (4) and MP

2) If $\Gamma \vdash_{NIEL^-_{\wedge,\to}} \mathbf{K}\vec{A}$ and $\vec{A} \vdash B$, then $IEL^- \vdash \bigwedge \Gamma \to \mathbf{K}B$.

(1) $\quad \bigwedge \Gamma \to \bigwedge_{i=1}^{n} \mathbf{K}A_i$ $\qquad\qquad\qquad$ assumption

(2) $\quad \bigwedge_{i=1}^{n} \mathbf{K}A_i \to \mathbf{K} \bigwedge_{i=1}^{n} A_i$ $\qquad\qquad$ IEL theorem

(3) $\quad \bigwedge \Gamma \to \mathbf{K} \bigwedge_{i=1}^{n} A_i$ $\qquad\qquad\quad$ from (1), (2) and transitivity

(4) $\quad \bigwedge_{i=1}^{n} A_i \to B$ $\qquad\qquad\qquad$ assumption

(5) $\quad (\bigwedge_{i=1}^{n} A_i \to B) \to \mathbf{K}(\bigwedge_{i=1}^{n} A_i \to B)$ $\quad$ co-reflection

(6) $\quad \mathbf{K}(\bigwedge_{i=1}^{n} A_i \to B)$ $\qquad\qquad\quad$ from (4), (5) and MP

(7) $\quad \mathbf{K} \bigwedge_{i=1}^{n} A_i \to \mathbf{K}B$ $\qquad\qquad$ from (6) and normality

(8) $\quad \bigwedge \Gamma \to \mathbf{K}B$ $\qquad\qquad\qquad$ from (3), (7) and transitivity

$\hfill \square$

Lemma 2. *If $IEL^- \vdash A$, then $NIEL^- \vdash A$.*

Proof. Straightforward derivation of modal axioms in $NIEL^-$. We consider this derivation below using terms. $\hfill \square$

At the next step we build the typed lambda-calculus based on $NIEL^-_{\wedge,\to}$ by proof-assingment in rules.

At first, we define lambda-terms and types for this lambda-calculus.

Definition 3. *The set of terms:*
Let $\mathbb{V}$ be the set of variables. The set $\Lambda_{\mathbf{K}}$ of terms is defined by the grammar:
$$\Lambda_{\mathbf{K}} ::= \mathbb{V} \mid (\lambda\Lambda.\Lambda_{\mathbf{K}}) \mid (\Lambda_{\mathbf{K}}\Lambda_{\mathbf{K}}) \mid (\Lambda_{\mathbf{K}}, \Lambda_{\mathbf{K}}) \mid (\pi_1\Lambda_{\mathbf{K}}) \mid (\pi_2\Lambda_{\mathbf{K}}) \mid$$
$$(\mathbf{pure}\ \Lambda_{\mathbf{K}}) \mid (\mathbf{let\ pure}\ \Lambda_{\mathbf{K}} = \Lambda_{\mathbf{K}}\ \mathbf{in}\ \Lambda_{\mathbf{K}})$$

Definition 4. *The set of types:*
Let $\mathbb{T}$ be the set of atomic types. The set $\mathbb{T}_{\mathbf{K}}$ of types with applicative functor $\mathbf{K}$ is generated by the grammar:

$$\mathbb{T}_{\mathbf{K}} ::= \mathbb{T} \mid (\mathbb{T}_{\mathbf{K}} \to \mathbb{T}_{\mathbf{K}}) \mid (\mathbb{T}_{\mathbf{K}} \times \mathbb{T}_{\mathbf{K}}) \mid (\mathbf{K}\mathbb{T}_{\mathbf{K}}) \qquad\qquad (1)$$

Context, domain of context and range of context are defined standardly [11][12].

Our type system is based on the Curry-style typing rules:

Definition 5. *Modal typed lambda calculus $\lambda_{\mathbf{K}}$ based on $NIEL^-_{\wedge,\to}$:*

$$\frac{}{\Gamma, x : A \vdash x : A}\ ax$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \to_i \qquad\qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \to_e$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash N : B}{\Gamma \vdash \langle M, N \rangle : A \times B} \times_i \qquad \frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash \pi_i M : A_i} \times_e, \ i \in \{1, 2\}$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \textbf{pure } M : \mathbf{K}A} \ \mathbf{K}_I \qquad \frac{\Gamma \vdash \vec{M} : \mathbf{K}\vec{A} \qquad \vec{x} : \vec{A} \vdash N : B}{\Gamma \vdash \textbf{let pure } \vec{x} = \vec{M} \textbf{ in } N : \mathbf{K}B} \ let_{\mathbf{K}}$$

$\mathbf{K}_I$-typing rule is the same as $\bigcirc$-introduction in lax logic (also known as monadic metalanguage [17]) and in typed lambda-calculus which is derived by proof-assignment for lax-logic proofs. $\mathbf{K}_I$ allows to inject an object of type α into the functor. $\mathbf{K}_I$ reflects the Haskell method **pure** for Applicative class. It plays the same role as the **return** method in Monad class.

$let_{\mathbf{K}}$ is similar to the $\Box$-rule in typed lambda calculus for intuitionistic normal modal logic $\mathbf{IK}$, which is described in [19].

In fact, our calculus is the extention of typed lambda calculus for $\mathbf{IK}$ with typing rule appropriate to co-reflection.

Here are some examples of closed terms:

- $(\lambda x.\textbf{pure } x) : A \to \mathbf{K}A$;

- $\lambda f.\lambda x.\textbf{let pure } g, y = f, x \textbf{ in } gy : \mathbf{K}(A \to B) \to \mathbf{K}A \to \mathbf{K}B$

- $\lambda f.\lambda x.\textbf{let pure } g, y = \textbf{pure } f, x \textbf{ in } gy : (A \to B) \to \mathbf{K}A \to \mathbf{K}B$

Now we define free variables and substitutions. β-reduction, multi-step β-reduction and β-equality are defined standardly:

Definition 6. *Set $FV(M)$ of free variables for arbitrary term M:*
  1) $FV(x) = \{x\}$;
  2) $FV(\lambda x.M) = FV(M) \setminus \{x\}$;
  3) $FV(MN) = FV(M) \cup FV(N)$;
  4) $FV(\langle M, N \rangle) = FV(M) \cup FV(N)$;
  5) $FV(\pi_i M) \subseteq FV(M)$, $i \in \{1, 2\}$;
  6) $FV(pure \ M) = FV(M)$;
  7) $FV(\textbf{let pure } \vec{N} = \vec{M} \textbf{ in } M) = \bigcup\limits_{i=1}^{n} FV(M)$, where $n = |\vec{M}|$.

Definition 7. *Substitution:*
  1) $x[x := N] = N$, $x[y := N] = x$;
  2) $(MN)[x := N] = M[x := N]N[x := N]$;
  3) $(\lambda x.M)[x := N] = \lambda x.M[x := N]$;
  4) $(M, N)[x := P] = (M[x := P], N[x := P])$;
  5) $(\pi_i M)[x := P] = \pi_i(M[x := P])$, $i \in \{1, 2\}$;
  6) $(\textbf{pure } M)[x := P] = \textbf{pure } (M[x := P])$;
  7) $(\textbf{let pure } \vec{x} = \vec{M} \textbf{ in } N)[y := P] = \textbf{let pure } \vec{x} = (\vec{M}[y := P]) \textbf{ in } M$.

Definition 8. *Type substituition*

The substituition of type C for type variable B in type A inductively defined as follows:

  1) $B[B := C] = B$ and $D[B := C] = D$, if $B \neq D$;
  2) $(A_1 \alpha A_2)[B := C] = (A_1[B := C])\alpha(A_2[B := C])$, where $\alpha \in \{\to, \times\}$;
  3) $(\mathbf{K}A)[B := C] = \mathbf{K}(A[B := C])$.
  4) Let Γ be the context, then $\Gamma[B := C] = \{x : (A[B := C]) \mid x : A \in \Gamma\}$

Definition 9. *β-reduction and η-reduction rules for $\lambda_{\mathbf{K}}$.*

  1) $(\lambda x.M)N \to_\beta M[x := N]$;
  2) $\pi_1\langle M, N\rangle \to_\beta M$;
  3) $\pi_2\langle M, N\rangle \to_\beta N$;

  4)   **let pure** $\vec{x}, y, \vec{z} = \vec{M},$ **let pure** $\vec{w} = \vec{N}$ **in** $Q, \vec{P}$ **in** $R \to_\beta$
        let pure $\vec{x}, \vec{w}, \vec{z} = \vec{M}, \vec{N}, \vec{P}$ **in** $R[y := Q]$

  5) **let pure** $\vec{x} = $ **pure** $\vec{M}$ **in** $N \to_\beta$ **pure** $N[\vec{x} := \vec{M}]$
  6) $\lambda x.fx \to_\eta f$;
  7) $\langle\pi_1 P, \pi_2 P\rangle \to_\eta P$;
  8) **let pure** $x = M$ **in** $x \to_\eta M$;

By default we use call-by-name evaluation strategy.

3   Basic lemmas

Now we will prove standard lemmas for contexts in type systems[3]:

Lemma 3. *Generation for $\mathbf{K}_I$.*
*Let $\Gamma \vdash$ **pure** $M : \mathbf{K}A$, then $\Gamma \vdash M : A$;*

Proof. Induction on the structure of **pure** M. $\square$

Lemma 4. *Basic lemmas .*
  i) Let $\Gamma \vdash M : A$ and $\Gamma \subseteq \Delta$, then $\Delta \vdash M : A$;
  ii) Let $\Gamma, x : A \vdash M : B$ and $\Gamma \vdash N : A$, then $\Gamma \vdash M[x := N] : B$.
  iii) Let $\Gamma \vdash M : A$, then $\Gamma[B := C] \vdash M : (A[B := C])$.

Proof.
  i-ii-iii) Induction on $\Gamma \vdash M : A$.

       $\square$

Theorem 1. *Subject reduction*
Let $\Gamma \vdash M : A$ and $M \twoheadrightarrow_{\beta\eta} N$, then $\Gamma \vdash N : A$

Proof. For cases with application, abstraction and pairs see [12] [13].
  1) Let $\Gamma \vdash$ **let pure** $\vec{x}, y, \vec{z} = \vec{M},$ **let pure** $\vec{w} = \vec{N}$ **in** $Q, \vec{P}$ **in** $R : \mathbf{K}B$, then $\Gamma \vdash$ **let pure** $\vec{x}, \vec{w}, \vec{z} = \vec{M}, \vec{N}, \vec{P}$ **in** $R[y := Q] : \mathbf{K}B$
  2) Let $\Gamma \vdash$ **let pure** $x = M$ **in** $x : \mathbf{K}A$, then $\Gamma \vdash M : \mathbf{K}A$.
  See [19].
  3) If the derivation ends in

---

[3]We will not prove cases with $\to$-constructor, they are proved standardly in the same lemmas for simply typed lambda calculus, for example see [11] [12] [14]. We will consider only modal cases

$$\frac{\Gamma \vdash \mathbf{pure}\ \vec{M} : \mathbf{K}\vec{A} \qquad \vec{x} : \vec{A} \vdash N : B}{\Gamma \vdash \mathbf{let\ pure}\ \vec{x} = \mathbf{pure}\ \vec{M}\ \mathbf{in}\ N : \mathbf{K}B}$$

So $\Gamma \vdash \vec{M} : \vec{A}$ by generation and $\Gamma \vdash N[\vec{x} := \vec{M}] : B$ by weakening and substitution.

Then we can transform this into the next derivation:

$$\frac{\Gamma \vdash N[\vec{x} := \vec{M}] : B}{\Gamma \vdash \mathbf{pure}\ N[\vec{x} := \vec{M}] : \mathbf{K}B}\ \mathbf{K}_I$$

$\square$

Theorem 2.
$\twoheadrightarrow_\beta$ *is strongly normalizing;*

Proof.

We modify and apply Tait's technique of logical relation for modalities. For strong normalization proof with Tait's method for simply typed lambda calculus see [13].

Definition 10. *The set of strongly computable terms:*

- $SC_A = \{M : A \mid M$ *is strongly normalizing*$\}$ *for* $A \in \mathbb{T}$;

- $SC_{A \rightarrow B} = \{M : A \rightarrow B \mid \forall N \in SC_A, MN \in SC_B\}$, *for* $A, B \in \mathbb{T}_\mathbf{K}$

- $SC_{\mathbf{K}A} = \{M : \mathbf{K}A \mid M$ *is strongly normalizing*$\}$ *for* $A \in \mathbb{T}$;

- $SC_{\mathbf{K}(A \rightarrow B)} = \{M : \mathbf{K}(A \rightarrow B) | \forall f \in SC_{A \rightarrow B}, \forall x \in SC_A, \forall N \in SC_{\mathbf{K}A}, \mathbf{let\ pure}\ f, x = M, N\ \mathbf{in}\ fx\}$ *for* $A, B \in \mathbb{T}_\mathbf{K}$.

Lemma 5.

- *If* $M \in SC_A$, *then* M *is strongly normalizing;*

- *Let* $M \in SC_A$ *and* $M \rightarrow_\beta N$, *then* $N \in SC_A$;

- *Let* N *is non-introduced,* $N \in SC_A$. *Then, if* $M \rightarrow_\beta N$, *then* $M \in SC_A$;

Proof.

By induction on the structure of A.

1) $A \equiv \mathbf{K}A$, where $A \in \mathbb{T}$.

i) Follows from the definition;

ii) Immediately;

iii) Let N is non-introduced and $N \in SC_A$, such that $M \rightarrow_\beta N$. Any reduction path $M \rightarrow_\beta \ldots$ passes through $M \rightarrow_\beta N$.

N is strongly normalizing, so M too.

2) $A \equiv \mathbf{K}(B \rightarrow C)$

i) Suppose $M \in SC_{\mathbf{K}(B \rightarrow C)}$. Let $N \in SC_{\mathbf{K}B}$. So $\mathbf{let\ pure}\ f, x = M, N\ \mathbf{in}\ fx \in SC_{\mathbf{K}C}$.

So M is strongly normalizing, since $\mathbf{let\ pure}\ f, x = M, N\ \mathbf{in}\ fx$ is strongly normalizing.

ii) Let $M_1 \in SC_{\mathbf{K}(B \rightarrow C)}$ and $M_1 \rightarrow_\beta M_2$. Fix $N \in SC_{\mathbf{K}B}$.

Then **let pure** $f, x = M_1, N$ **in** $fx \in SC_{\mathbf{K}C}$.

Hence, **let pure** $f, x = M_1, N$ **in** $fx \in SC_{\mathbf{K}C} \to_\beta$ **let pure** $f, x = M_2, N$ **in** fx.

So **let pure** $f, x = M_2, N$ **in** $fx \in SC_{\mathbf{K}C}$. Then $M_2 \in SC_{\mathbf{K}(B \to C)}$.

iii) Let M_2 be non-introduced, $M_2 \in SC_{\mathbf{K}(B \to C)}$ and $M_1 \to_\beta M_2$.

Let $N \in SC_{\mathbf{K}B}$. So **let pure** $f, x = M_2, N$ **in** $fx \in SC_{\mathbf{K}C}$.

So **let pure** $f, x = M_1, N$ **in** $fx \to_\beta$ **let pure** $f, x = M_2, N$ **in** $fx \in SC_{\mathbf{K}C}$.

Thus **let pure** $f, x = M_1, N$ **in** $fx \in SC_{\mathbf{K}C}$ by IH, so $M_1 \in SC_{\mathbf{K}(B \to C)}$. $\square$

Lemma 6.

   If $M \in SC_A$, then **pure** $M \in SC_{\mathbf{K}A}$

Proof. Induction on the structure of M. $\square$

Lemma 7.

   Let $x_1 : A_1, \ldots, x_n : A_n \vdash M : A$, then forall i, $M_i \in SC_{A_i}$. Then $M[x_1 := M_1, \ldots, x_n := M_n] \in SC_A$.

Proof.

1) Let the derivation ends in:

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash M : A}{x_1 : A_1, \ldots, x_n : A_n \vdash \mathbf{pure}\, M : \mathbf{K}A}$$

By assumption $M[x_1 := M_1, \ldots, x_n := M_n] \in SC_A$, so **pure** $M[x_1 := M_1, \ldots, x_n := M_n] \in SC_{\mathbf{K}A}$.

2) Let the derivation ends in:

$$\frac{x_1 : A_1, \ldots, x_n : A_n \vdash \vec{M'} : \mathbf{K}\vec{A} \qquad \vec{x} : \vec{A} \vdash N : B}{x_1 : A_1, \ldots, x_n : A_n \vdash \mathbf{let\ pure}\ \vec{x} = \vec{M'}\ \mathbf{in}\ N : \mathbf{K}B}$$

By IH forall $i \in \{1, \ldots, \text{length}(\vec{M'})\}$, $M'_i[x_1 := M_1, \ldots, x_n := M_n] \in SC_{\mathbf{K}A_i}$.

So **let pure** $\vec{x} = \vec{M'}[x_1 := M_1, \ldots, x_n := M_n]$ **in** $N \in SC_{\mathbf{K}B}$, otherwise we can build infinite reduction path in $\vec{M'}[x_1 := M_1, \ldots, x_n := M_n]$. $\square$

Corollary 1. *All terms are strongly computable, therefore are strongly normalizing.*

$\square$

Theorem 3.

   $\twoheadrightarrow_\beta$ *is confluent.*

Proof. We modify and apply Barendregt's technique with term underlying. We will consider the fragment of the grammar for terms without constructors for pairs for simplicity.

Definition 11. *The set of underlined terms.*

- $x \in \mathbb{V} \Rightarrow x \in \underline{\Lambda}$;

- $M \in \underline{\Lambda} \Rightarrow (\lambda x.M) \in \underline{\Lambda}$;

- $M, N \in \underline{\Lambda} \Rightarrow (MN) \in \underline{\Lambda}$;

- $M \in \underline{\Lambda} \Rightarrow (\textbf{pure } M) \in \underline{\Lambda}$;

- $\vec{x} \in \mathbb{V}, \vec{M}, N \in \underline{\Lambda} \Rightarrow \textbf{let pure } \vec{x} = \vec{M} \textbf{ in } N \in \underline{\Lambda}$;

- $M, N \in \underline{\Lambda} \Rightarrow (\lambda_i x.M)N \in \underline{\Lambda}$, for all $i \in \mathbb{N}$.

Definition 12. *Subsitution for term with labelled lambda:*
$$((\lambda_i x.M)N)[y := Z] = (\lambda_i x.M[y := Z])(N[y := Z])$$

Definition 13. *Index erasing*
Let us define map $|.| : \underline{\Lambda} \to \Lambda$ as follows:

- $|x| = x$;

- $|\lambda x.M| = \lambda x.|M|$;

- $|MN| = |M||N|$;

- $|\textbf{pure } M| = \textbf{pure } |M|$;

- $|\textbf{let pure } \vec{x} = \vec{M} \textbf{ in } N| = \textbf{let pure } \vec{x} = |\vec{M}| \textbf{ in } |N|$;

- $|(\lambda_i x.M)N| = (\lambda x.M)N$

Definition 14. *Reduction rules:*

- $(\lambda x.M)N \to_{\underline{\beta}} M[x := N]$;

- $\dfrac{\textbf{let pure } \vec{x}, y, \vec{z} = \vec{M}, \textbf{let pure } \vec{w} = \vec{N} \textbf{ in } Q, \vec{P} \textbf{ in } R \to_{\underline{\beta}}}{\textbf{let pure } \vec{x}, \vec{w}, \vec{z} = \vec{M}, \vec{N}, \vec{P} \textbf{ in } R[y := Q]}$;

- $\textbf{let pure } \vec{x} = \textbf{pure } \vec{M} \textbf{ in } N \to_{\underline{\beta}} \textbf{pure } N[\vec{x} := \vec{M}]$;

- $(\lambda x_i.M)N \to_{\underline{\beta}} M[x := N]$

$\twoheadrightarrow_{\underline{\beta}}$ is a reflexive-transitive closure of $\to_{\underline{\beta}}$.

Definition 15. *Indexed redex erasing:*
Let us define the next map $\phi : \underline{\Lambda} \to \Lambda$:

- $\phi(x) = x$;

- $\phi(\lambda x.M) = \lambda x.\phi(M)$;

- $\phi(MN) = \phi(M)\phi(N)$;

- $\phi(\textbf{pure } M) = \textbf{pure } \phi(M)$;

- $\phi(\textbf{let pure } \vec{x} = \vec{M} \textbf{ in } N) = \textbf{let pure } \vec{x} = \phi(\vec{M}) \textbf{ in } \phi(N)$;

- $\phi((\lambda_i x.M)N) = M[x := N]$

Lemma 8. $\forall \underline{M}, \underline{N} \in \underline{\Lambda} \; \forall M, N \in \Lambda, if \; |\underline{M}| = M, |\underline{N}| = N, then$

- *If* $M \twoheadrightarrow_\beta N$*, then* $\underline{M} \twoheadrightarrow_{\underline{\beta}} \underline{N}$

- *Vice versa*

Proof. Induction on the generation $\to_\beta$ and $\to_{\underline{\beta}}$ correspondently. The general statement follows from transitivity of multi-step reductions of both types. □

Lemma 9. $\phi(M[x := N]) = \phi(M)[x := \phi(N)]$.

Proof. We treat only cases with **pure** and with **let**. For the rest cases see [15].

1)

$\phi(\mathbf{pure}\ (M[x := N])) =$ By the definition of ϕ
$\mathbf{pure}\ (\phi(M[x := N])) =$ Induction hypothesis
$\mathbf{pure}\ (\phi(M)[x := \phi(N)]) =$ Substitution definition
$(\mathbf{pure}\ \phi(M))[x := \phi(N)]$

2)

$\phi((\mathbf{let\ pure}\ \vec{x} = \vec{M}\ \mathbf{in}\ N)[y := P]) =$ Substitution definition
$\phi(\mathbf{let\ pure}\ \vec{x} = (\vec{M}[y := P])\ \mathbf{in}\ N) =$ By the definition of ϕ
$\mathbf{let\ pure}\ \vec{x} = \phi(\vec{M}[y := P])\ \mathbf{in}\ \phi(N) =$ Induction hypothesis
$\mathbf{let\ pure}\ \vec{x} = (\phi(\vec{M})[y := \phi(P)])\ \mathbf{in}\ \phi(N) =$ Substitution definition
$(\mathbf{let\ pure}\ \vec{x} = \phi(\vec{M})\ \mathbf{in}\ \phi(N))[y := \phi(P)]$

□

Lemma 10.

- If $M \twoheadrightarrow_{\underline{\beta}} N$, then $\phi(M) \twoheadrightarrow_\beta \phi(N)$

- If $|M| = N$ and $\phi(M) = P$, then $N \twoheadrightarrow_\beta P$.
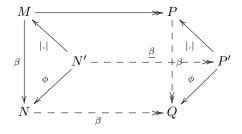
Proof.

i) Induction on the generation of $\twoheadrightarrow_{\underline{\beta}}$ using previous lemma.
ii) Induction on the structure of M. □

Lemma 11. *Strip lemma.*

If $M \to_\beta N$ and $M \twoheadrightarrow_\beta P$. Then there exists some term Q, such that $N \twoheadrightarrow_\beta Q$ and $P \twoheadrightarrow_\beta Q$.

Proof. Proof is similar to [15] [18]. We build the following diagram, which commutes by lemmas 8 and 10.

□

Corollary 2. *If $M \twoheadrightarrow_\beta N$ and $M \twoheadrightarrow_\beta P$. Then there exists some term Q, such that $N \twoheadrightarrow_\beta Q$ and $P \twoheadrightarrow_\beta Q$.*

Proof. Unfold $M \twoheadrightarrow_\beta N$ as the sequence of one-step reductions and apply strip lemma on the every step. $\qquad\square$

$\square$

Theorem 4.

Normal form in call-by-name $\lambda_{\mathbf{K}}$ has the subformula property: if M is in normal formal, then its all subterms are in normal form too.

Proof. By induction on the structure of M. Case with **let pure** $\vec{x} = \vec{M}$ **in** N was considered by Kakutani [19] [20]. Similarly, if **pure** M is in normal form, so M is in normal form to and has subformula property by hypothesis. So **pure** M has subformula property. $\qquad\square$

4 Categorical semantics

Definition 16. *Monoidal functor*

Let $\langle \mathcal{C}, \otimes_1, \mathbb{1} \rangle$ and $\langle \mathcal{D}, \otimes_2, \mathbb{1}' \rangle$ are monoidal categories.

A monoidal functor $\mathcal{F} : \langle \mathcal{C}, \otimes_1, \mathbb{1} \rangle \to \langle \mathcal{D}, \otimes_2, \mathbb{1}' \rangle$ is a functor $\mathcal{F} : \mathcal{C} \to \mathcal{D}$ with additional natural transformations, which satisfy the well-known conditions described in [23]:

1) $u : \mathbb{1}' \to \mathcal{F}\mathbb{1}$;

2) $*_{A,B} : \mathcal{F}A \otimes_2 \mathcal{F}B \to \mathcal{F}(A \otimes_1 B)$.

Definition 17. *Applicative functor*

An applicative functor is a triple $\langle \mathcal{C}, \mathcal{K}, \eta \rangle$, where $\mathcal{C}$ is a symmetric monoidal category, $\mathcal{K}$ is a monoidal endofunctor and $\eta : Id_{\mathcal{C}} \Rightarrow \mathcal{K}$ is a natural transformation (similar to unit in monad), such that:

1) $u = \eta_{\mathbb{1}}$;

2) $*_{A,B} \circ (\eta_A \otimes \eta_B) = \eta_{A \otimes B}$;

3) Weak commutativity condition:

$$
\begin{array}{ccccc}
A \otimes \mathcal{K}B & \xrightarrow{\eta_A \otimes id_{\mathcal{K}B}} & \mathcal{K}A \otimes \mathcal{K}B & \xrightarrow{*_{A,B}} & \mathcal{K}(A \otimes B) \\
\downarrow{\sigma_{A,\mathcal{K}B}} & & & & \downarrow{\mathcal{K}(\sigma_{A,B})} \\
\mathcal{K}B \otimes A & \xrightarrow{id_{\mathcal{K}B} \otimes \eta_A} & \mathcal{K}B \otimes \mathcal{K}A & \xrightarrow{*_{B,A}} & \mathcal{K}(B \otimes A)
\end{array}
$$

4.1 Soundness and completeness

Theorem 5. *Soundness*

Let $\Gamma \vdash M : A$ and $M =_{\beta\eta} N$, then $[\![\Gamma \vdash M : A]\!] = [\![\Gamma \vdash N : A]\!]$

Proof.

Definition 18. *Semantical translation from $\lambda_{\mathbf{K}}$ to some cartesian closed category $\mathcal{C}$ with applicative functor $\mathcal{K}$:*

1) Interpretation for types:

$[\![A]\!] := \hat{A}, A \in \mathbb{T}$, where $\hat{A}$ is an object of $\mathcal{C}$ obtained by some given assignment;

$[\![A \to B]\!] := [\![A]\!] \to [\![B]\!]$;

$[\![A \times B]\!] := [\![A]\!] \times [\![B]\!]$.

2) Interpretation for modal types: $[\![\mathbf{K}A]\!] = \mathcal{K}[\![A]\!]$;

3) Interpretaion for contexts:
$[\![\Gamma = \{x_1 : A_1, ..., x_n : A_n\}]\!] := [\![\Gamma]\!] = [\![A_1]\!] \times ... \times [\![A_n]\!]$;

4) Interpretation for typing assignment: $[\![\Gamma \vdash M : A]\!] := [\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$.

5) Interpretation for typing rules:

$$\frac{}{[\![\Gamma, x : A \vdash x : A]\!] = \pi_2 : [\![\Gamma]\!] \times [\![A]\!] \to [\![A]\!]}$$

$$\frac{[\![\Gamma, x : A \vdash M : B]\!] = f : [\![\Gamma]\!] \times [\![A]\!] \to [\![B]\!]}{[\![\Gamma \vdash (\lambda x.M) : A \to B]\!] = \Lambda(f) : [\![\Gamma]\!] \to [\![B]\!]^{[\![A]\!]}}$$

$$\frac{[\![\Gamma \vdash M : A \to B]\!] = [\![M]\!] : [\![\Gamma]\!] \to [\![B]\!]^{[\![A]\!]} \qquad [\![\Gamma \vdash N : A]\!] = [\![N]\!] : [\![\Gamma]\!] \to [\![A]\!]}{[\![\Gamma \vdash (MN) : B]\!] = [\![\Gamma]\!] \xrightarrow{\langle [\![M]\!], [\![N]\!]\rangle} [\![B]\!]^{[\![A]\!]} \times [\![A]\!] \xrightarrow{\epsilon} [\![B]\!]}$$

$$\frac{[\![\Gamma \vdash M : A]\!] = f : [\![\Gamma]\!] \to [\![A]\!] \qquad [\![\Gamma \vdash N : B]\!] = g : [\![\Gamma]\!] \to [\![B]\!]}{[\![\Gamma \vdash (M, N) : A \times B]\!] = \langle f, g \rangle : [\![\Gamma]\!] \to [\![A]\!] \times [\![B]\!]}$$

$$\frac{[\![\Gamma \vdash p : A_1 \times A_2]\!] = f : [\![\Gamma]\!] \to [\![A_1]\!] \times [\![A_2]\!]}{[\![\Gamma \vdash \pi_i p : A_i]\!] = [\![\Gamma]\!] \xrightarrow{f} [\![A_1]\!] \times [\![A_2]\!] \xrightarrow{\pi_i} [\![A_i]\!]} \quad i \in \{1, 2\}$$

$$\frac{[\![\Gamma \vdash M : A]\!] = [\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]}{[\![\Gamma \vdash \mathbf{pure}\ M : \mathbf{K}A]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![A]\!] \xrightarrow{\eta_{[\![A]\!]}} \mathcal{K}[\![A]\!]}$$

$$\frac{[\![\Gamma \vdash \vec{M} : \mathbf{K}\vec{A}]\!] = \langle [\![M_1]\!], \dots, [\![M_n]\!] \rangle : [\![\Gamma]\!] \to \prod_{i=1}^{n} \mathcal{K}[\![A_i]\!] \qquad [\![\vec{x} : \vec{A} \vdash N : B]\!] = [\![N]\!] : \prod_{i=1}^{n} [\![A_i]\!] \to [\![B]\!]}{[\![\Gamma \vdash \mathbf{let\ pure}\ \vec{x} = \vec{M}\ \mathbf{in}\ M : \mathbf{K}B]\!] = \mathcal{K}([\![N]\!]) \circ *_{[\![A_1]\!], \dots, [\![A_n]\!]} \circ \langle [\![M_1]\!], \dots, [\![M_n]\!] \rangle : [\![\Gamma]\!] \to \mathcal{K}[\![B]\!]}$$

Definition 19. *Simultaneous substitution*

Let $\Gamma = \{x_1 : A_1, ..., x_n : A_n\}$, $\Gamma \vdash M : A$ *and for all* $i \in \{1, ..., n\}$,
$\Gamma \vdash M_i : A_i$.

We define simultaneous substitution $M[\vec{x} := \vec{M}]$ *recursively by:*

1) $x_i[\vec{x} := \vec{M}] = M_i$;

2) $(\lambda x.M)[\vec{x} := \vec{M}] = \lambda x.(M[\vec{x} := \vec{M}])$;

3) $(MN)[\vec{x} := \vec{M}] = (M[\vec{x} = \vec{M}])(N[\vec{x} := \vec{M}])$;

4) $\langle M, N \rangle = \langle (M[\vec{x} = \vec{M}]), (N[\vec{x} := \vec{M}]) \rangle$;

5) $(\pi_i P)[\vec{x} := \vec{M}] = \pi_i(P[\vec{x} = \vec{M}])$;

6) $(\mathbf{pure}\ M)[\vec{x} := \vec{M}] = \mathbf{pure}\ (M[\vec{x} = \vec{M}])$;

7) $(\mathbf{let\ pure}\ \vec{x} = \vec{M}\ \mathbf{in}\ N)[\vec{y} := \vec{P}] = \mathbf{let\ pure}\ \vec{x} = (\vec{M}[\vec{y} := \vec{P}])\ \mathbf{in}\ N$

Lemma 12.

$[\![M[x_1 := M_1, \dots, x_n := M_n]]\!] = [\![M]\!] \circ \langle [\![M_1]\!], \dots, [\![M_n]\!] \rangle$.

Proof.

    1) $[\![\Gamma \vdash (\mathbf{pure}\ M)[\vec{x} := \vec{M}] : \mathbf{K}A]\!] = [\![\Gamma \vdash \mathbf{pure}\ M : \mathbf{K}A]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle.$

$[\![\Gamma \vdash (\mathbf{pure}\ M)[\vec{x} := \vec{M}] : \mathbf{K}A]\!] = [\![\Gamma \vdash \mathbf{pure}\ (M[\vec{x} := \vec{M}]) : \mathbf{K}A]\!]$	Substitution definition
$= \eta_{[\![A]\!]} \circ [\![(M[\vec{x} := \vec{M}])]\!]$	Translation for pure
$= \eta_{[\![A]\!]} \circ ([\![M]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle)$	Induction hypothesis
$= (\eta_{[\![A]\!]} \circ [\![M]\!]) \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle$	Associativity of composition
$= [\![\Gamma \vdash \mathbf{pure}\ M : \mathbf{K}A]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle$	Translation for pure

    2) $[\![\Gamma \vdash (\mathbf{let\ pure}\ \vec{x} = \vec{M}\ \mathbf{in}\ N)[\vec{y} := \vec{P}] : \mathbf{K}B]\!] = [\![\Gamma \vdash \mathbf{let\ pure}\ \vec{x} = \vec{M}\ \mathbf{in}\ N : \mathbf{K}B]\!] \circ \langle [\![P_1]\!], \ldots, [\![P_n]\!]\rangle$

$[\![\Gamma \vdash (\mathbf{let\ pure}\ \vec{x} = \vec{M}\ \mathbf{in}\ N)[\vec{y} := \vec{P}] : \mathbf{K}B]\!] =$	Substitution definition
$[\![\Gamma \vdash \mathbf{let\ pure}\ \vec{x} = (\vec{M}[\vec{y} := \vec{P}])\ \mathbf{in}\ N : \mathbf{K}B]\!] =$	Interpretaion for $let_\mathbf{K}$
$\mathcal{K}([\![N]\!]) \circ *_{[\![A_1]\!],\ldots,[\![A_n]\!]} \circ [\![\Gamma \vdash (\vec{M}[\vec{y} := \vec{P}]) : \mathbf{K}\vec{A}]\!] =$	Induction hypothesis
$\mathcal{K}([\![N]\!]) \circ *_{[\![A_1]\!],\ldots,[\![A_n]\!]} \circ ([\![\vec{M}]\!] \circ \langle [\![P_1]\!], \ldots, [\![P_n]\!]\rangle) =$	Associativity of composition
$(\mathcal{K}([\![N]\!]) \circ *_{[\![A_1]\!],\ldots,[\![A_n]\!]} \circ [\![\vec{M}]\!]) \circ \langle [\![P_1]\!], \ldots, [\![P_n]\!]\rangle =$	By interpretation
$[\![\Gamma \vdash \mathbf{let\ pure}\ \vec{x} = \vec{M}\ \mathbf{in}\ N : \mathbf{K}B]\!] \circ \langle [\![P_1]\!], \ldots, [\![P_n]\!]\rangle$	

<div align="right">□</div>

Lemma 13.

    Let $\Gamma \vdash M : A$ and $M \twoheadrightarrow_{\beta\eta} N$, then $[\![\Gamma \vdash M : A]\!] = [\![\Gamma \vdash N : A]\!]$;

Proof.

    Cases with β-reductions for $let_\mathbf{K}$ are shown in [20]. Let us consider cases with **pure**.

    1) $[\![\Gamma \vdash \mathbf{let\ pure}\ \vec{x} = \mathbf{pure}\ \vec{M}\ \mathbf{in}\ N : \mathbf{K}B]\!] = [\![\Gamma \vdash \mathbf{pure}\ N[\vec{x} := \vec{M}] : \mathbf{K}B]\!]$

$[\![\Gamma \vdash \mathbf{let\ pure}\ \vec{x} = \mathbf{pure}\ \vec{M}\ \mathbf{in}\ N : \mathbf{K}B]\!] =$
        By interpretation
$\mathcal{K}([\![N]\!]) \circ *_{[\![A_1]\!],\ldots,[\![A_n]\!]} \circ \langle \eta_{[\![A_1]\!]} \circ [\![M_1]\!], \ldots, \eta_{[\![A_n]\!]} \circ [\![M_n]\!]\rangle =$
        By the property of a pair of morphisms
$\mathcal{K}([\![N]\!]) \circ *_{[\![A_1]\!],\ldots,[\![A_n]\!]} \circ (\eta_{[\![A_1]\!]} \times \cdots \times \eta_{[\![A_n]\!]}) \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle =$
        Associativity of composition
$\mathcal{K}([\![N]\!]) \circ (*_{[\![A_1]\!],\ldots,[\![A_n]\!]} \circ (\eta_{[\![A_1]\!]} \times \ldots \eta_{[\![A_n]\!]})) \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle =$
        By the definition of an applicative functor
$\mathcal{K}([\![N]\!]) \circ \eta_{[\![A_1]\!] \times \cdots \times [\![A_n]\!]} \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle =$
        Naturality of η
$\eta_{[\![B]\!]} \circ [\![N]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle =$
        Associativity of composition
$\eta_{[\![B]\!]} \circ ([\![N]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle)\rangle =$
        Simultaneous substitution lemma
$\eta_{[\![B]\!]} \circ [\![N[\vec{x} := \vec{M}]]\!]$
        By interpetation
$[\![\Gamma \vdash \mathbf{pure}\ (N[\vec{x} := \vec{M}]) : \mathbf{K}B]\!]$

<div align="right">□</div>

<div align="right">□</div>

Theorem 6. *Completeness*
Let $\llbracket \Gamma \vdash M : A \rrbracket = \llbracket \Gamma \vdash N : A \rrbracket$, *then* $M =_{\beta\eta} N$.

Proof.

We will consider term model for simply typed lambda calculus $\times$ and $\to$ standardly described in [22]:

Definition 20. *Equivalence on term pairs:*
Let us define relation $\sim_{A,B} \subseteq \mathbb{V} \times \Lambda_{\mathbf{K}}$, *such that:*
$(x, M) \sim_{A,B} (y, N) \Leftrightarrow x : A \vdash M : B \ \& \ y : A \vdash N : A \ \& \ M =_{\beta\eta} N[y := x];$

We will denote equivalence class as $[x, M]_{A,B} = \{(y, N) | (x, M) \sim_{A,B} (y, N)\}$ (we will drop indeces below).

Definition 21. *Category* $\mathcal{C}(\lambda)$:

- $Ob_{\mathcal{C}} = \{\hat{A} \mid A \in \mathbb{T}\} \cup \{\mathbb{1}\};$

- $Hom_{\mathcal{C}(\lambda)}(\hat{A}, \hat{B}) = (\mathbb{V} \times \Lambda_{\mathbf{K}})/_{\sim_{A,B}};$

- *Let* $[x, M] \in Hom_{\mathcal{C}(\lambda)}(\hat{A}, \hat{B})$ *and* $[y, N] \in Hom_{\mathcal{C}(\lambda)}(\hat{B}, \hat{C})$, *then* $[y, M] \circ [x, M] = [x, N[y := M]];$

- *Identity morphism* $id_{\hat{A}} = [x, x] \in Hom_{\mathcal{C}(\lambda)(\hat{A})};$

- $\mathbb{1}$ *is a terminal object;*

- $\widehat{A \times B} = \hat{A} \times \hat{B};$

- *Canonical projection is defined as* $[x, \pi_i x] \in Hom_{\mathcal{C}(\lambda)}(\hat{A}_1 \times \hat{A}_2, \hat{A}_i)$ *for* $i \in \{1, 2\};$

- $\widehat{A \to B} = \hat{B}^{\hat{A}};$

- *Evaluation arrow* $\epsilon = [x, (\pi_2 x)(\pi_1 x)] \in Hom_{\mathcal{C}(\lambda)(\hat{B}^{\hat{A}} \times \hat{A}, \hat{B})}.$

It is sufficient to show $\mathbf{K}$ is an applicative functor on $\mathcal{C}(\lambda)$.

Definition 22. *Let us define an endofunctor* $\mathcal{K} : \mathcal{C}(\lambda) \to \mathcal{C}(\lambda)$, *such that forall* $[x, M] \in Hom_{\mathcal{C}(\lambda)}(\hat{A}, \hat{B}), \mathbf{K}([x, M]) = [y, \mathbf{let\ pure}\ x = y\ \mathbf{in}\ M] \in Hom_{\mathcal{C}(\lambda)}(\mathbf{K}\hat{A}, \mathbf{K}\hat{B})$ *(denotation: fmap f for an arbitrary arrow f).*

Lemma 14. *Functoriality*

- *fmap* $(g \circ f) = fmap\ (g) \circ fmap\ (f);$

- *fmap* $(id_{\hat{A}}) = id_{\mathbf{K}\hat{A}}.$

Proof. Easy checking using reduction rules.

$\square$

Definition 23. *Let us define natural transformations:*
1) $\eta : Id \Rightarrow \mathcal{K}$, *s. t.* $\forall \hat{A} \in Ob_{\mathcal{C}(\lambda)}$, $\eta_{\hat{A}} = [x, \mathbf{pure}\ x] \in Hom_{\mathcal{C}(\lambda)}(\hat{A}, \mathbf{K}\hat{A});$
2) $*_{A,B} : \mathbf{K}\hat{A} \times \mathbf{K}\hat{B} \to \mathbf{K}(\hat{A} \times \hat{B})$, *s. t.* $\forall \hat{A}, \hat{B} \in Ob_{\mathcal{C}(\lambda)}, *_{\hat{A}, \hat{B}} = [p, \mathbf{let\ pure}\ x, y = \pi_1 p, \pi_2 p\ \mathbf{in}\ \langle x, y \rangle] \in Hom_{\mathcal{C}(\lambda)}(\mathbf{K}A \times \mathbf{K}B, \mathbf{K}(A \times B)).$

Implementation for $*$ in our term model is a modification of $\mathbf{let_K}$-rule:

13

$$\dfrac{\dfrac{p : \mathbf{K}A \times \mathbf{K}B \vdash p : \mathbf{K}A \times \mathbf{K}B}{p : \mathbf{K}A \times \mathbf{K}B \vdash \pi_1 p : \mathbf{K}A} \quad \dfrac{p : \mathbf{K}A \times \mathbf{K}B \vdash p : \mathbf{K}A \times \mathbf{K}B}{p : \mathbf{K}A \times \mathbf{K}B \vdash \pi_2 p : \mathbf{K}B} \quad \dfrac{x : A \vdash x : A \quad\quad y : B \vdash y : B}{x : A, y : B \vdash \langle x, y \rangle : A \times B}}{p : \mathbf{K}A \times \mathbf{K}B \vdash \mathbf{let\ pure}\ x, y = \pi_1 p, \pi_2 p\ \mathbf{in}\ \langle x, y \rangle : \mathbf{K}(A \times B)}$$

Lemma 15.

$\quad$ **K** *is a monoidal endofunctor*

Proof.

$\quad$ See [19] $\hfill \square$

Lemma 16. *Naturality for η and for $*$*

- *fmap $f \circ \eta_A = \eta_B \circ f$;*

- $*_{\hat{A}, \hat{B}} \circ (\eta_A \times \eta_B) = \eta_{\hat{A} \times \hat{B}}$;

Proof.

$\quad$ i) fmap $f \circ \eta_{\hat{A}} = \eta_{\hat{B}} \circ f$

$\eta_{\hat{B}} \circ f =$	By the definition
$[y, \mathbf{pure}\ y] \circ [x, M] =$	By the definition of composition
$[x, \mathbf{pure}\ y[y := M]] =$	By substitution
$[x, \mathbf{pure}\ M]$	

On the other hand:

fmap $f \circ \eta_{\hat{A}} =$	By the definiton
$[z, \mathbf{let\ pure}\ x = z\ \mathbf{in}\ M] \circ [x, \mathbf{pure\ x}] =$	By the definition of composition
$[x, \mathbf{let\ pure}\ x = z\ \mathbf{in}\ M[z := \mathbf{pure}\ x]] =$	By substitution
$[x, \mathbf{let\ pure}\ x = \mathbf{pure}\ x\ \mathbf{in}\ M] =$	β-reduction rule
$[x, \mathbf{pure}\ M[x := x]] =$	By substitution
$[x, \mathbf{pure}\ M]$	

$\quad$ ii) $*_{\hat{A}, \hat{B}} \circ (\eta_{\hat{A}} \times \eta_{\hat{B}}) = \eta_{\hat{A} \times \hat{B}}$

$*_{\hat{A}, \hat{B}} \circ (\eta_{\hat{A}} \times \eta_{\hat{B}}) =$
By unfolding
$[q, \mathbf{let\ pure}\ x, y = \pi_1 q, \pi_2 q\ \mathbf{in}\ \langle x, y \rangle] \circ [p, \langle \mathbf{pure}\ (\pi_1 p), \mathbf{pure}\ (\pi_2 p) \rangle] =$
Composition
$[p, \mathbf{let\ pure}\ x, y = \pi_1 q, \pi_2 q\ \mathbf{in}\ \langle x, y \rangle[q := \langle \mathbf{pure}\ (\pi_1 p), \mathbf{pure}\ (\pi_2 p) \rangle]] =$
By substitution
$[p, \mathbf{let\ pure}\ x, y = \pi_1(\langle \mathbf{pure}\ (\pi_1 p), \mathbf{pure}\ (\pi_2 p) \rangle), \pi_2(\langle \mathbf{pure}\ (\pi_1 p), \mathbf{pure}\ (\pi_2 p) \rangle)\ \mathbf{in}\ \langle x, y \rangle] =$
Reduction rules
$[p, \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \mathbf{pure}\ (\pi_2 p)\ \mathbf{in}\ \langle x, y \rangle] =$
Reduction rule
$[p, \mathbf{pure}\ (\langle x, y \rangle[x := \pi_1 p, y := \pi_2 p])] =$
Substitution
$[p, \mathbf{pure}\ \langle \pi_1 p, \pi_2 p \rangle] =$
η-reduction
$[p, \mathbf{pure}\ p] =$
By definition
$\eta_{\hat{A} \times \hat{B}}$

□

Tensorial strength is defined as follows:

Definition 24. *Tensorial strength*

Let $[p, \langle \mathbf{pure}\ (\pi_1 p), \pi_2 p \rangle] \in Hom_{\mathcal{C}(\lambda)}(\hat{A} \times \mathbf{K}\hat{B}, \mathbf{K}\hat{A} \times \mathbf{K}\hat{B})$.

So tensorial strength is defined as $\tau_{\hat{A},\hat{B}} = *_{\hat{A},\hat{B}} \circ [p, \langle \mathbf{pure}\ (\pi_1 p), \pi_2 p \rangle]$.

It is clearly that tensorial strength defined above can be simplified as follows:

$*_{\hat{A},\hat{B}} \circ [p, \langle \mathbf{pure}\ (\pi_1 p), \pi_2 p \rangle] = \qquad\qquad$ By definition

$[p', \mathbf{let\ pure}\ x, y = \pi_1 p', \pi_2 p'\ \mathbf{in}\ \langle x, y \rangle] \circ [p, \langle \mathbf{pure}\ (\pi_1 p), \pi_2 p \rangle] = \qquad$ By composition

$[p, \mathbf{let\ pure}\ x, y = \pi_1 p', \pi_2 p'\ \mathbf{in}\ \langle x, y \rangle[p' := \langle \mathbf{pure}\ (\pi_1 p), \pi_2 p \rangle]] = \qquad$ By substitution

$[p, \mathbf{let\ pure}\ x, y = \pi_1(\langle \mathbf{pure}\ (\pi_1 p), \pi_2 p \rangle), \pi_2(\langle \pi_1 p, \mathbf{pure}\ (\pi_2 p) \rangle)\ \mathbf{in}\ \langle x, y \rangle] = \quad$ By β-reduction rules

$[p, \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle x, y \rangle]$

Lemma 17. *Weak commutativity.*

$fmap\ ([p, \langle \pi_2 p, \pi_1 p \rangle]) \circ \tau_{\hat{A},\hat{B}} =$

$*_{\hat{B},\hat{A}} \circ [q, \langle \pi_1 q, \mathbf{pure}\ (\pi_2 q) \rangle] \circ [p, \langle \pi_2 p, \pi_1 p \rangle]$

Proof.

$fmap\ ([r, \langle \pi_2 r, \pi_1 r \rangle]) \circ \tau_{\hat{A},\hat{B}} =$

By the definition of τ

$fmap\ ([r, \langle \pi_2 r, \pi_1 r \rangle]) \circ [p, \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle x, y \rangle] =$

By the definition of fmap

$[q, \mathbf{let\ pure}\ r = q\ \mathbf{in}\ \langle \pi_2 r, \pi_1 r \rangle] \circ [p, \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle x, y \rangle] =$

Composition

$[p, \mathbf{let\ pure}\ r = q\ \mathbf{in}\ \langle \pi_2 r, \pi_1 r \rangle[q := \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle x, y \rangle]] =$

By β-reduction rules

$[p, \mathbf{let\ pure}\ r = (\mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle x, y \rangle)\ \mathbf{in}\ \langle \pi_2 r, \pi_1 r \rangle] =$

By β-reduction rules

$[p, \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle \pi_2 r, \pi_1 r \rangle[r := \langle x, y \rangle]] =$

By substitution

$[p, \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle \pi_2 \langle x, y \rangle, \pi_1 \langle x, y \rangle \rangle] =$

By β-reduction rules

$[p, \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle y, x \rangle] =$

On the other hand

$*_{\hat{B},\hat{A}} \circ [q, \langle \pi_1 q, \mathbf{pure}\ (\pi_2 q) \rangle] \circ [p, \langle \pi_2 p, \pi_1 p \rangle] =$

By the definition of $*$

$[r, \mathbf{let\ pure}\ y, x = \pi_1 r, \pi_2 r\ \mathbf{in}\ \langle y, x \rangle] \circ [q, \langle \pi_1 q, \mathbf{pure}\ (\pi_2 q) \rangle] \circ [p, \langle \pi_2 p, \pi_1 p \rangle] =$

Composition

$[r, \mathbf{let\ pure}\ y, x = \pi_1 r, \pi_2 r\ \mathbf{in}\ \langle y, x \rangle] \circ [p, \langle \pi_1 q, \mathbf{pure}\ (\pi_2 q) \rangle[q := \langle \pi_2 p, \pi_1 p \rangle]] =$

By subsitution and by β-reduction rules

$[r, \mathbf{let\ pure}\ y, x = \pi_1 r, \pi_2 r\ \mathbf{in}\ \langle y, x \rangle] \circ [p, \langle \pi_2 p, \mathbf{pure}\ (\pi_1 p) \rangle]] =$

Composition

$[p, \mathbf{let\ pure}\ y, x = \pi_1 r, \pi_2 r\ \mathbf{in}\ \langle y, x \rangle[r := \langle \pi_2 p, \mathbf{pure}\ (\pi_1 p) \rangle]] =$

By subsitution and by β-reduction rules

$[p, \mathbf{let\ pure}\ y, x = \pi_2 p, \mathbf{pure}\ (\pi_1 p)\ \mathbf{in}\ \langle y, x \rangle] =$

By symmetricity of assingment

$[p, \mathbf{let\ pure}\ x, y = \mathbf{pure}\ (\pi_1 p), \pi_2 p\ \mathbf{in}\ \langle y, x \rangle]$

□

15

Lemma 18. K *is an applicative functor*

Proof. Immediately follows from previous lemmas in the section. □

Similar to [24], we apply the translation from $\lambda_{\mathbf{K}}$ to some cartesian closed category with an abritraty applicative functor $\mathcal{K}$, then we have $[\![\Gamma \vdash M : A]\!] = [x, M[x_i := \pi_i x]]$, so $M =_{\beta\eta} N \Leftrightarrow [\![\Gamma \vdash M : A]\!] = [\![\Gamma \vdash N : A]\!]$.

□

References

[1] Artemov S. and Protopopescu T., "Intuitionistic Epistemic Logic", *The Review of Symbolic Logic*, 2016, vol. 9, no 2. pp. 266-298.

[2] Krupski V. N. and Yatmanov A., "Sequent Calculus for Intuitionistic Epistemic Logic IEL", *Logical Foundations of Computer Science: International Symposium, LFCS 2016, Deerfield Beach, FL, USA, January 4-7, 2016. Proceedings*, 2016, pp. 187-201.

[3] Haskell Language. // URL: https://www.haskell.org. (Date: 1.08.2017)

[4] Idris. A Language with Dependent Types.// URL:https://www.idris-lang.org. (Date: 1.08.2017)

[5] Purescript. A strongly-typed functional programming language that compiles to JavaScript. URL: http://www.purescript.org. (Date: 1.08.2017)

[6] Elm. A delightful language for reliable webapps. // URL: http://elm-lang.org. (Date: 1.08.2017)

[7] Hackage, "The base package" // URL: https://hackage.haskell.org/package/base-4.10.0.0 (Date: 1.08.2017)

[8] Lipovaca M, "Learn you a Haskell for Great Good!". //URL: http://learnyouahaskell.com/chapters (Date: 1.08.2017)

[9] McBride C. and Paterson R., "Applicative programming with effects", *Journal of Functional Programming*, 2008, vol. 18, no 01. pp 1-13.

[10] McBride C. and Paterson R, "Functional Pearl. Idioms: applicative programming with effects", *Journal of Functional Programming*, 2005. vol. 18, no 01. pp 1-20.

[11] R. Nederpelt and H. Geuvers, "Type Theory and Formal Proof: An Introduction". *Cambridge University Press*, New York, NY, USA, 2014. pp. 436.

[12] Sorensen M. H. and Urzyczyn P, "Lectures on the Curry-Howard isomorphism", *Studies in Logic and the Foundations of Mathematics*, vol. 149, *Elsevier Science*, 1998. pp 261.

[13] Pierce B. C., "Types and Programming Languages". *Cambridge, Mass: The MIT Press*, 2002. pp. 605.

[14] Girard J.-Y., Taylor P. and Lafont Y, "Proofs and Types", *Cambridge University Press*, New York, NY, USA, 1989. pp. 175.

[15] Barendregt. H. P., "Lambda calculi with types" // Abramsky S., Gabbay Dov M., and S. E. Maibaum, "Handbook of logic in computer science (vol. 2), Osborne Handbooks Of Logic In Computer Science", Vol. 2. *Oxford University Press, Inc.*, New York, NY, USA, 1993. pp 117-309.

[16] Hindley J. Roger, "Basic Simple Type Theory". *Cambridge University Press*, New York, NY, USA, 1997. pp. 185.

[17] Pfenning F. and Davies R., "A judgmental reconstruction of modal logic", *Mathematical Structures in Computer Science*, vol. 11, no 4, 2001, pp. 511-540.

[18] H.P. Barendregt. The Lambda Calculus — Its Syntax and Semantics. Studies in Logic and the Foundations of Mathematics, vol. 103. Amsterdam: North-Holland, 1985.

[19] Yoshihiko KAKUTANI, A Curry-Howard Correspondence for Intuitionistic Normal Modal Logic, Computer Software, Released February 29, 2008, Online ISSN , Print ISSN 0289-6540.

[20] Kakutani Y. (2007) Call-by-Name and Call-by-Value in Normal Modal Logic. In: Shao Z. (eds) Programming Languages and Systems. APLAS 2007. Lecture Notes in Computer Science, vol 4807. Springer, Berlin, Heidelberg

[21] T. Abe. Completeness of modal proofs in
first-order predicate logic. Computer Software, JSSST Journal, 24:165 – 177, 2007.

[22] Lambek, J. and Scott P.J. (1986) Introduction to Higher Order Categorical Logic, Cambridge Studies in Advanced Mathematics 7, Cambridge: Cambridge University Press.

[23] Samuel Eilenberg and Max Kelly, Closed categories. Proc. Conf. Categorical Algebra (La Jolla, Calif., 1965).

[24] Samson Abramsky and Nikos Tzevelekos, Introduction to Categories and Categorical Logic