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 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.

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 (kind $*$) and returns another simple type of kind $*$. For more detailed description of the type system with kinds used in Haskell see [12].

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.

We investigate the relationship between intuitionistic epistemic logic IEL$^-$ and applicative programming with side-effects by constructing 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 and we prove that obtained type system is sound and complete for applicative functor on cartesian closed category (using the categorical definition proposed by Paterson [26]).

$\lambda_{\mathbf{K}}$ 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 a proof-theoretical view at this kind of computations in functional programming and prove strong normalization and confluence.

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

The first is to define the natural deduction calculus for NIEL$^-$:

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

$$\frac{\Gamma \vdash A}{\Gamma \vdash \mathbf{K}A}\,\mathbf{K}_I \qquad\qquad \frac{\Gamma \vdash \mathbf{K}A_1, \ldots, \Gamma \vdash \mathbf{K}A_n \qquad A_1, \ldots, A_n \vdash B}{\Gamma \vdash \mathbf{K}B}$$

The first rule allows to derive co-reflextion. The second modal rule is a counterpart of $\Box_I$ rule in natural deduction calculus for constructive K (see [25]).

We will denote $\Gamma \vdash \mathbf{K}A_1, \ldots, \Gamma \vdash \mathbf{K}A_n$ and $A_1, \ldots, A_n \vdash B$ as $\Gamma \vdash \mathbf{K}\vec{A}$ and $\vec{A} \vdash B$ for brevity.

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

Proof. Induction on the derivation.

Let us consider cases with modality.

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

(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 $\Gamma \vdash_{\mathrm{NIEL}^-} \mathbf{K}\vec{A}$ and $\vec{A} \vdash B$, then $\mathrm{IEL}^- \vdash \bigwedge \Gamma \to \mathbf{K}B$.

(1)	$\bigwedge \Gamma \to \bigwedge\limits_{i=1}^{n} \mathbf{K}A_i$	assumption
(2)	$\bigwedge\limits_{i=1}^{n} \mathbf{K}A_i \to \mathbf{K} \bigwedge\limits_{i=1}^{n} A_i$	IEL^- theorem
(3)	$\bigwedge \Gamma \to \mathbf{K} \bigwedge\limits_{i=1}^{n} A_i$	from (1), (2) and transitivity
(4)	$\bigwedge\limits_{i=1}^{n} A_i \to B$	assumption
(5)	$(\bigwedge\limits_{i=1}^{n} A_i \to B) \to \mathbf{K}(\bigwedge\limits_{i=1}^{n} A_i \to B)$	co-reflection
(6)	$\mathbf{K}(\bigwedge\limits_{i=1}^{n} A_i \to B)$	from (4), (5) and MP
(7)	$\mathbf{K} \bigwedge\limits_{i=1}^{n} A_i \to \mathbf{K}B$	from (6) and normality
(8)	$\bigwedge \Gamma \to \mathbf{K}B$	from (3), (7) and transitivity

$\square$

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

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

At the next step we build the typed lambda-calculus based on the NIEL^- with implication and dijunction by proof-assingment in rules. Obtained fragment is equivalent to IEL^- without axioms for negation and disjunction.

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:

$$\Lambda_{\mathbf{K}} ::= \mathbb{V} \mid (\lambda \mathbb{V}.\Lambda_{\mathbf{K}}) \mid (\Lambda_{\mathbf{K}}\Lambda_{\mathbf{K}}) \mid (\Lambda_{\mathbf{K}}, \Lambda_{\mathbf{K}}) \mid (\pi_1 \Lambda_{\mathbf{K}}) \mid (\pi_2 \Lambda_{\mathbf{K}}) \mid (\mathbf{pure}\ \Lambda_{\mathbf{K}}) \mid (\mathbf{let\ pure}\ \mathbb{V}^* = \Lambda_{\mathbf{K}}^* \ \mathbf{in}\ \Lambda_{\mathbf{K}})$$

Where $\mathbb{V}^*$ and $\Lambda_{\mathbf{K}}^*$ denote the set of finite sequences of variables $\bigcup\limits_{i=0}^{\infty} \mathbb{V}^i$ and the set of finite sequences of terms $\bigcup\limits_{i=0}^{\infty} {\Lambda_{\mathbf{K}}}^i$. Note that the sequence of variables $\vec{x}$ and the sequence of terms $\vec{M}$ should have the same length. Otherwise, term is not well-formed.

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:

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

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* $NIEL^{-}_{\wedge,\to}$*:*

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

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

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

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

$\mathbf{K}_I$-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 **IK**, which is described in [19].

$\Gamma \vdash \vec{M} : \mathbf{K}\vec{A}$ is a syntax sugar for the sequence $\Gamma \vdash M_1 : \mathbf{K}A_1, \dots, \Gamma \vdash M_n : \mathbf{K}A_n$ and $\vec{x} : \vec{A} \vdash N : B$ is a short form for $x_1 : A_1, \dots, x_n : A_n \vdash N : B$. **let pure** $\vec{x} = \vec{M}$ **in** N is a simultaneous local binding in N. We use this short form instead of **let pure** $x_1, \dots, x_n = M_1, \dots, M_n$ **in** N.

In fact, our calculus is the extention of typed lambda calculus for **IK** by $\mathbf{K}_I$-rule that is appropriate to co-reflection.

Here are some examples of closed terms:

$$\dfrac{\dfrac{x : A \vdash x : A}{x : A \vdash \mathbf{pure}\, x : \mathbf{K}A}}{\vdash (\lambda x.\mathbf{pure}\, x) : A \to \mathbf{K}A}$$

$$\dfrac{\dfrac{\dfrac{f : \mathbf{K}(A \to B) \vdash f : \mathbf{K}(A \to B) \qquad x : \mathbf{K}A \vdash x : \mathbf{K}A \qquad \dfrac{g : A \to B \vdash g : A \to B \qquad y : A \vdash y : A}{g : A \to B, y : A \vdash gy : B} \to_e}{f : \mathbf{K}(A \to B), x : \mathbf{K}A \vdash \mathbf{let\ pure}\, g, y = f, x\ \mathbf{in}\ gy : \mathbf{K}B}\ \mathrm{let}_{\mathbf{K}}}{f : \mathbf{K}(A \to B) \vdash \lambda x.\mathbf{let\ pure}\, g, y = f, x\ \mathbf{in}\ gy : \mathbf{K}A \to \mathbf{K}B}}{\vdash \lambda f.\lambda x.\mathbf{let\ pure}\, g, y = f, x\ \mathbf{in}\ gy : \mathbf{K}(A \to B) \to \mathbf{K}A \to \mathbf{K}B}$$

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

Definition 6. *The set $FV(M)$ of free variables for a term M:*

1) $FV(x) = \{x\}$;
2) $FV(\lambda x.M) = FV(M)\backslash\{x\}$;
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) $FV(\textbf{let pure}\ \vec{x} = \vec{M}\ \textbf{ in }\ N) = \bigcup_{i=1}^{n} FV(M)$, 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) $(\lambda x.M)[x := N] = \lambda x.M[y := N]$, $y \in FV(M)$;
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) $(\textbf{let pure}\ \vec{x} = \vec{M}\ \textbf{in}\ N)[y := P] = \textbf{let pure}\ \vec{x} = (\vec{M}[y := P])\ \textbf{in}\ N$.

Substitution and free variable for terms of kind **let pure** are defined similary to [19].

Definition 8. *Type substituition*

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

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) $$\begin{array}{l}\textbf{let pure}\ \vec{x}, y, \vec{z} = \vec{M}, \textbf{let pure}\ \vec{w} = \vec{N}\ \textbf{in}\ Q, \vec{P}\ in\ R \to_\beta \\ \textbf{let pure}\ \vec{x}, \vec{w}, \vec{z} = \vec{M}, \vec{N}, \vec{P}\ \textbf{in}\ R[y := Q]\end{array}$$
5) $\textbf{let pure}\ \vec{x} = \textbf{pure}\ \vec{M}\ \textbf{in}\ N \to_\beta \textbf{pure}\ N[\vec{x} := \vec{M}]$
6) $\textbf{let pure}\ \_\_ = \_\_\ \textbf{in}\ M \to_\beta \textbf{pure}\ M$, where $\_\_$ is an empty sequence of terms.
7) $\lambda x.fx \to_\eta f$;
8) $\langle \pi_1 P, \pi_2 P\rangle \to_\eta P$;
9) $\textbf{let pure}\ x = M\ \textbf{in}\ x \to_\eta M$;

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

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

Lemma 3. *Generation for $\mathbf{K}_I$.*

Let $\Gamma \vdash \textbf{pure}\ M : \mathbf{K}A$, then $\Gamma \vdash M : A$;

[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]. We will consider only modal cases

Proof. Straightforwardly. $\square$

Lemma 4. *Basic lemmas.*

- *If* $\Gamma \vdash M : A$ *and* $\Gamma \subseteq \Delta$, *then* $\Delta \vdash M : A$;
- *If* $\Gamma \vdash M : A$, *then* $\Delta \vdash M : A$, *where* $\Delta = \{x_i : A_i \mid (x_i : A_i) \in \Gamma \;\&\; x_i \in FV(M)\}$
- *If* $\Gamma, x : A \vdash M : B$ *and* $\Gamma \vdash N : A$, *then* $\Gamma \vdash M[x := N] : B$.
- *If* $\Gamma \vdash M : A$, *then* $\Gamma[B := C] \vdash M : (A[B := C])$.

Proof.

1) The derivation ends in

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

By IH $\Delta \vdash \vec{M} : \mathbf{K}\vec{A}$, so $\Delta \vdash \mathbf{let\ pure}\, \vec{x} = \vec{M}\ \mathbf{in}\ N : \mathbf{K}B$.

2) Similary.

3) The derivation ends in

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

Let $\Gamma \vdash P : C$.
By IH, $\Gamma \vdash \vec{M}[z := P] : \mathbf{K}\vec{A}$. So $\Gamma \vdash \mathbf{let\ pure}\, \vec{x} = \vec{M}[z := P]\ \mathbf{in}\ N : \mathbf{K}B$

4-5) Similary.

$\square$

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

Proof. Induction on the derivation $\Gamma \vdash M : A$ and on the generation of $\rightarrow_{\beta\eta}$.
For cases with application, abstraction and pairs see [12] [13].
1) If $\Gamma \vdash \mathbf{let\ pure}\, \vec{x}, y, \vec{z} = \vec{M}, \mathbf{let\ pure}\, \vec{w} = \vec{N}\ \mathbf{in}\ Q, \vec{P}\ \mathbf{in}\ R : \mathbf{K}B$, then $\Gamma \vdash \mathbf{let\ pure}\, \vec{x}, \vec{w}, \vec{z} = \vec{M}, \vec{N}, \vec{P}\ \mathbf{in}\ R[y := Q] : \mathbf{K}B$ by rule 4).
2) Let $\Gamma \vdash \mathbf{let\ pure}\, x = M\ \mathbf{in}\ x : \mathbf{K}A$, then $\Gamma \vdash M : \mathbf{K}A$ by rule 9).
See [19].
3) The derivation ends in

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

So $\Gamma \vdash \vec{M} : \vec{A}$ by Lemma 4 and $\Gamma \vdash N[\vec{x} := \vec{M}] : B$ by Lemma 4, part 3.
Then we can transform this into the following derivation:

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

4) The derivation ends in

$$\frac{\vdash M : A}{\vdash \textbf{let pure}\, \_\_ = \_\_\ \textbf{in}\, M : \mathbf{K}A}$$

So, if $\vdash M : A$, then $\vdash \textbf{pure}\, M : \mathbf{K}A$.
Note that this part of the lemma works conversly too. □

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

Proof.
We modify and apply Tait's technique of logical relation for modalities. See [13] [**?**].
We treat only modal cases below.

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

- $SC_A = \{M : A \mid M$ *is strongly normalizing*$\}$ *for* $A \in \mathbb{T}$*;*
- $SC_{A \to B} = \{M : A \to B \mid \forall N \in SC_A, MN \in SC_B\}$*, for* $A, B \in \mathbb{T}_\mathbf{K}$ *for* $A, B \in \mathbb{T}_\mathbf{K}$*;*
- $SC_{\mathbf{K}A} = \{M : \mathbf{K}A \mid M$ *is strongly normalizing*$\}$ *for* $A \in \mathbb{T}$*;*
- $\forall i \in \{1, \dots, n\}, \prod\limits_{i=1}^{n} SC_{\mathbf{K}A_i} = \{\vec{M} = (M_1, \dots, M_n) \mid \forall N \in SC_B, FV(N) = \{x_1, \dots, x_n\}\ \&\ \forall i, x_i \in SC_{A_i} \Rightarrow \textbf{let pure}\, \vec{x} = \vec{M}\ \textbf{in}\, N \in SC_{\mathbf{K}B}\}$

Definition 11. *A term M is neutral, if it has one of the following forms:*

- MN*;*
- *If M is neutral, then* **pure** M *is neutral;*
- *If $\vec{M}$ and N are neutral, then* **let pure** $\vec{x} = \vec{M}$ **in N** *is neutral. $\vec{x}$ is a sequence of free variables of a term N.*

Lemma 5.

- *If $M \in SC_A$ and $A \in \mathbb{T}_\mathbf{K}$, then M is strongly normalizing;*
- *If $M \in SC_A$, $A \in \mathbb{T}_\mathbf{K}$ and $M \to_\beta N$, then $N \in SC_A$;*
- *If N is neutral, $N \in SC_A$. Then, if $M \to_\beta N$, 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-ii-iii) Immediately.

2)

i) Suppose $\vec{M} = (M_1, \ldots, M_n) \in \prod_{i=1}^{n} SC_{\mathbf{K}A_i}$.

Let $N \in SC_B$, such that $FV(N) = \{x_1, \ldots, x_n\}$ and $\forall i, x_i \in SC_{A_i}$.

So **let pure** $\vec{x} = \vec{M}$ **in** $N \in SC_{\mathbf{K}B}$ by IH.

So $\vec{M}$ are strongly normalizing, since **let pure** $\vec{x} = \vec{M}$ **in** N is strongly normalizing by IH.

ii) Let $\vec{M_1} \in \prod_{i=1}^{n} SC_{\mathbf{K}A_i}$ and $\vec{M_1} \to_\beta \vec{M_2}$.

Let $N \in SC_B$, such that $FV(N) = \{x_1, \ldots, x_n\}$ and $\forall i, x_i \in SC_{A_i}$.

So **let pure** $\vec{x} = \vec{M_1}$ **in** $N \to_\beta$ **let pure** $\vec{x} = \vec{M_2}$ **in** N

and **let pure** $\vec{x} = \vec{M_2}$ **in** $N \in SC_{\mathbf{K}B}$ by assumption.

So $\vec{M_2} \in \prod_{i=1}^{n} SC_{\mathbf{K}A_i}$.

iii) Let M_2 be neutral, $M_2 \in \prod_{i=1}^{n} SC_{\mathbf{K}A_i}$ and $M_1 \to_\beta M_2$.

Let $N \in SC_B$, , such that $FV(N) = \{x_1, \ldots, x_n\}$ and $\forall i, x_i \in SC_{A_i}$.

So **let pure** $\vec{x} = \vec{M_2}$ **in** $N \in SC_{\mathbf{K}B}$.

Thus **let pure** $\vec{x} = \vec{M_1}$ **in** $N \to_\beta$ **let pure** $\vec{x} = \vec{M_2}$ **in** $\in N$.

Hence **let pure** $\vec{x} = \vec{M_1}$ **in** $N \in SC_{\mathbf{K}B}$ by IH, so $\vec{M_1} \in \prod_{i=1}^{n} SC_{\mathbf{K}A_i}$. $\square$

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.
Induction on the derivation of $x_1 : A_1, \ldots, x_n : A_n \vdash M : A$.

1) 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) 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 grammar without constructors for pairs.

Definition 12. *The set of underlined terms.*

- $x \in \mathbb{V} \Rightarrow x \in \underline{\Lambda}$;
- $M \in \underline{\Lambda} \Rightarrow (\lambda x.M) \in \underline{\Lambda}$;
- $M, N \in \underline{\Lambda} \Rightarrow (MN) \in \underline{\Lambda}$;
- $M \in \underline{\Lambda} \Rightarrow (\mathbf{pure}\, M) \in \underline{\Lambda}$;
- $\vec{x} \in \mathbb{V}^*, \vec{M} \in \underline{\Lambda}^*, N \in \underline{\Lambda} \Rightarrow \mathbf{let\ pure}\, \vec{x} = \vec{M}\ \mathbf{in}\ N \in \underline{\Lambda}$, *where* $length(\vec{x}) = length(\vec{M})$;
- $M, N \in \underline{\Lambda} \Rightarrow (\lambda_i x.M)N \in \underline{\Lambda}$, *for all* $i \in \mathbb{N}$.

Definition 13. *Subsitution for term with labelled lambda:*
$((\lambda_i x.M)N)[y := Z] = (\lambda_i x.M[y := Z])(N[y := Z])$

Definition 14. *Index erasing*
Let us define map $|.| : \underline{\Lambda} \to \Lambda$ *as follows:*

- $|x| = x$;
- $|\lambda x.M| = \lambda x.|M|$;
- $|MN| = |M||N|$;
- $|\mathbf{pure}\, M| = \mathbf{pure}\, |M|$;
- $|\mathbf{let\ pure}\, \vec{x} = \vec{M}\ \mathbf{in}\ N| = \mathbf{let\ pure}\, \vec{x} = |\vec{M}|\ \mathbf{in}\ |N|$;
- $|(\lambda_i x.M)N| = (\lambda x.|M|)|N|$

Definition 15. *Reduction rules:*

- $(\lambda x.M)N \to_{\underline{\beta}} M[x := N]$;
- $\begin{array}{l} \mathbf{let\ pure}\, \vec{x}, y, \vec{z} = \vec{M}, \mathbf{let\ pure}\, \vec{w} = \vec{N}\ \mathbf{in}\ Q, \vec{P}\ in\ R \to_{\underline{\beta}} \\ \mathbf{let\ pure}\, \vec{x}, \vec{w}, \vec{z} = \vec{M}, \vec{N}, \vec{P}\ \mathbf{in}\ R[y := Q] \end{array}$;
- $\mathbf{let\ pure}\, \vec{x} = \mathbf{pure}\, \vec{M}\ \mathbf{in}\ N \to_{\underline{\beta}} \mathbf{pure}\, N[\vec{x} := \vec{M}]$;
- $\mathbf{let\ pure}\, \_\_ = \_\_\ \mathbf{in}\ M \to_{\underline{\beta}} \mathbf{pure}\, M$
- $(\lambda x_i.M)N \to_{\underline{\beta}} M[x := N]$

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

Definition 16. *Indexed redex erasing:*
Let us define the next map $\phi : \underline{\Lambda} \to \Lambda$*:*

- $\phi(x) = x$;
- $\phi(\lambda x.M) = \lambda x.\phi(M)$;
- $\phi(MN) = \phi(M)\phi(N)$;
- $\phi(\mathbf{pure}\, M) = \mathbf{pure}\, \phi(M)$;
- $\phi(\mathbf{let\ pure}\, \vec{x} = \vec{M}\ \mathbf{in}\ N) = \mathbf{let\ pure}\, \vec{x} = \phi(\vec{M})\ \mathbf{in}\ \phi(N)$;
- $\phi((\lambda_i x.M)N) = \phi(M)[x := \phi(N)]$

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. $\square$

Lemma 11. *Strip lemma.*

If $M \rightarrow_\beta N$ 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.

$$\begin{tikzcd}
M \arrow[rr, two heads] \arrow[dd, "\beta"'] & & P \arrow[dd, dashed, "\beta"] & \\
& N' \arrow[ul, "|\cdot|"] \arrow[dl, "\phi"] \arrow[rr, dashed, two heads, "\underline{\beta}"] & & P' \arrow[ul, "|\cdot|"'] \arrow[dl, "\phi"] \\
N \arrow[rr, dashed, two heads, "\beta"'] & & Q &
\end{tikzcd}$$

$\square$

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. $\square$

$\square$

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 $\mathbf{let\ pure}\ \vec{x} = \vec{M}\ \mathbf{in}\ N$ was considered by Kakutani [19] [20].

If **pure** M is in normal form, so M is in normal form and its subterms are in normal form too by hypothesis.

Thus if **pure** M is in normal form, then all its subterms are in normal form too. $\square$

3 Categorical semantics

Definition 17. *Monoidal category*

Monoidal category is a category $\mathcal{C}$ with additional monoidal structure:

- *A bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \rightarrow C$ called tensor product;*
- *Identity object $\mathbb{1}$;*
- *A natural isomorphism called associator: $\alpha_{A,B,C} : (A \otimes B) \otimes C \cong A \otimes (B \otimes C)$;*

- *A natural isomorphism called left unitor:* $L_A : \mathbb{1} \otimes A \cong A$*;*
- *A natural isomorphism called right unitor* $R_A : A \otimes \mathbb{1} \cong A$*;*
- *A coherence condition called MacLane pentagon, i.e. the following diagram commutes:*

$$\begin{tikzcd}
 & (A \otimes (B \otimes C)) \otimes D \arrow[dr, "\alpha_{A,B\otimes C,D}"] & \\
((A \otimes B) \otimes C) \otimes D \arrow[ur, "\alpha_{A,B,C}\otimes id_D"] \arrow[d, "\alpha_{A\otimes B,C,D}"'] & & A \otimes ((B \otimes C) \otimes D) \arrow[d, "id_A \otimes \alpha_{B,C,D}"] \\
(A \otimes B) \otimes (C \otimes D) \arrow[rr, "\alpha_{A,B,C\otimes D}"'] & & A \otimes (B \otimes (C \otimes D))
\end{tikzcd}$$

- *A coherence condition called triangle identity:*

$$\begin{tikzcd}
(A \otimes \mathbb{1}) \otimes B \arrow[rr, "\alpha_{A,\mathbb{1},B}"] \arrow[dr, "R_A \otimes id_B"'] & & A \otimes (\mathbb{1} \otimes B) \arrow[dl, "id_A \otimes L_B"] \\
 & A \otimes B &
\end{tikzcd}$$

- *A monoidal category* $\mathcal{C}$ *is called symmetric if forall* $A, B \in Ob_{\mathcal{C}}$*, there is an isomorphism* $\sigma_{A,B} : A \otimes B \cong B \otimes A$*.*

Definition 18. *Cartesian closed category*
Cartesian closed category is a category with a terminal object, finite products and exponentiation.

Note that, any cartesian closed category is the special case of a (symmetric) monoidal category, where tensor is a product and identity object is a terminal object.

Definition 19. *Lax monoidal functor*
Let $\langle \mathcal{C}, \otimes_1, \mathbb{1}_{\mathcal{C}} \rangle$ *and* $\langle \mathcal{D}, \otimes_2, \mathbb{1}_{\mathcal{D}} \rangle$ *are monoidal categories.*
A lax 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:*

- $u : \mathbb{1}_{\mathcal{D}} \to \mathcal{F}\mathbb{1}_{\mathcal{C}}$*;*
- $*_{A,B} : \mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}B \to \mathcal{F}(A \otimes_{\mathcal{C}} B)$*.*

and coherence maps:

- *Associativity:*

$$\begin{tikzcd}
(\mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}B) \otimes_{\mathcal{D}} \mathcal{F}C \arrow[r, "\alpha^{\mathcal{D}}_{\mathcal{F}A,\mathcal{F}B,\mathcal{F}C}"] \arrow[d, "*_{A,B} \otimes_{\mathcal{D}} id_{\mathcal{F}B}"'] & \mathcal{F}A \otimes_{\mathcal{D}} (\mathcal{F}B \otimes_{\mathcal{D}} \mathcal{F}C) \arrow[d, "id_{\mathcal{F}A} \otimes_{\mathcal{D}} *_{B,C}"] \\
\mathcal{F}(A \otimes_{\mathcal{C}} B) \otimes_{\mathcal{D}} \mathcal{C} \arrow[d, "*_{A \otimes_{\mathcal{C}} B, C}"'] & \mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}(B \otimes_{\mathcal{C}} C) \arrow[d, "*_{A, B \otimes_{\mathcal{C}} C}"] \\
\mathcal{F}((A \otimes_{\mathcal{C}} B) \otimes_{\mathcal{C}} C) \arrow[r, "\mathcal{F}(\alpha^{\mathcal{C}}_{A,B,C})"'] & \mathcal{F}(A \otimes_{\mathcal{C}} (B \otimes_{\mathcal{C}} C))
\end{tikzcd}$$

- *Left unitality:*

$$\begin{tikzcd}
\mathbb{1}_{\mathcal{D}} \otimes_{\mathcal{D}} \mathcal{F}A \arrow[r, "u \otimes_{\mathcal{D}} id_{\mathcal{F}A}"] \arrow[d, "L^{\mathcal{D}}_{\mathcal{F}A}"'] & \mathcal{F}\mathbb{1}_{\mathcal{C}} \otimes_{\mathcal{D}} \mathcal{F}A \arrow[d, "*_{\mathbb{1}_{\mathcal{C}},A}"] \\
\mathcal{F}A & \mathcal{F}(\mathbb{1}_{\mathcal{C}} \otimes_{\mathcal{C}} A) \arrow[l, "\mathcal{F}(L^{\mathcal{C}}_A)"']
\end{tikzcd}$$

- *Right unitality:*

$$\begin{tikzcd}
\mathcal{F}A \otimes_{\mathcal{D}} \mathbb{1}_{\mathcal{D}} \arrow[r, "id_{\mathcal{F}A} \otimes_{\mathcal{D}} u"] \arrow[d, "R^{\mathcal{D}}_{\mathcal{F}A}"'] & \mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}\mathbb{1}_{\mathcal{C}} \arrow[d, "*_{A,\mathbb{1}_{\mathcal{C}}}"] \\
\mathcal{F}A & \mathcal{F}(A \otimes_{\mathcal{C}} \mathbb{1}_{\mathcal{C}}) \arrow[l, "\mathcal{F}(R^{\mathcal{C}}_A)"]
\end{tikzcd}$$

Definition 20. *Strong functor on a monoidal category is an endofunctor equipped with a natural transformation so called tensorial strength:*

$$\tau_{A,B} : A \otimes \mathcal{K}B \to \mathcal{K}(A \otimes B)$$

making two diagrams commute:

$$\begin{tikzcd}
(A \otimes B) \otimes \mathcal{K}C \arrow[rr, "\tau_{A\otimes B,C}"] \arrow[d, "\alpha_{A,B,\mathcal{K}C}"'] & & \mathcal{K}((A \otimes B) \otimes C) \arrow[d, "\mathcal{K}(\alpha_{A,B,C})"] \\
A \otimes (B \otimes \mathcal{K}C) \arrow[r, "id_A \otimes \tau_{B,C}"'] & A \otimes \mathcal{K}(B \otimes C) \arrow[r, "\tau_{A,(B\otimes C)}"'] & \mathcal{K}(A \otimes (B \otimes C))
\end{tikzcd}$$

$$\begin{tikzcd}
\mathbb{1} \otimes \mathcal{K}A \arrow[r, "\mu_{\mathbb{1},A}"] \arrow[dr, "R_{\mathcal{K}A}"'] & \mathcal{K}(\mathbb{1} \otimes A) \arrow[d, "\mathcal{K}(R_A)"] \\
& \mathcal{K}A
\end{tikzcd}$$

Definition 21. *Applicative functor*

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

- $u = \eta_{\mathbb{1}}$*;*
- $*_{A,B} \circ (\eta_A \otimes \eta_B) = \eta_{A\otimes B}$*, i.e. the following diagram commutes:*

$$\begin{tikzcd}
A \otimes B \arrow[r, "\eta_A \otimes \eta_B"] \arrow[dr, "\eta_{A\otimes B}"'] & \mathcal{K}A \otimes \mathcal{K}B \arrow[d, "*_{A,B}"] \\
& \mathcal{K}(A \otimes B)
\end{tikzcd}$$

- $\tau_{A,B} = *_{A,B} \circ \eta_A \otimes id_{\mathcal{K}B}$*;*
- $\tau_{A,B} \circ id_A \otimes \eta_B = \eta_{A\otimes B}$*, i.e. the following diagram commutes:*

$$\begin{tikzcd}
A \otimes B \arrow[r, "id_A \otimes \eta_B"] \arrow[dr, "\eta_{A\otimes B}"'] & A \otimes \mathcal{K}B \arrow[d, "\tau_{A,B}"] \\
& \mathcal{K}(A \otimes B)
\end{tikzcd}$$

By default we will consider some strong lax monoidal functor on cartesian closed category.

Note that lax monoidal functor on a cartesian closed category is a special case of an applicative functor [26], but we will define η explicitly for simplicity.

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

- *Interpretation for types:*
  - $[\![A]\!] := \hat{A}, A \in \mathbb{T}$*, where* $\hat{A}$ *is an object of* $\mathcal{C}$ *obtained by some given assignment;*
  - $[\![A \to B]\!] := [\![B]\!]^{[\![A]\!]}$*;*
  - $[\![A \times B]\!] := [\![A]\!] \times [\![B]\!]$.
- *Interpretation for modal types:*
  - $[\![\mathbf{K}A]\!] = \mathcal{K}[\![A]\!]$*;*
- *Interpretaion for contexts:*
  - $[\![\quad]\!] = \mathbb{1}$*, where* $\mathbb{1}$ *is a terminal object of a given ccc;*
  - $[\![\Gamma, x : A]\!] = [\![\Gamma]\!] \times [\![A]\!]$
- *Interpretation for typing assignment:*
  - $[\![\Gamma \vdash M : A]\!] := [\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$.
- *Interpretation for typing rules:*

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

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

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

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

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

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

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

Definition 23. *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:

- $x_i[\vec{x} := \vec{M}] = M_i$;
- $(\lambda x.M)[\vec{x} := \vec{M}] = \lambda x.(M[\vec{x} := \vec{M}])$;
- $(MN)[\vec{x} := \vec{M}] = (M[\vec{x} = \vec{M}])(N[\vec{x} := \vec{M}])$;
- $\langle M, N \rangle = \langle (M[\vec{x} = \vec{M}]), (N[\vec{x} := \vec{M}]) \rangle$;
- $(\pi_i P)[\vec{x} := \vec{M}] = \pi_i(P[\vec{x} = \vec{M}])$;
- $(\mathbf{pure}\, M)[\vec{x} := \vec{M}] = \mathbf{pure}\, (M[\vec{x} = \vec{M}])$;
- $(\mathbf{let\ pure}\, \vec{x} = \vec{M}\ \mathbf{in}\ N)[\vec{y} := \vec{P}] = \mathbf{let\ pure}\, \vec{x} = (\vec{M}[\vec{y} := \vec{P}])\ \mathbf{in}\ N$

Lemma 12.

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

Proof.

1)

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

2)

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

□

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

2) $[\![\vdash \mathbf{let\ pure}\ \_\_ = \_\_\ \mathbf{in}\ M : \mathbf{K}A]\!] = [\![\vdash \mathbf{pure}\ M : \mathbf{K}A]\!]$
$[\![\vdash \mathbf{let\ pure}\ \_\_ = \_\_\ \mathbf{in}\ M : \mathbf{K}A]\!] =$
By interpretation
$\mathcal{K}([\![M]\!]) \circ u_{\mathbb{1}} =$
By the definition of an applicative functor
$\mathcal{K}([\![M]\!]) \circ \eta_{\mathbb{1}} =$ □
By naturality of η
$\eta_{[\![A]\!]} \circ [\![M]\!] =$
By interpretation
$[\![\vdash \mathbf{pure}\ M : \mathbf{K}A]\!]$

□

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

Proof.

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

Definition 24. *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 indices below).

Definition 25. *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 26. *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. ∎

Definition 27. *Let us define natural transformations:*

- $\eta : Id \Rightarrow \mathcal{K}$, *s. t.* $\forall \hat{A} \in Ob_{\mathcal{C}(\lambda)}$, $\eta_{\hat{A}} = [x, \mathbf{pure}\ x] \in Hom_{\mathcal{C}(\lambda)}(\hat{A}, \mathbf{K}\hat{A})$;

- $*_{A,B} : \mathbf{K}\hat{A} \times \mathbf{K}\hat{B} \to \mathbf{K}(\hat{A} \times \hat{B})$, *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 let$_{\mathbf{K}}$-rule:

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

Lemma 15.
$\mathbf{K}$ *is a lax monoidal endofunctor*

Proof.
See [19] ∎

Lemma 16. *Naturality and coherence for* η*:*

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

Proof.

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
$[z, \mathbf{let\ pure}\, x = z\ \mathbf{in}\ M] \circ [x, \mathbf{pure}\, \mathbf{x}] =$
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
$[x, \mathbf{pure}\, M]$

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
$\eta_{\hat{A}\times\hat{B}}$

□

Definition 28.
$u_{\mathbb{1}} = [\blacksquare, \mathbf{let\ pure}\, \_\_ = \_\_\ \mathbf{in}\ \blacksquare] \in Hom_{\mathcal{C}(\lambda)}(\mathbb{1}, \mathbf{K}\mathbb{1})$.

Lemma 17.
$u_{\mathbb{1}} = \eta_{\mathbb{1}}$

Proof. Immediately. $\square$

Tensorial strength is defined as follows:

Definition 29. *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:

$*_{\hat{A},\hat{B}} \circ [p, \langle \mathbf{pure}\,(\pi_1 p), \pi_2 p\rangle] =$
By definition
$[p', \mathbf{let\ pure}\, x, y = \pi_1 p', \pi_2 p'\ \mathbf{in}\, \langle x, y\rangle] \circ [p, \langle \mathbf{pure}\,(\pi_1 p), \pi_2 p\rangle] =$
By composition
$[p, \mathbf{let\ pure}\, x, y = \pi_1 p', \pi_2 p'\ \mathbf{in}\, \langle x, y\rangle[p' := \langle \mathbf{pure}\,(\pi_1 p), \pi_2 p\rangle]] =$
By substitution
$[p, \mathbf{let\ pure}\, x, y = \pi_1(\langle \mathbf{pure}\,(\pi_1 p), \pi_2 p\rangle), \pi_2(\langle \pi_1 p, \mathbf{pure}\,(\pi_2 p)\rangle)\ \mathbf{in}\, \langle x, y\rangle] =$
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 18. *Coherence for tensorial strength*

- *fmap* $\alpha_{\hat{A},\hat{B},\hat{C}} \circ \tau_{\hat{A}\times\hat{B},\hat{C}} = \tau_{\hat{A},\hat{B}\times\hat{C}} \circ (id_{\hat{A}} \times \tau_{\hat{B},\hat{C}}) \circ \alpha_{\hat{A},\hat{B},\mathbf{K}\hat{C}}$;
- $\mathbf{K}(R_{\hat{A}}) \circ \tau_{\mathbb{1},\hat{A}} = R_{\mathbf{K}\hat{A}}$.

where $\alpha_{\hat{A},\hat{B},\hat{C}} = [p, \langle \pi_1(\pi_1 p), \langle \pi_2(\pi_1 p), \pi_2 p\rangle\rangle]$ *and* $R = \pi_2$.

Proof. The second part is obvious. We will consider only the first statement.
1) Let us define $\tau_{\hat{A}\times\hat{B},\hat{C}}$ as follows:
$\tau_{\hat{A}\times\hat{B},\hat{C}} = [p,\ \mathbf{let\ pure}\, x, y, z = \mathbf{pure}\,(\pi_1(\pi_1 p)), \mathbf{pure}\,(\pi_2(\pi_1 p)), \pi_2 p\ \mathbf{in}\, \langle\langle x, y\rangle, z\rangle]$

Then:
fmap $\alpha_{\hat{A},\hat{B},\hat{C}} \circ \tau_{\hat{A}\times\hat{B},\hat{C}} =$
$[q, \mathbf{let\ pure}\, r = q\ \mathbf{in}\, \langle \pi_1(\pi_1 r), \langle \pi_2(\pi_1 r), \pi_2 r\rangle\rangle] \circ$
$\circ\, [p,\ \mathbf{let\ pure}\, x, y, z = \mathbf{pure}\,(\pi_1(\pi_1 p)), \mathbf{pure}\,(\pi_2(\pi_1 p)), \pi_2 p\ \mathbf{in}\, \langle\langle x, y\rangle, z\rangle] =$
Composition
$[p, \mathbf{let\ pure}\, r = q\ \mathbf{in}\, \langle \pi_1(\pi_1 r), \langle \pi_2(\pi_1 r), \pi_2 r\rangle\rangle$
$[q :=\ \mathbf{let\ pure}\, x, y, z = \mathbf{pure}\,(\pi_1(\pi_1 p)), \mathbf{pure}\,(\pi_2(\pi_1 p)), \pi_2 p\ \mathbf{in}\, \langle\langle x, y\rangle, z\rangle]] =$
Substitution and reduction
$[p, \mathbf{let\ pure}\, r = (\mathbf{let\ pure}\, x, y, z = \mathbf{pure}\,(\pi_1(\pi_1 p)), \mathbf{pure}\,(\pi_2(\pi_1 p)), \pi_2 p\ \mathbf{in}\, \langle\langle x, y\rangle, z\rangle)$
$\mathbf{in}\, \langle \pi_1(\pi_1 r), \langle \pi_2(\pi_1 r), \pi_2 r\rangle\rangle] =$
β-reduction rule
$[p, \mathbf{let\ pure}\, x, y, z = \mathbf{pure}\,(\pi_1(\pi_1 p)), \mathbf{pure}\,(\pi_2(\pi_1 p)), \pi_2 p\ \mathbf{in}\, \langle \pi_1(\pi_1 r), \langle \pi_2(\pi_1 r), \pi_2 r\rangle\rangle$
$[r := \langle\langle x, y\rangle, z\rangle]] =$
β-reduction rule
$[p, \mathbf{let\ pure}\, x, y, z = \mathbf{pure}\,(\pi_1(\pi_1 p)), \mathbf{pure}\,(\pi_2(\pi_1 p)), \pi_2 p\ \mathbf{in}\, \langle x, \langle y, z\rangle\rangle]$

On the other hand:
Let us define as follows $\tau_{\hat{A},\hat{B}\times\hat{C}}$:
$\tau_{\hat{A},\hat{B}\times\hat{C}} = [r, \mathbf{let\ pure}\, x, y, z = (\mathbf{pure}\,\pi_1 r, \mathbf{let\ pure}\, q' = \pi_2 r\ \mathbf{in}\, \pi_1 q', \mathbf{let\ pure}\, q' = \pi_2 r\ \mathbf{in}\, \pi_2 q')$
$\mathbf{in}\, \langle x, \langle y, z\rangle\rangle]$

We may simplify the current instance of tensorial strength:

$[p, \textbf{let pure}\, x, y, z = (\textbf{pure}\, \pi_1 p, \textbf{let pure}\, p' = \pi_2 p\, \textbf{in}\, \pi_1 p', \textbf{let pure}\, p' = \pi_2 p\, \textbf{in}\, \pi_2 p')$
$\textbf{in}\, \langle x, \langle y, z\rangle\rangle] =$
Substitution and reduction
$[p, \textbf{let pure}\, x, p', z = (\textbf{pure}\, \pi_1 p, \pi_2 p, \textbf{let pure}\, p' = \pi_2 p\, \textbf{in}\, \pi_2 p')\, \textbf{in}\, \langle x, \langle \pi_1 p', z\rangle\rangle] =$
Substitution and reduction
$[p, \textbf{let pure}\, x, p' = \textbf{pure}\, \pi_1 p, \pi_2 p\, \textbf{in}\, \langle x, \langle \pi_1 p', \pi_2 p'\rangle\rangle]$

Then:

$\tau_{\hat{A}, \hat{B}\times\hat{C}} \circ (id_{\hat{A}} \times \tau_{\hat{B},\hat{C}}) \circ \alpha_{\hat{A},\hat{B},\mathbf{K}\hat{C}} =$
Unfolding
$\tau_{\hat{A}, \hat{B}\times\hat{C}} \circ [q, \langle \pi_1 q, \textbf{let pure}\, y, z = \textbf{pure}\, (\pi_1(\pi_2 q)), \pi_2(\pi_2 q)\, \textbf{in}\, \langle y, z\rangle\rangle] \circ$
$\circ\, [p, \langle \pi_1(\pi_1 p), \langle \pi_2(\pi_1 p), \pi_2 p\rangle\rangle] =$
Composition
$\tau_{\hat{A}, \hat{B}\times\hat{C}} \circ [p, \langle \pi_1 q, \textbf{let pure}\, y, z = \textbf{pure}\, (\pi_1(\pi_2 q)), \pi_2(\pi_2 q)\, \textbf{in}\, \langle y, z\rangle\rangle$
$[q := \langle \pi_1(\pi_1 p), \langle \pi_2(\pi_1 p), \pi_2 p\rangle\rangle]] =$
Substitution and reduction
$\tau_{\hat{A}, \hat{B}\times\hat{C}} \circ [p, \langle \pi_1(\pi_1 p), \textbf{let pure}\, y, z = \textbf{pure}\, \pi_2(\pi_1 p), \pi_2 p\, \textbf{in}\, \langle y, z\rangle\rangle] =$
Composition
$[r, \textbf{let pure}\, x, p' = \textbf{pure}\, \pi_1 r, \pi_2 r\, \textbf{in}\, \langle x, \langle \pi_1 p', \pi_2 p'\rangle\rangle] \circ$
$\circ\, [p, \langle \pi_1(\pi_1 p), \textbf{let pure}\, y, z = \textbf{pure}\, \pi_2(\pi_1 p), \pi_2 p\, \textbf{in}\, \langle y, z\rangle\rangle] =$
Substitution and reduction
$[p, \textbf{let pure}\, x, p' = (\textbf{pure}\, (\pi_1(\pi_1 p)), \textbf{let pure}\, y, z = (\textbf{pure}\, \pi_2(\pi_1 p), \pi_2 p\, \textbf{in}\, \langle y, z\rangle\rangle)$
$\textbf{in}\, \langle x, \langle \pi_1 p', \pi_2 p'\rangle\rangle] =$
Reduction
$[p, \textbf{let pure}\, x, y, z = \textbf{pure}\, (\pi_1(\pi_1 p)), \textbf{pure}\, \pi_2(\pi_1 p), \pi_2 p\, \textbf{in}\, \langle x, \langle \pi_1 p', \pi_2 p'\rangle\rangle[p' := \langle y, z\rangle]] =$
Substitution and reduction
$[p, \textbf{let pure}\, x, y, z = \textbf{pure}\, (\pi_1(\pi_1 p)), \textbf{pure}\, \pi_2(\pi_1 p), \pi_2 p\, \textbf{in}\, \langle x, \langle y, z\rangle\rangle]$

□

Lemma 19.
$\tau_{A,B} \circ id_A \times \eta_B = \eta_{\widehat{A\times B}}$

Proof.

$\tau_{\hat{A},\hat{B}} \circ id_{\hat{A}} \times \eta_{\hat{B}} =$
Unfolding
$[q, \textbf{let pure}\, x, y = \textbf{pure}\, (\pi_1 q), \pi_2 q\, \textbf{in}\, \langle x, y\rangle] \circ [p, \langle \pi_1 p, \textbf{pure}\, (\pi_2 p)\rangle] =$
Composition
$[p, \textbf{let pure}\, x, y = \textbf{pure}\, (\pi_1 q), \pi_2 q\, \textbf{in}\, \langle x, y\rangle[q := \langle \pi_1 p, \textbf{pure}\, (\pi_2 p)\rangle]]$
Substitution
$[p, \textbf{let pure}\, x, y = \textbf{pure}\, (\pi_1 p), \textbf{pure}\, (\pi_2 p)\, \textbf{in}\, \langle x, y\rangle] =$
β-reduction rule
$[p,\ \textbf{pure}\, (\langle x, y\rangle)[x := \pi_1 p, y := \pi_2 p]]$
Substitution
$[p,\ \textbf{pure}\, (\langle \pi_1 p, \pi_2 p\rangle)] =$
η-reduction
$[p,\ \textbf{pure}\, p] =$
The definition of η
$\eta_{\widehat{A\times B}}$

□

Lemma 20. **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 $M =_{\beta\eta} N \Leftrightarrow [\![\Gamma \vdash M : A]\!] = [\![\Gamma \vdash N : A]\!]$.

☐

4 Acknowledgment

We are grateful to Valeria de Paiva, Neel Krishnaswami and Vladimir Vasyukov for consulting and advice.

Special thanks to Vladimir Krupski for general idea, critics and support.

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

[25] G. A. Kavvos. The Many Worlds of Modal Λ–calculi: I. Curry-Howard for Necessity, Possibility and Time

[26] Ross Paterson. in Mathematics of Program Construction, Madrid, 2012, Lecture Notes in Computer Science, vol. 7342, pp. 300–323, Springer, 2012.