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, $\Box A$ 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) $\Box(A \to B) \to (\Box A \to \Box B)$ *(normality);*
3) $A \to \Box A$ *(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] or Purescript [5] 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[6] or in the next one textbook [7]

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 [11].

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 λ_{IEL^-}) which is Curry-Howard isomorphic to IEL^-. So we will consider IEL$^-$ 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 [25]).

λ_{IEL^-} consists of the rules for simply typed lambda-calculus and special typing rules for lifting types into the applicative functor $\Box$. 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:*

$$\frac{\Gamma \vdash A}{\Gamma \vdash \Box A}\Box_I \qquad \frac{\Gamma \vdash \Box A_1, \dots, \Gamma \vdash \Box A_n \qquad A_1, \dots, A_n \vdash B}{\Gamma \vdash \Box B}$$

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 [24]).

We will denote $\Gamma \vdash \Box A_1, \dots, \Gamma \vdash \Box A_n$ and $A_1, \dots, A_n \vdash B$ as $\Gamma \vdash \Box \vec{A}$ 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 $\mathrm{IEL}^- \vdash \bigwedge \Gamma \to \Box A$.

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

2) If $\Gamma \vdash_{\mathrm{NIEL}^-} \Box \vec{A}$ and $\vec{A} \vdash B$, then $\mathrm{IEL}^- \vdash \bigwedge \Gamma \to \Box B$.

(1)	$\bigwedge \Gamma \to \Box A_1, \dots, \bigwedge \Gamma \to \Box A_1$	assumption
(2)	$\bigwedge \Gamma \to \bigwedge\limits_{i=1}^{n} \Box A_i$	IEL^- theorem
(3)	$\bigwedge\limits_{i=1}^{n} \Box A_i \to \Box \bigwedge\limits_{i=1}^{n} A_i$	IEL^- theorem
(4)	$\bigwedge \Gamma \to \Box \bigwedge\limits_{i=1}^{n} A_i$	from (1), (2) and transitivity
(5)	$\bigwedge\limits_{i=1}^{n} A_i \to B$	assumption
(6)	$(\bigwedge\limits_{i=1}^{n} A_i \to B) \to \Box(\bigwedge\limits_{i=1}^{n} A_i \to B)$	co-reflection
(7)	$\Box(\bigwedge\limits_{i=1}^{n} A_i \to B)$	from (4), (5) and MP
(8)	$\Box \bigwedge\limits_{i=1}^{n} A_i \to \Box B$	from (6) and normality
(9)	$\bigwedge \Gamma \to \Box B$	from (3), (7) and transitivity

$\Box$

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. $\Box$

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 $\Lambda_{\Box}$ of terms is defined by the grammar:

$$\Lambda_{\Box} ::= \mathbb{V} \mid (\lambda \mathbb{V}.\Lambda_{\Box}) \mid (\Lambda_{\Box}\Lambda_{\Box}) \mid \langle \Lambda_{\Box}, \Lambda_{\Box} \rangle \mid (\pi_1 \Lambda_{\Box}) \mid (\pi_2 \Lambda_{\Box}) \mid$$
$$(\mathbf{box}\ \Lambda_{\Box}) \mid (\mathbf{let\ box}\ \mathbb{V}^* = \Lambda_{\Box}^* \ \mathbf{in}\ \Lambda_{\Box})$$

Where $\mathbb{V}^*$ and $\Lambda_{\Box}^*$ denote the set of finite sequences of variables $\bigcup\limits_{i=0}^{\infty} \mathbb{V}^i$ and the set of finite sequences of terms $\bigcup\limits_{i=0}^{\infty} \Lambda_{\Box}^i$. 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 $\mathbb{T}_{\Box}$ of types with applicative functor $\Box$ is generated by the grammar:

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

Context, domain of context and range of context are defined standardly [10][11].

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

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

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

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

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

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathbf{box}\ M : \Box A} \Box_I \qquad \frac{\Gamma \vdash \vec{M} : \Box\vec{A} \qquad \vec{x} : \vec{A} \vdash N : B}{\Gamma \vdash \mathbf{let\,box}\, \vec{x} = \vec{M}\ \mathbf{in}\ N : \Box B}\, let_\Box$$

$\Box_I$-typing rule is the same as $\bigcirc$-introduction in monadic metalanguage [16]. $\Box_I$ allows to inject an object of type A into the functor $\Box$. $\Box_I$ reflects the Haskell method **box** for Applicative class. It plays the same role as the **return** method in Monad class.

let$_\Box$ is similar to the $\Box$-rule in typed lambda calculus for intuitionistic normal modal logic **IK**, which is described in [18].

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

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

Here are some examples of derivation trees:

$$\cfrac{\cfrac{x : A \vdash x : A}{x : A \vdash \mathbf{box}\, x : \Box A}}{\vdash (\lambda x.\mathbf{box}\, x) : A \to \Box A}$$

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

Now we define free variables and substitutions:

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. $FV(\mathbf{box}\ M) = FV(M)$;

7. $FV(\mathbf{let\,box}\, \vec{x} = \vec{M}\ \mathbf{in}\ N) = \bigcup\limits_{i=1}^{n} FV(M)$, *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. $(\lambda x.M)[x := N] = \lambda x.M[y := N]$, $y \in FV(M)$;

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. $(\mathbf{box}\, M)[x := P] = \mathbf{box}\,(M[x := P])$;

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

Substitution and free variable for terms of kind **let box** are defined similary to [18].

Definition 8. *β-reduction and η-reduction rules for λ_{IEL^-}.*

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. $\mathbf{let\,box}\, \vec{x}, y, \vec{z} = \vec{M}, \mathbf{let\,box}\, \vec{w} = \vec{N}\ \mathbf{in}\ Q, \vec{P}\ in\ R \to_\beta$
   $\mathbf{let\,box}\, \vec{x}, \vec{w}, \vec{z} = \vec{M}, \vec{N}, \vec{P}\ \mathbf{in}\ R[y := Q]$

5. $\mathbf{let\,box}\, \vec{x} = \mathbf{box}\, \vec{M}\ \mathbf{in}\ N \to_\beta \mathbf{box}\, N[\vec{x} := \vec{M}]$

6. $\mathbf{let\,box}\, \_\_ = \_\_\ \mathbf{in}\, M \ \to_\beta\ \mathbf{box}\, M$, *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. $\mathbf{let\,box}\, x = M\ \mathbf{in}\ x \to_\eta M$;

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

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

[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 [10] [11]. We will consider only modal cases

Lemma 3. *Generation for* $\Box_I$.
Let $\Gamma \vdash \mathbf{box}\, M : \Box A$*, then* $\Gamma \vdash M : A$*;*

Proof. Straightforwardly. $\square$

Lemma 4. *Basic lemmas.*

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

Proof.
1) The derivation ends in

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

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

2)-3) Similary.

$\square$

Theorem 1. *Subject reduction*
If $\Gamma \vdash M : A$ *and* $M \twoheadrightarrow_r N$*, then* $\Gamma \vdash N : A$

Proof. Induction on the derivation $\Gamma \vdash M : A$ and on the generation of $\rightarrow_r$.
For cases with axiom, application, abstraction and pairs see [11] [12].
1) If $\Gamma \vdash \mathbf{let\,box}\, \vec{x}, y, \vec{z} = \vec{M}, \mathbf{let\,box}\, \vec{w} = \vec{N}\, \mathbf{in}\, Q, \vec{P}\, \mathbf{in}\, R : \Box B$, then $\Gamma \vdash \mathbf{let\,box}\, \vec{x}, \vec{w}, \vec{z} = \vec{M}, \vec{N}, \vec{P}\, \mathbf{in}\, R[y := Q] : \Box B$ by rule 4).
2) Let $\Gamma \vdash \mathbf{let\,box}\, x = M\, \mathbf{in}\, x : \Box A$, then $\Gamma \vdash M : \Box A$ by rule 9).
See [18].
3) The derivation ends in

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

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:

$$\frac{\Gamma \vdash N[\vec{x} := \vec{M}] : B}{\Gamma \vdash \mathbf{box}\, N[\vec{x} := \vec{M}] : \Box B}\ \Box_I$$

4) The derivation ends in

$$\frac{\vdash M : A}{\vdash \mathbf{let\,box}\, \_\_ = \_\_\, \mathbf{in}\, M : \Box A}$$

So, if $\vdash M : A$, then $\vdash \mathbf{box}\, M : \Box A$.

$\square$

Theorem 2.
$\twoheadrightarrow_r$ is strongly normalizing;

Proof.
Let us define the transformation from λ_{IEL^-} into the simple type theory with $\rightarrow$, $\times$ and natural number type $\mathbb{N}$ [4] with additional typing and reduction rules:

$$\frac{}{\Gamma \vdash 0 : \mathbb{N}}$$

$$\frac{\Gamma \vdash n : \mathbb{N}}{\Gamma \vdash \mathbf{succ}\, n : \mathbb{N}}$$

$$\frac{\Gamma \vdash n : \mathbb{N} \qquad \Gamma \vdash m : \mathbb{N}}{\Gamma \vdash n + m : \mathbb{N}}$$

1. $n + 0 \rightarrow_\beta n$;
2. $(n + \mathbf{succ}\, m) \rightarrow_\beta \mathbf{succ}\,(n + m)$

The transformation $|.|$ is defined as follows:

Definition 9. *Interpretaion of types:*

1. $A \in \mathbb{T} \Rightarrow |A| = A$;
2. $|A \rightarrow B| = |A| \rightarrow |B|$;
3. $|A \times B| = |A| \times |B|$;
4. $|\Box A| = \mathbb{N} \times |A|$.

Definition 10. *Interpretation of terms:*

1. $x \in \mathbb{V} \Rightarrow |x| = x$;
2. $|\lambda x.M| = \lambda x.|M|$;
3. $|(MN)| = |M||N|$;
4. $|\langle M, N\rangle| = \langle |M|, |N|\rangle$;
5. $|\pi_i M| = \pi_i |M|,\ i \in \{1, 2\}$;
6. $|\mathbf{box}\, M| = \langle 0, M\rangle$;
7. $|\mathbf{let\ box}\, \_\_ = \_\_\ \mathbf{in}\, M| = \langle 0, M\rangle$;
8. $|\mathbf{let\ box}\, x = N\ \mathbf{in}\, M| = \langle \pi_1|N|, |M|[x := \pi_2|N|]\rangle$
9. $|\mathbf{let\ box}\, \vec{x} = \vec{N}\ \mathbf{in}\, M| = \langle \sum_{i=1}^{n} \pi_1|N|, |M|[\vec{x} := \pi_2\vec{N}]\rangle$

Let us consider the interpretation for $\mathrm{let}_\Box$ rule:

[4] Strong normalization for stronger system was shown here [13]

$$\frac{|\Gamma \vdash \vec{N} : \Box\vec{A}| = |\Gamma| \vdash |\vec{N}| : \mathbb{N} \times |\vec{A}| \qquad |\vec{x} : \vec{A} \vdash M : B| = \vec{x} : |\vec{A}| \vdash |M| : |B|}{|\Gamma \vdash \textbf{let box}\, \vec{x} = \vec{N} \textbf{ in } M : \Box B| = |\Gamma| \vdash \langle \sum_{i=1}^{n} \pi_1 |N|, |M|[\vec{x} := \pi_2 \vec{N}] \rangle : \mathbb{N} \times |B|} \text{let}_\Box$$

Lemma 5. *:*
$|M[x := N]| = |M|[x := |N|]$ *for any term* M*.*

Proof. Induction on the structure of M. □

Lemma 6. $M \twoheadrightarrow_r N \Rightarrow |M| \twoheadrightarrow_{\beta\eta} |N|$

Proof. Let us consider cases with $\beta\Box$, $\beta\Box$**box** and $\Box id$.

1. $|\textbf{let box}\, x = (\textbf{let box}\, y = N \textbf{ in } P) \textbf{ in } M| =$
   $|\textbf{let box}\, y = N \textbf{ in } M[x := P]|$

   $|\textbf{let box}\, x = (\textbf{let box}\, y = N \textbf{ in } P) \textbf{ in } M| =$
   Interpretation
   $\langle \pi_1 |N|, |M|[x := |P|[y := \pi_2 |N|]] \rangle$
   $|\textbf{let box}\, y = N \textbf{ in } M[x := P]| =$
   $\langle \pi_1 |N|, |M|[x := |P|][y := \pi_2 |N|] \rangle \equiv$
   So far as $y \notin FV(M)$
   $\langle \pi_1 |N|, |M|[x := |P|[y := \pi_2 |N|]]$

2. $|\textbf{let box}\, \vec{x} = \textbf{box}\, \vec{N} \textbf{ in } M| = |\textbf{box}\, M[\vec{x} := \vec{N}]|$

   $|\textbf{let box}\, \vec{x} = \textbf{box}\, \vec{N} \textbf{ in } M| =$
   Interpretation
   $\langle 0 + \dots + 0, |M|[\vec{x} := |\vec{N}|] \rangle \twoheadrightarrow_\beta$
   Multistep reduction for natural numbers
   $\langle 0, |M|[\vec{x} := |\vec{N}|] \rangle =$
   Interpretation
   $|\textbf{box}\, M[\vec{x} := \vec{N}]|$

3. $|\textbf{let box}\, x = M \textbf{ in } x| = |M|$

   $|\textbf{let box}\, x = M \textbf{ in } x| =$
   Interpretation
   $\langle \pi_1 |M|, x[x := \pi_2 |M|] \rangle =$
   Substitution
   $\langle \pi_1 |M|, \pi_2 |M| \rangle \rightarrow_\eta$
   η-reduction for pairs
   $|M|$

4. $|\textbf{let box}\, \_\_ = \_\_ \textbf{ in } M| = \langle 0, |M| \rangle = |\textbf{box}\, M|$

Hence λ_{IEL^-} sounds for $\lambda_{\rightarrow,\times,\mathbb{N}}$, then multistep reduction in λ_{IEL^-} is strongly normalizing, so far as multistep reduction in $\lambda_{\rightarrow,\times,\mathbb{N}}$ is strongly normalizing.
□

□

Theorem 3.

$\twoheadrightarrow_r$ *is confluent.*

Proof. By Newman's lemma [11], if relation is strongly normalizing and locally conluent, then this relation is confluent.

It is sufficient to show that $\twoheadrightarrow_r$ is locally confluent.

Lemma 7. *Local confluence*

If $M \rightarrow_r N$ and $M \rightarrow_r Q$, then there exists some term P, such that $N \twoheadrightarrow_r P$ and $Q \twoheadrightarrow_r P$.

Proof. Let us consider the following critical pairs and show that they are joinable:

1.

$$\begin{array}{ccc} & \textbf{let box}\, x = (\textbf{let box}\, \vec{y} = \textbf{box}\, \vec{N}\ \textbf{in}\ P)\ \textbf{in}\ M & \\ \swarrow^{\beta\Box} & & \searrow^{\beta\Box\textbf{box}} \\ \textbf{let box}\, \vec{y} = \textbf{box}\, \vec{N}\ \textbf{in}\ M[x := P] & & \textbf{let box}\, x = \textbf{box}\, P[\vec{y} := \vec{N}]\ \textbf{in}\ M \end{array}$$

$\textbf{let box}\, \vec{y} = \textbf{box}\, \vec{N}\ \textbf{in}\ M[x := P] \rightarrow_{\beta\Box\textbf{box}}$
$\textbf{box}\, M[x := P][\vec{y} := \vec{N}]$
$\textbf{let box}\, x = \textbf{box}\, P[\vec{y} := \vec{N}]\ \textbf{in}\ M \rightarrow_{\beta\Box\textbf{box}}$
$\textbf{box}\, M[x := P[\vec{y} := \vec{N}]] \equiv$
So far as $x \notin \vec{y}$
$\textbf{box}\, M[x := P][\vec{y} := \vec{N}]$

2.

$$\begin{array}{ccc} & \textbf{let box}\, x = (\textbf{let box}\, \_\_ = \_\_\ \textbf{in}\ N)\ \textbf{in}\ M & \\ \swarrow^{\beta\Box} & & \searrow^{\beta\Box\_\_} \\ \textbf{let box}\, \_\_ = \_\_\ \textbf{in}\ M[x := N] & & \textbf{let box}\, x = \textbf{box}\, N\ \textbf{in}\ M \end{array}$$

$\textbf{let box}\, \_\_ = \_\_\ \textbf{in}\ M[x := N] \rightarrow_{\beta\Box\_\_} \textbf{box}\, (M[x := N])$
$\textbf{let box}\, x = \textbf{box}\, N\ \textbf{in}\ M \rightarrow_{\beta\Box\textbf{box}} \textbf{box}\, (M[x := N])$

$\Box$

Also we may consider four critical pairs which are used in local confluence proof for lambda-calculus based on **IK** and [18].

$\Box$

Theorem 4.

Normal form in call-by-name λ_{IEL^-} 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 box** $\vec{x} = \vec{M}$ **in** N was considered by Kakutani [18] [19].

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

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

3 Categorical semantics

Definition 11. *Monoidal category*

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

1. *A bifunctor $\otimes : \mathcal{C} \times \mathcal{C} \to C$ called tensor product;*
2. *Identity object $\mathbb{1}$;*
3. *A natural isomorphism called associator: $\alpha_{A,B,C} : (A \otimes B) \otimes C \cong A \otimes (B \otimes C)$;*
4. *A natural isomorphism called left unitor: $L_A : \mathbb{1} \otimes A \cong A$;*
5. *A natural isomorphism called right unitor $R_A : A \otimes \mathbb{1} \cong A$;*
6. *A coherence condition called MacLane pentagon, i.e. the following diagram commutes:*

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

7. *A coherence condition called triangle identity:*

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

Definition 12. *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 monoidal category, where tensor is a product and identity object is a terminal object.

Definition 13. *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 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:

1. $u : \mathbb{1}_{\mathcal{D}} \to \mathcal{F}\mathbb{1}_{\mathcal{C}}$;

2. $A \circledast B : \mathcal{F}A \otimes_{\mathcal{D}} \mathcal{F}B \to \mathcal{F}(A \otimes_{\mathcal{C}} B)$.

and coherence maps:

- *Associativity:*

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

- *Left unitality:*

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

- *Right unitality:*

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

Definition 14. *Applicative functor*
An applicative functor is a triple $\langle \mathcal{C}, \mathcal{F}, \eta \rangle$, where $\mathcal{C}$ is a monoidal category, $\mathcal{F}$ is a monoidal endofunctor and $\eta : Id_{\mathcal{C}} \Rightarrow \mathcal{F}$ is a natural transformation (similar to unit in monad), such that:

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

2. $A \circledast B \circ (\eta_A \otimes \eta_B) = \eta_{A\otimes B}$, *i.e. the following diagram commutes:*

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

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

3.1 Soundness and completeness

Definition 15. *Semantical translation from λ_{IEL^-} to some cartesian closed category $\mathcal{C}$ with an applicative functor $\boxdot$:*

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;*
   - $[\![A \to B]\!] := [\![B]\!]^{[\![A]\!]}$;
   - $[\![A \times B]\!] := [\![A]\!] \times [\![B]\!]$.
2. *Interpretation for modal types:*
   (a) $[\![\Box A]\!] = \boxdot[\![A]\!]$;
3. *Interpretaion for contexts:*
   (a) $[\![\quad]\!] = \mathbb{1}$, *where* $\mathbb{1}$ *is a terminal object of a given ccc;*
   (b) $[\![\Gamma, x : A]\!] = [\![\Gamma]\!] \times [\![A]\!]$
4. *Interpretation for typing assignment:*
   (a) $[\![\Gamma \vdash M : A]\!] := [\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]$.
5. *Interpretation for typing rules:*

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

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

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

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

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

$$\frac{[\![\Gamma \vdash M : A]\!] = [\![M]\!] : [\![\Gamma]\!] \to [\![A]\!]}{[\![\Gamma \vdash \mathbf{box}\, M : \Box A]\!] := [\![\Gamma]\!] \xrightarrow{[\![M]\!]} [\![A]\!] \xrightarrow{\eta_{[\![A]\!]}} \boxdot[\![A]\!]}$$

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

Theorem 5. *Soundness*
Let $\Gamma \vdash M : A$ *and* $M =_r N$, *then* $[\![\Gamma \vdash M : A]\!] = [\![\Gamma \vdash N : A]\!]$

Proof.

Lemma 8.
$[\![M[x_1 := M_1, \ldots, x_n := M_n]]\!] = [\![M]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!] \rangle$.

Proof.
1)

$[\![\Gamma \vdash (\mathbf{box}\, M)[\vec{x} := \vec{M}] : \Box A]\!] =$
By substitution definition
$[\![\Gamma \vdash \mathbf{box}\, (M[\vec{x} := \vec{M}]) : \Box A]\!]$
Interpretation for **box**
$\eta_{[\![A]\!]} \circ [\![(M[\vec{x} := \vec{M}])]\!]$
Assumption
$\eta_{[\![A]\!]} \circ ([\![M]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!] \rangle) =$
Associativity of composition
$(\eta_{[\![A]\!]} \circ [\![M]\!]) \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!] \rangle =$
Interpretation for **box**
$[\![\Gamma \vdash \mathbf{box}\, M : \Box A]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!] \rangle =$

2)

$[\![\Gamma \vdash (\mathbf{let\,box}\, \vec{x} = \vec{M}\ \mathbf{in}\, N)[\vec{y} := \vec{P}] : \Box B]\!] =$
Substitution
$[\![\Gamma \vdash \mathbf{let\,box}\, \vec{x} = (\vec{M}[\vec{y} := \vec{P}])\ \mathbf{in}\, N : \Box B]\!] =$
Interpretation for $let_{\Box}$
$\boxdot([\![N]\!]) \circ ([\![A_1]\!] * \cdots * [\![A_n]\!]) \circ [\![\Gamma \vdash (\vec{M}[\vec{y} := \vec{P}]) : \Box \vec{A}]\!] =$
Induction hypothesis
$\boxdot([\![N]\!]) \circ ([\![A_1]\!] * \cdots * [\![A_n]\!]) \circ ([\![\vec{M}]\!] \circ \langle [\![P_1]\!], \ldots, [\![P_n]\!] \rangle) =$
Associativity of composition
$(\boxdot([\![N]\!]) \circ ([\![A_1]\!] * \cdots * [\![A_n]\!]) \circ [\![\vec{M}]\!]) \circ \langle [\![P_1]\!], \ldots, [\![P_n]\!] \rangle =$
Interpretaion
$[\![\Gamma \vdash \mathbf{let\,box}\, \vec{x} = \vec{M}\ \mathbf{in}\, N : \Box B]\!] \circ \langle [\![P_1]\!], \ldots, [\![P_n]\!] \rangle$

$\Box$

Lemma 9.
Let $\Gamma \vdash M : A$ *and* $M \to_r N$, *then* $[\![\Gamma \vdash M : A]\!] = [\![\Gamma \vdash N : A]\!]$*;*

Proof.
Cases with β-reductions for $let_{\Box}$ are shown in [19]. Let us consider cases with **box**.

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

$[\![\Gamma \vdash \textbf{let box}\ \vec{x} = \textbf{box}\ \vec{M}\ \textbf{in}\ N : \Box B]\!] =$
By interpretation
$\boxdot([\![N]\!]) \circ ([\![A_1]\!] \circledast \cdots \circledast [\![A_n]\!]) \circ \langle \eta_{[\![A_1]\!]} \circ [\![M_1]\!], \ldots, \eta_{[\![A_n]\!]} \circ [\![M_n]\!]\rangle =$
By the property of a pair of morphisms
$\boxdot([\![N]\!]) \circ ([\![A_1]\!] \circledast \cdots \circledast [\![A_n]\!]) \circ (\eta_{[\![A_1]\!]} \times \cdots \times \eta_{[\![A_n]\!]}) \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle =$
Associativity of composition
$\boxdot([\![N]\!]) \circ (([\![A_1]\!] \circledast \cdots \circledast [\![A_n]\!]) \circ (\eta_{[\![A_1]\!]} \times \cdots \times \eta_{[\![A_n]\!]})) \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle =$
By the definition of an applicative functor
$\boxdot([\![N]\!]) \circ \eta_{[\![A_1]\!] \times \cdots \times [\![A_n]\!]} \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle =$
Naturality of η
$\eta_{[\![B]\!]} \circ [\![N]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle =$
Associativity of composition
$\eta_{[\![B]\!]} \circ ([\![N]\!] \circ \langle [\![M_1]\!], \ldots, [\![M_n]\!]\rangle) =$
Substitution lemma
$\eta_{[\![B]\!]} \circ [\![\Gamma \vdash N[\vec{x} := \vec{M}] : \Box B]\!]$
By interpetation
$[\![\Gamma \vdash \textbf{box}\ (N[\vec{x} := \vec{M}]) : \Box B]\!]$

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

□

Theorem 6. *Completeness*
Let $[\![\Gamma \vdash M : A]\!] = [\![\Gamma \vdash N : A]\!]$*, then* $M =_r N$.

Proof.
We will consider term model for simply typed lambda calculus $\times$ and $\rightarrow$ standardly described in [21]:

Definition 16. *Equivalence on term pairs:*
Let us define relation $\sim_{A,B} \subseteq (\mathbb{V} \times \Lambda_{\Box})^2$*, such that:*
$(x, M) \sim_{A,B} (y, N) \Leftrightarrow x : A \vdash M : B \ \&\ y : A \vdash N : A \ \&\ M =_r N[y := x]$;

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

1. $Ob_{\mathcal{C}} = \{\hat{A} \mid A \in \mathbb{T}\} \cup \{\mathbb{1}\}$;
2. $Hom_{\mathcal{C}(\lambda)}(\hat{A}, \hat{B}) = \{[x, M] \mid x : A \vdash_{\lambda_{IEL^-}} M : B\}$;
3. *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* $[y, M] \circ [x, M] = [x, N[y := M]]$;

4. *Identity morphism* $id_{\hat{A}} = [x, x] \in Hom_{\mathcal{C}(\lambda)(\hat{A},\hat{A})}$;
5. $\mathbb{1}$ *is a terminal object;*
6. $\widehat{A \times B} = \hat{A} \times \hat{B}$;
7. *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* $i \in \{1, 2\}$;
8. $\widehat{A \to B} = \hat{B}^{\hat{A}}$;
9. *Evaluation arrow* $\epsilon = [x, (\pi_2 x)(\pi_1 x)] \in Hom_{\mathcal{C}(\lambda)(\hat{B}^{\hat{A}} \times \hat{A}, \hat{B})}$.

We define endofunctor $\boxdot$ on $\mathcal{C}(\lambda)$ and natural transformation η from $id_{\mathcal{C}(\lambda)}$ to this endofunctor. It is sufficient to show $\boxdot$ and η form an applicative functor on $\mathcal{C}(\lambda)$ for theorem proof.

Definition 18. *Let us define an endofunctor* $\boxdot : \mathcal{C}(\lambda) \to \mathcal{C}(\lambda)$, *such that forall* $[x, M] \in Hom_{\mathcal{C}(\lambda)}(\hat{A}, \hat{B})$, $\boxdot([x, M]) = [y, \textbf{let box}\, x = y\, \textbf{in}\, M] \in Hom_{\mathcal{C}(\lambda)}(\boxdot\hat{A}, \boxdot\hat{B})$ *(denotation: fmap f for an arbitrary arrow f).*

Lemma 10. *Functoriality*

1. *fmap* $(g \circ f) =$ *fmap* $(g) \circ$ *fmap* (f);
2. *fmap* $(id_{\hat{A}}) = id_{\boxdot}\hat{A}$.

Proof. Easy checking using reduction rules. □

Definition 19. *Let us define natural transformations:*

1. $\eta : Id_{\mathcal{C}(\lambda)} \Rightarrow \boxdot$, *s. t.* $\forall \hat{A} \in Ob_{\mathcal{C}(\lambda)}$, $\eta_{\hat{A}} = [x, \textbf{box}\, x] \in Hom_{\mathcal{C}(\lambda)}(\hat{A}, \boxdot\hat{A})$;
2. $\hat{A} \circledast \hat{B} : \boxdot\hat{A} \times \boxdot\hat{B} \to \boxdot(\hat{A} \times \hat{B})$, *s. t.* $\forall \hat{A}, \hat{B} \in Ob_{\mathcal{C}(\lambda)}, *_{\hat{A},\hat{B}} = [p, \textbf{let box}\, x, y = \pi_1 p, \pi_2 p\, \textbf{in}\, \langle x, y\rangle] \in Hom_{\mathcal{C}(\lambda)}(\boxdot\hat{A} \times \boxdot\hat{B}, \boxdot(\hat{A} \times \hat{B}))$.

Implementation for $*$ in our term model is the instance of let$_\Box$-rule:

$$\dfrac{\dfrac{p : \Box A \times \Box B \vdash p : \Box A \times \Box B}{p : \Box A \times \Box B \vdash \pi_1 p : \Box A} \quad \dfrac{p : \Box A \times \Box B \vdash p : \Box A \times \Box B}{p : \Box A \times \Box B \vdash \pi_2 p : \Box B} \quad \dfrac{x : A \vdash x : A \qquad y : B \vdash y : B}{x : A, y : B \vdash \langle x, y\rangle : A \times B}}{p : \Box A \times \Box B \vdash \textbf{let box}\, x, y = \pi_1 p, \pi_2 p\, \textbf{in}\, \langle x, y\rangle : \Box(A \times B)}$$

Lemma 11.
$\boxdot$ *is a monoidal endofunctor*

Proof.
See [18] □

Lemma 12. *Naturality and coherence for* η*:*

1. *fmap* $f \circ \eta_A = \eta_B \circ f$;
2. $(\hat{A} \circledast \hat{B}) \circ (\eta_A \times \eta_B) = \eta_{\hat{A} \times \hat{B}}$;

Proof.

1. fmap $f \circ \eta_{\hat{A}} = \eta_{\hat{B}} \circ f$

   $\eta_{\hat{B}} \circ f =$
   $[y, \mathbf{box}\, y] \circ [x, M] =$
   Composition
   $[x, \mathbf{box}\, y[y := M]] =$
   By substitution
   $[x, \mathbf{box}\, M]$

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

2. $(\hat{A} \circledast \hat{B}) \circ (\eta_A \times \eta_B) = \eta_{\hat{A} \times \hat{B}}$

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

$\square$

Definition 20.
$u_{\mathbb{1}} = [\blacksquare, \mathbf{let\ box}\, \_\_ = \_\_\ \mathbf{in}\, \blacksquare] \in Hom_{\mathcal{C}(\lambda)}(\mathbb{1}, \boxdot\mathbb{1})$.

Lemma 13.
$u_{\mathbb{1}} = \eta_{\mathbb{1}}$

Proof. Immediately. $\square$

Lemma 14. $\langle \mathcal{C}(\lambda), \boxdot, \eta \rangle$ *is an applicative functor*

Proof. Immediately follows from previous lemmas in the section. $\square$

$\square$

4 Relation with Moggi's monadic metalanguage

Definition 21. *Monadic metalanguage*

Monadic metalanguage is a simply typed lambda calculus with additional typing rules:

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash \mathbf{val}\, M : \bigcirc A} \bigcirc_I \qquad \frac{\Gamma \vdash M : \bigcirc A \qquad \Gamma, x : A \vdash N : \bigcirc B}{\Gamma \vdash \mathbf{let\ val}\, x = M \,\mathbf{in}\, N : \bigcirc B} \bigcirc_E$$

Definition 22. *Reduction rules for monadic metalanguage*

1. $\mathbf{let\ val}\, x = \mathbf{val}\, M \,\mathbf{in}\, N \to_\beta N[x := M]$;
2. $\mathbf{let\ val}\, x = (\mathbf{let\ val}\, y = N \,\mathbf{in}\, P) \,\mathbf{in}\, M \to_\beta \mathbf{let\ val}\, y = N \,\mathbf{in}\, (\mathbf{let\ val}\, x = P \,\mathbf{in}\, M)$;
3. $\mathbf{let\ val}\, x = M \,\mathbf{in}\, \mathbf{val} x \to_\eta M$.

Let us define translation $\ulcorner . \urcorner$ from λ_{IEL^-} into monadic metalanguage. Without loss of generality we will consider λ_{IEL^-} with $\to$ and $\square$:

Definition 23. *Translation for types*

1. $\ulcorner A \urcorner = A$ *for an atomic* A;
2. $\ulcorner A \to B \urcorner = \ulcorner A \urcorner \to \ulcorner B \urcorner$;
3. $\ulcorner \square A \urcorner = \bigcirc \ulcorner A \urcorner$

Definition 24. *Translation for terms*

1. $\ulcorner x \urcorner = x$ *for* $x \in \mathbb{V}$;
2. $\ulcorner \lambda x.M \urcorner = \lambda x. \ulcorner M \urcorner$;
3. $\ulcorner MN \urcorner = \ulcorner M \urcorner \ulcorner N \urcorner$;
4. $\ulcorner \mathbf{box}\, M \urcorner = \mathbf{val}\, \ulcorner M \urcorner$
5. $\ulcorner \mathbf{let\ box}\, \_\_ = \_\_ \,\mathbf{in}\, N \urcorner = \mathbf{val}\, \ulcorner N \urcorner$;
6. $\ulcorner \mathbf{let\ box}\, \vec{x} = \vec{M} \,\mathbf{in}\, N \urcorner = \mathbf{let\ val}\, \vec{x} = \ulcorner \vec{M} \urcorner \,\mathbf{in\ val}\, \ulcorner N \urcorner$;

Where $\mathbf{let\ val}\, \vec{x} = \ulcorner \vec{M} \urcorner \,\mathbf{in\ val}\, \ulcorner N \urcorner$ denotes $\mathbf{let\ val}\, x_n = \ulcorner M_n \urcorner \,\mathbf{in}\, (\dots (\mathbf{let\ val}\, x_1 = \ulcorner M_1 \urcorner \,\mathbf{in}\, \ulcorner N \urcorner) \dots)$

Definition 25. *Interpretation for modal rules*

$$\frac{\ulcorner \Gamma \vdash M : A \urcorner = \ulcorner \Gamma \urcorner \vdash \ulcorner M \urcorner : \ulcorner A \urcorner}{\ulcorner \Gamma \vdash \mathbf{box}\, M : \square A \urcorner = \ulcorner \Gamma \urcorner \vdash \mathbf{val}\, \ulcorner M \urcorner : \bigcirc \ulcorner A \urcorner}$$

$$\frac{\ulcorner \Gamma \vdash \vec{M} : \Box\vec{A} \urcorner = \ulcorner \Gamma \urcorner \vdash \ulcorner \vec{M} \urcorner : \bigcirc \ulcorner \vec{A} \urcorner \qquad \dfrac{\ulcorner \vec{x} : \vec{A} \vdash N : B \urcorner = \vec{x} : \ulcorner \vec{A} \urcorner \vdash \ulcorner N \urcorner : \ulcorner B \urcorner}{\vec{x} : \ulcorner \vec{A} \urcorner \vdash \mathbf{val}\, \ulcorner N \urcorner : \bigcirc \ulcorner B \urcorner}}{\ulcorner \Gamma \vdash \mathbf{let\ box}\, \vec{x} = \vec{M}\ \mathbf{in}\ N : \Box B \urcorner = \ulcorner \Gamma \urcorner \vdash \mathbf{let\ val}\, \vec{x} = \ulcorner \vec{M} \urcorner\ \mathbf{in\ val}\, \ulcorner N \urcorner : \bigcirc \ulcorner B \urcorner}$$

Lemma 15.
$\ulcorner M[x := N] \urcorner = \ulcorner M \urcorner [x := \ulcorner N \urcorner]$

Proof. Induction on the structure of M. □

Lemma 16.
If $M =_r N$, *then* $\ulcorner M \urcorner =_{\beta\eta} \ulcorner N \urcorner$.

Proof.
1)
$\ulcorner \mathbf{let\ box}\, x = (\mathbf{let\ box}\, \vec{y} = \vec{N}\ \mathbf{in}\ P)\ \mathbf{in}\ M \urcorner =$
$\mathbf{let\ val}\, x = (\mathbf{let\ val}\, \vec{y} = \ulcorner \vec{N} \urcorner\ \mathbf{in\ val}\, \ulcorner P \urcorner)\ \mathbf{in\ val}\, \ulcorner M \urcorner \to_\beta$
$\mathbf{let\ val}\, \vec{y} = \ulcorner \vec{N} \urcorner\ \mathbf{in}\ (\mathbf{let\ val}\, x = \ulcorner P \urcorner\ \mathbf{in\ val}\, \ulcorner M \urcorner) \to_\beta$
$\mathbf{let\ val}\, \vec{y} = \ulcorner \vec{N} \urcorner\ \mathbf{in\ val}\, \ulcorner M \urcorner [x := \ulcorner P \urcorner] =$
$\ulcorner \mathbf{let\ box}\, \vec{y} = \vec{N}\ \mathbf{in}\ M[x := P] \urcorner$

2)
$\ulcorner \mathbf{let\ box}\, \vec{x} = \mathbf{box}\, \vec{N}\ \mathbf{in}\ M \urcorner =$
$\mathbf{let\ val}\, \vec{x} = \mathbf{val}\, \ulcorner \vec{N} \urcorner\ \mathbf{in\ val} \ulcorner M \urcorner \to_\beta$
$\mathbf{val}\, \ulcorner M \urcorner [\vec{x} := \ulcorner \vec{N} \urcorner] =$
$\ulcorner \mathbf{box}\, M[\vec{x} := \vec{N}] \urcorner$

3)
$\ulcorner \mathbf{let\ box}\, x = M\ \mathbf{in}\ x \urcorner =$
$\mathbf{let\ val}\, x = \ulcorner M \urcorner\ \mathbf{in\ val}\, x \to_\eta$
$\ulcorner M \urcorner$

□

Theorem 7.
If $\Gamma \vdash M : A$, *then* $\ulcorner \Gamma \urcorner \vdash \ulcorner M \urcorner : \ulcorner A \urcorner$

Proof. Follows from lemmas above. □

5 Acknowledgment

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

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

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

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

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

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

[11] 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.

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

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

[14] 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.

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

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

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

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

[19] 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

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

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

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

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

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

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