

# 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. We prove that this typed lambda-calculus has the strong normalization and Church-Rosser properties.

## 1 Introduction

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

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

**Definition 1.** *Intuitionistic epistemic logic  $IEL$ :*

- 1) *IPC axioms;*
  - 2)  $\mathbf{K}(A \rightarrow B) \rightarrow (\mathbf{K}A \rightarrow \mathbf{K}B)$  (*normality*);
  - 3)  $A \rightarrow \mathbf{K}A$  (*co-reflection*);
- Rule: MP.*

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

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

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.

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

---

<sup>1</sup>Type class in Haskell is a general interface for special group of datatypes.

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

```

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

```

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

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

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

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

At first we define the natural deduction for  $IEL^-$  with  $\mathbf{K}$ -modality and binary connectives  $\rightarrow$  and  $\wedge$  (we call that calculus  $NIEL_{\wedge, \rightarrow}^-$ ):

**Definition 2.** *Natural deduction  $NIEL_{\wedge, \rightarrow}^-$  for  $IEL^-$  with  $\rightarrow$  and  $\wedge$ :*

$$\begin{array}{c}
\frac{}{\Gamma, A \vdash A} \text{ ax} \\
\\
\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow_i \qquad \frac{\Gamma \vdash A \rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \rightarrow_i \\
\\
\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_i \qquad \frac{\Gamma \vdash A_1 \wedge A_2}{\Gamma \vdash A_i} \wedge_e, i \in \{1, 2\} \\
\\
\frac{\Gamma \vdash A}{\Gamma \vdash \mathbf{K}A} \mathbf{K}_I \qquad \frac{\Gamma \vdash \mathbf{K}\vec{A} \quad \vec{A} \vdash B}{\Gamma \vdash \mathbf{K}B}
\end{array}$$

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

**Lemma 1.**  $\Gamma \vdash_{NIEL_{\wedge, \rightarrow}^-} A \Rightarrow IEL^- \vdash \bigwedge \Gamma \rightarrow A$ .

*Proof.* Induction on the derivation.

Let us consider cases with modality.

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

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

2) If  $\Gamma \vdash_{NIEL_{\bigwedge, \rightarrow}^-} \mathbf{K}\vec{A}$  and  $\vec{A} \vdash B$ , then  $IEL^- \vdash \bigwedge \Gamma \rightarrow \mathbf{K}B$ .

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

□

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

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

It is clearly that these lemmas could be extended for  $IEL^-$  with  $\vee$  and  $\neg$  similary.

At the next step we build the typed lambda-calculus based on  $NIEL_{\bigwedge, \rightarrow}^-$  by proof-assingment in rules.

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

**Definition 3.** *The set of terms:*

*Let  $\mathbb{V}$  be the set of variables. The set  $\Lambda_{\mathbf{K}}$  of terms is defined by the grammar:*

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

**Definition 4.** *The set of types:*

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

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

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  $NIEL_{\bigwedge, \rightarrow}^-$ :*

$$\overline{\Gamma, x : A \vdash x : A}^{ax}$$

$$\begin{array}{c} \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x. M : A \rightarrow B} \rightarrow_i \qquad \frac{\Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash x : A}{\Gamma \vdash fx : B} \rightarrow_e \\[10pt] \frac{\Gamma \vdash M : A \quad \Gamma \vdash N : B}{\Gamma \vdash \langle x, y \rangle : A \times B} \times_i \qquad \frac{\Gamma \vdash M : A_1 \times A_2}{\Gamma \vdash \pi_i M : A_i} \times_e, i \in \{1, 2\} \\[10pt] \frac{\Gamma \vdash x : A}{\Gamma \vdash \mathbf{pure} \ x : \mathbf{KA}} \mathbf{K}_I \qquad \frac{\Gamma \vdash \vec{M} : \mathbf{KA} \quad \vec{x} : \vec{A} \vdash M : B}{\Gamma \vdash \mathbf{let} \ \mathbf{pure} \ \vec{x} = \vec{M} \ \mathbf{in} \ M : \mathbf{KB}} \mathbf{let}_{\mathbf{K}} \end{array}$$

$\mathbf{K}_I$ -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  $\alpha$  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.

$\mathbf{let}_{\mathbf{K}}$  is the same as the  $\square$ -rule in typed lambda calculus for intuitionistic normal modal logic  $\mathbf{IK}$ , which is described in [19].

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

Here are some examples of derivation trees.

$$\begin{array}{c} \frac{\frac{x : A \vdash x : A}{x : A \vdash \mathbf{pure} \ x : \mathbf{KA}} \mathbf{K}_I}{\vdash (\lambda x. \mathbf{pure} \ x) : A \rightarrow \mathbf{KA}} \rightarrow_i \\[20pt] \frac{\frac{f : \mathbf{K}(A \rightarrow B) \vdash f : \mathbf{K}(A \rightarrow B) \quad x : \mathbf{KA} \vdash x : \mathbf{KA} \quad \frac{g : A \rightarrow B \quad y : A}{g : A \rightarrow B, y : A \vdash gy : B}}{f : \mathbf{K}(A \rightarrow B), x : \mathbf{KA} \vdash \mathbf{let} \ \mathbf{pure} \ \langle g, y \rangle = \langle f, x \rangle \ \mathbf{in} \ gy : \mathbf{KB}}}{f : \mathbf{K}(A \rightarrow B) \vdash \lambda x. \mathbf{let} \ \mathbf{pure} \ \langle g, y \rangle = \langle f, x \rangle \ \mathbf{in} \ gy : \mathbf{KA} \rightarrow \mathbf{KB}}}{\vdash \lambda f. \lambda x. \mathbf{let} \ \mathbf{pure} \ \langle g, y \rangle = \langle f, x \rangle \ \mathbf{in} \ gy : \mathbf{K}(A \rightarrow B) \rightarrow \mathbf{KA} \rightarrow \mathbf{KB}} \\[20pt] \frac{\frac{f : A \rightarrow B \vdash f : A \rightarrow B}{f : A \rightarrow B \vdash \mathbf{pure} \ f : \mathbf{K}(A \rightarrow B)} \quad x : \mathbf{KA} \vdash x : \mathbf{KA} \quad \frac{g : A \rightarrow B \quad y : A}{g : A \rightarrow B, y : A \vdash gy : B}}{f : A \rightarrow B, x : \mathbf{KA} \vdash \mathbf{let} \ \mathbf{pure} \ \langle g, y \rangle = \langle \mathbf{pure} \ f, x \rangle \ \mathbf{in} \ gy : \mathbf{KB}}}{f : A \rightarrow B \vdash \lambda x. \mathbf{let} \ \mathbf{pure} \ \langle g, y \rangle = \langle \mathbf{pure} \ f, x \rangle \ \mathbf{in} \ gy : \mathbf{KA} \rightarrow \mathbf{KB}}}{\lambda f. \lambda x. \mathbf{let} \ \mathbf{pure} \ \langle g, y \rangle = \langle \mathbf{pure} \ f, x \rangle \ \mathbf{in} \ gy : (A \rightarrow B) \rightarrow \mathbf{KA} \rightarrow \mathbf{KB}} \end{array}$$

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

**Definition 6.** Set  $FV(M)$  of free variables for arbitrary term  $M$ :

- 1)  $FV(x) = \{x\}$ ;
- 2)  $FV(\lambda x. M) = FV(M) \setminus \{x\}$ ;

- 3)  $FV(MN) = FV(M) \cup FV(N)$ ;
- 4)  $FV(\langle M, N \rangle) = FV(M) \cup FV(N)$ ;
- 5)  $FV(\pi_i M) \subseteq FV(M)$ ,  $i \in \{1, 2\}$ ;
- 6)  $FV(\text{pure } M) = FV(M)$ ;
- 7)  $FV(\text{let pure } \vec{N} = \vec{M} \text{ in } M) = \bigcup_{i=1}^n FV(M)$ , 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)  $(\lambda x.M)[x := N] = \lambda x.M[x := N]$ ;
- 4)  $(M, N)[x := P] = (M[x := P], N[x := P])$ ;
- 5)  $(\pi_i M)[x := P] = \pi_i(M[x := P])$ ,  $i \in \{1, 2\}$ ;
- 6)  $(\text{pure } M)[x := P] = \text{pure } (M[x := P])$ ;
- 7)  $(\text{let pure } \vec{x} = \vec{M} \text{ in } M)[y := P] = \text{let pure } \vec{x} = (\vec{M}[y := P]) \text{ in } M$ .

**Definition 8.**  $\beta$ -reduction and  $\eta$ -reduction rules for  $\lambda\mathbf{K}$ .

- 1)  $(\lambda x.M)N \rightarrow_\beta M[x := N]$ ;
- 2)  $\pi_1 \langle M, N \rangle \rightarrow_\beta M$ ;
- 3)  $\pi_2 \langle M, N \rangle \rightarrow_\beta N$ ;
- 4)  $\text{let pure } \langle \vec{x}, y, \vec{z} \rangle = \langle \vec{M}, \text{let pure } \vec{w} = \vec{N} \text{ in } Q, \vec{P} \rangle \text{ in } R \rightarrow_\beta$   
 $\text{let pure } \langle \vec{x}, \vec{w}, \vec{z} \rangle = \langle \vec{M}, \vec{N}, \vec{P} \rangle \text{ in } R[y := Q]$
- 5)  $\text{let pure } \vec{x} = \text{pure } \vec{M} \text{ in } N \rightarrow_\beta \text{pure } N[\vec{x} := \vec{M}]$
- 6)  $\lambda x.fx \rightarrow_\eta f$ ;
- 7)  $\langle \pi_1 P, \pi_2 P \rangle \rightarrow_\eta P$ ;
- 8)  $\text{let pure } \_ = \_ \text{ in } N \rightarrow_\eta \text{pure } N$ ;
- 9)  $\text{let pure } x = M \text{ in } x \rightarrow_\eta M$ ;
- 10)  $M \rightarrow_{\beta\eta} N \Rightarrow \text{pure } M \rightarrow_{\beta\eta} \text{pure } N$

### 3 Basic lemmas

Now we will prove standard lemmas for contexts in type systems<sup>3</sup>:

**Lemma 3.** *Generation lemma.*

- i) Let  $\Gamma \vdash \text{pure } M : \mathbf{K}A$ , then  $\Gamma \vdash M : A$ ;
- ii) Let  $\Gamma \vdash \text{let pure } \vec{x} = \vec{M} \text{ in } N : \mathbf{K}B$ , there are some  $A_1, \dots, A_n \in \mathbb{T}_{\mathbf{K}}$ , such that  $\Gamma \vdash \vec{M} : \mathbf{K}\vec{A}$  and  $\vec{x} : \vec{A} \vdash N : B$ .

*Proof.*

Induction on  $\Gamma \vdash \text{pure } M : \mathbf{K}A$  and  $\Gamma \vdash \text{let pure } \vec{x} = \vec{M} \text{ in } N : \mathbf{K}B$  correspondently.  $\square$

**Lemma 4.** *Weakening.*

Let  $\Gamma \vdash M : A$  and  $\Gamma \subseteq \Delta$ , then  $\Delta \vdash M : A$ .

---

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

*Proof.*

- 1) Let  $\Gamma, x : A \vdash x : A$  and  $\Gamma \subseteq \Delta$ , then  $\Delta, x : A \vdash x : A$  trivially.
- 2) Let  $\Gamma \vdash \mathbf{pure} M : \mathbf{KA}$ . Then  $\Gamma \vdash M : A$  by generation and  $\Delta \vdash M : A$  by assumption. So  $\Delta \vdash \mathbf{pure} M : \mathbf{KA}$  by  $\mathbf{K}_I$ .
- 3) Let  $\Gamma \vdash \mathbf{let pure} \vec{x} = \vec{M} \mathbf{in} N : \mathbf{KB}$  and  $\Gamma \subseteq \Delta$ . Then  $\Gamma \vdash \vec{M} : \mathbf{KA}$  and  $\vec{x} : \vec{A} \vdash N : B$ .  
By assumption  $\Delta \vdash \vec{M} : \mathbf{KA}$ . So  $\Delta \vdash \mathbf{let pure} \vec{x} = \vec{M} \mathbf{in} N : \mathbf{KB}$  by  $\mathbf{let_K}$ .  $\square$

**Definition 9.** *Type substitution*

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

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

**Lemma 5.** *Substitution lemma.*

- i) Let  $\Gamma, x : A \vdash M : B$  and  $\Gamma \vdash N : A$ , then  $\Gamma \vdash M[x := N] : B$ .
- ii) Let  $\Gamma \vdash M : A$ , then  $\Gamma[B := C] \vdash M : (A[B := C])$ .

*Proof.*

i) For term substitution:

- 1) Let  $\Gamma, x : A \vdash x : A$  and  $\Gamma \vdash N : A$ , but  $x[x := N] = N$ , so  $\Gamma \vdash N : A$ .
- 2) Let  $\Gamma, x : A \vdash \mathbf{pure} M : \mathbf{KB}$  and  $\Gamma \vdash N : A$ .  
By generation  $\Gamma, x : A \vdash M : B$  and by assumption  $\Gamma \vdash M[x := N] : B$ .  
By  $\mathbf{K}_I$ ,  $\Gamma \vdash \mathbf{pure} (M[x := N]) : \mathbf{KB}$ .
- 3) Let  $\Gamma, y : A \vdash \mathbf{let pure} \vec{x} = \vec{M} \mathbf{in} N : \mathbf{KB}$  and  $\Gamma \vdash N : A$ .  
By generation,  $\Gamma, y : A \vdash \vec{M} : \mathbf{KA}$  and  $\vec{x} : \vec{A} \vdash N : B$ .  
By hypothesis,  $\Gamma \vdash \vec{M}[x := N] : \mathbf{KA}$ .  
Hence  $\Gamma \vdash \mathbf{let pure} \vec{x} = \vec{M}[x := N] \mathbf{in} N : \mathbf{KB}$ .

ii) For type substitution

- 1) Let  $\Gamma, x : A \vdash x : A$ , so  $\Gamma[A := C], x : (A[A := C]) \vdash x : (A[A := C])$ , or  $\Gamma[A := C], x : C \vdash x : C$ .
- 2) Let  $\Gamma \vdash \mathbf{pure} M : \mathbf{KA}$ . By generation  $\Gamma \vdash M : A$  and by assumption  $\Gamma[B := C] \vdash M : A[B := C]$ .  
By  $\mathbf{K}_I$   $\Gamma \vdash \mathbf{pure} M : \mathbf{K}(A[B := C])$ .
- 3)  $\Gamma \vdash \mathbf{let pure} \vec{x} = \vec{M} \mathbf{in} N : \mathbf{KB}$ . By generation  $\Gamma \vdash \vec{M} : \mathbf{KA}$  and  $\vec{x} : \vec{A} \vdash N : B$ .  
By assumption  $\Gamma[B_1 := C] \vdash \vec{M} : \mathbf{K}\vec{A}[B_1 := C]$  and  $\vec{x} : \vec{A}[B_1 := C] \vdash N : B[B_1 := C]$ .  
So by  $\mathbf{let_K}$ ,  $\Gamma[B_1 := C] \vdash \mathbf{let pure} \vec{x} = \vec{M} \mathbf{in} N : \mathbf{K}(B[B_1 := C])$ .  $\square$

**Theorem 1.** *Subject reduction*

Let  $\Gamma \vdash M : A$  and  $M \rightarrow_{\beta\eta} N$ , then  $\Gamma \vdash N : A$

For cases with application, abstraction and pairs see [12] [13].

1) Let  $\Gamma \vdash \mathbf{let\ pure} \langle \vec{x}, y, \vec{z} \rangle = \langle \vec{M}, \mathbf{let\ pure} \vec{w} = \vec{N} \text{ in } Q, \vec{P} \rangle \text{ in } R : \mathbf{KB}$ ,  
then  $\Gamma \vdash \mathbf{let\ pure} \langle \vec{x}, \vec{w}, \vec{z} \rangle = \langle \vec{M}, \vec{N}, \vec{P} \rangle \text{ in } R[y := Q] : \mathbf{KB}$

2) Let  $\Gamma \vdash \mathbf{let\ pure} x = M \text{ in } x : \mathbf{KA}$ , then  $\Gamma \vdash M : \mathbf{KA}$ .

See [19].

3) Let  $\Gamma \vdash \mathbf{let\ pure} \vec{x} = \mathbf{pure} \vec{M} \text{ in } N : \mathbf{KB}$ .

By generation  $\Gamma \vdash \mathbf{pure} \vec{M} : \mathbf{KA}$  and  $\vec{x} : \vec{A} \vdash N : B$ .

Moreover,  $\Gamma \vdash \vec{M} : \vec{A}$ . By weakening and substitution lemma  $\Gamma \vdash N[\vec{x} = \vec{M}] : B$ .

By  $\mathbf{K}_I$ ,  $\Gamma \vdash \mathbf{pure} N[\vec{x} = \vec{M}] : \mathbf{KB}$ .

4) Let  $\vdash \mathbf{let\ pure} \_ = \_ \text{ in } N : \mathbf{KA}$

By generation  $\vdash N : A$ .

So  $\vdash \mathbf{pure} N : \mathbf{KA}$  by  $\mathbf{K}_I$ .

5) Let  $\Gamma \vdash \mathbf{pure} M : A$  and  $M \rightarrow_{\beta\eta} N$ .

By generation  $\Gamma \vdash M : A$  and  $\Gamma \vdash N : A$  by assumption.

So  $\Gamma \vdash \mathbf{pure} N : \mathbf{KA}$ .

## 4 Strong normalization

## 5 Confluence

## 6 Categorical semantics

**Definition 10.** *Lax monoidal functor*

Let  $\langle \mathcal{C}, \oplus_1, \mathbb{1} \rangle$  and  $\langle \mathcal{D}, \oplus_2, \mathbb{1}' \rangle$  are monoidal categories.

A lax monoidal functor  $\mathcal{F} : \langle \mathcal{C}, \oplus_1, \mathbb{1} \rangle \rightarrow \langle \mathcal{D}, \oplus_2, \mathbb{1}' \rangle$  is a functor  $\mathcal{F} : \mathcal{C} \rightarrow \mathcal{D}$  with additional natural transformations:

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

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

**Definition 11.** *Applicative functor*

An applicative functor is a triple  $\langle \mathcal{C}, \mathcal{K}, \eta \rangle$ , where  $\mathcal{C}$  is a symmetric monoidal category,  $\mathcal{K}$  is a lax monoidal endofunctor and  $\eta$  is a natural transformation, such that:

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

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

3) Weak commutativity condition holds:

$$A \otimes \mathcal{K}B \quad \mathcal{K}A \otimes \mathcal{K}B \quad \mathcal{K}(A \otimes B)$$

$$\mathcal{K}B \otimes A \quad \mathcal{K}B \otimes \mathcal{K}A \quad \mathcal{K}(B \otimes A)$$

By default we will consider an arbitrary closed functor on some cartesian closed category, which is the special case of an applicative functor.

We identify terminal objects. So  $\mathcal{K}(\mathbb{1}) = \mathbb{1}$  and  $\eta_{\mathbb{1}} = id_{\mathbb{1}}$  since  $\mathcal{K}$  is an endofunctor.

## 6.1 Soundness

### Theorem 2. Soundness

Let  $\Gamma \vdash M : A$  and  $M =_{\beta\eta} N$ , then  $\llbracket \Gamma \vdash M : A \rrbracket = \llbracket \Gamma \vdash N : A \rrbracket$

*Proof.*

**Definition 12.** Semantical translation from  $\lambda_K$  to CCC with applicative functor  $\mathcal{K}$ :

1) Interpretation for types:

$\llbracket A \rrbracket := \hat{A}, A \in \mathbb{T};$

$\llbracket A \rightarrow B \rrbracket := \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket;$

$\llbracket A \times B \rrbracket := \llbracket A \rrbracket \times \llbracket B \rrbracket.$

2) Interpretation for modal types:  $\llbracket \mathbf{K}A \rrbracket = \mathcal{K}\llbracket A \rrbracket;$

3) Interpretation for contexts:

$\llbracket \Gamma = \{x_1 : A_1, \dots, x_n : A_n\} \rrbracket := \llbracket \Gamma \rrbracket = \llbracket A_1 \rrbracket \times \dots \times \llbracket A_n \rrbracket;$

4) Interpretation for typing assignment:  $\llbracket \Gamma \vdash M : A \rrbracket := \llbracket M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket.$

5) Interpretation for typing rules:

$$\begin{array}{c}
\frac{}{\llbracket \Gamma, x : A \vdash x : A \rrbracket = \pi_2 : \llbracket \Gamma \rrbracket \times \llbracket A \rrbracket \rightarrow \llbracket A \rrbracket} \\
\frac{\llbracket \Gamma, x : A \vdash M : B \rrbracket = f : \llbracket \Gamma \rrbracket \times \llbracket A \rrbracket \rightarrow \llbracket B \rrbracket}{\llbracket \Gamma \vdash (\lambda x.M) : A \rightarrow B \rrbracket = \Lambda(f) : \llbracket \Gamma \rrbracket \rightarrow \llbracket B \rrbracket^{\llbracket A \rrbracket}} \\
\frac{\llbracket \Gamma \vdash M : A \rightarrow B \rrbracket = \llbracket M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket B \rrbracket^{\llbracket A \rrbracket} \quad \llbracket \Gamma \vdash N : A \rrbracket = \llbracket N \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket}{\llbracket \Gamma \vdash (MN) : B \rrbracket = \llbracket \Gamma \rrbracket \xrightarrow{\langle \llbracket M \rrbracket, \llbracket N \rrbracket \rangle} \llbracket B \rrbracket^{\llbracket A \rrbracket} \times \llbracket A \rrbracket \xrightarrow{\epsilon} \llbracket B \rrbracket} \\
\frac{\llbracket \Gamma \vdash M : A \rrbracket = f : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket \quad \llbracket \Gamma \vdash N : B \rrbracket = g : \llbracket \Gamma \rrbracket \rightarrow \llbracket B \rrbracket}{\llbracket \Gamma \vdash (M, N) : A \times B \rrbracket = \langle f, g \rangle : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket \times \llbracket B \rrbracket} \\
\frac{\llbracket \Gamma \vdash p : A_1 \times A_2 \rrbracket = f : \llbracket \Gamma \rrbracket \rightarrow \llbracket A_1 \rrbracket \times \llbracket A_2 \rrbracket}{\llbracket \Gamma \vdash \pi_i p : A_i \rrbracket = \llbracket \Gamma \rrbracket \xrightarrow{f} \llbracket A_1 \rrbracket \times \llbracket A_2 \rrbracket \xrightarrow{\pi_i} \llbracket A_i \rrbracket} \quad i \in \{1, 2\} \\
\frac{\llbracket \Gamma \vdash M : A \rrbracket = \llbracket M \rrbracket : \llbracket \Gamma \rrbracket \rightarrow \llbracket A \rrbracket}{\llbracket \Gamma \vdash \mathbf{pure} M : \mathbf{K}A \rrbracket := \llbracket \Gamma \rrbracket \xrightarrow{\llbracket M \rrbracket} \llbracket A \rrbracket \xrightarrow{\eta_{\llbracket A \rrbracket}} \mathcal{K}\llbracket A \rrbracket} \\
\frac{\llbracket \Gamma \vdash \vec{M} : \mathbf{K}\vec{A} \rrbracket = \langle \llbracket M_1 \rrbracket, \dots, \llbracket M_n \rrbracket \rangle : \llbracket \Gamma \rrbracket \rightarrow \prod_{i=1}^n \mathcal{K}\llbracket A_i \rrbracket \quad \llbracket \vec{x} : \vec{A} \vdash N : B \rrbracket = \llbracket N \rrbracket : \prod_{i=1}^n \llbracket A_i \rrbracket \rightarrow \llbracket B \rrbracket}{\llbracket \Gamma \vdash \mathbf{let pure} \vec{x} = \vec{M} \mathbf{in} M : \mathbf{K}B \rrbracket = \mathcal{K}(\llbracket N \rrbracket) \circ *_{\llbracket A_1 \rrbracket, \dots, \llbracket A_n \rrbracket} \circ \langle \llbracket M_1 \rrbracket, \dots, \llbracket M_n \rrbracket \rangle : \llbracket \Gamma \rrbracket \rightarrow \mathcal{K}\llbracket B \rrbracket}
\end{array}$$

### Definition 13. Simultaneous substitution

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

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

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

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

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

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



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

**Lemma 6.**

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

*Proof.*

$$1) \llbracket \Gamma \vdash (\mathbf{pure} M)[\vec{x} := \vec{M}] : \mathbf{KA} \rrbracket = \llbracket \Gamma \vdash \mathbf{pure} M : \mathbf{KA} \rrbracket \circ \langle \llbracket M_1 \rrbracket, \dots, \llbracket M_n \rrbracket \rangle.$$

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

$$2) \llbracket \Gamma \vdash (\mathbf{let} \mathbf{pure} \vec{x} = \vec{M} \mathbf{in} N)[\vec{y} := \vec{P}] : \mathbf{KB} \rrbracket = \llbracket \Gamma \vdash \mathbf{let} \mathbf{pure} \vec{x} = \vec{M} \mathbf{in} N : \mathbf{KB} \rrbracket \circ \langle \llbracket P_1 \rrbracket, \dots, \llbracket P_n \rrbracket \rangle$$

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

□

**Lemma 7.**

- i) Let  $\Gamma \vdash M : A$  and  $M \rightarrow_{\beta} N$ , then  $\llbracket \Gamma \vdash M : A \rrbracket = \llbracket \Gamma \vdash N : A \rrbracket$ ;
- ii) Let  $\Gamma \vdash M : A$  and  $M \rightarrow_{\eta} N$ , then  $\llbracket \Gamma \vdash M : A \rrbracket = \llbracket \Gamma \vdash N : A \rrbracket$ ;

*Proof.*

- i) For  $\beta$ -reduction

Cases with  $\beta$ -reductions for  $\mathbf{let}_{\mathbf{K}}$  are shown in [20]. Let us consider cases with  $\mathbf{pure}$ .

$$1) \llbracket \Gamma \vdash \mathbf{let} \mathbf{pure} \vec{x} = \mathbf{pure} \vec{M} \mathbf{in} N : \mathbf{KB} \rrbracket = \llbracket \Gamma \vdash \mathbf{pure} N[\vec{x} := \vec{M}] : \mathbf{KB} \rrbracket$$

$$\begin{aligned}
& \llbracket \Gamma \vdash \text{let pure } \vec{x} = \text{pure } \vec{M} \text{ in } N : \mathbf{KB} \rrbracket = \\
& \quad \text{By interpretation} \\
& \mathcal{K}(\llbracket N \rrbracket) \circ *_{\llbracket A_1 \rrbracket, \dots, \llbracket A_n \rrbracket} \circ \langle \eta_{\llbracket A_1 \rrbracket} \circ \llbracket M_1 \rrbracket, \dots, \eta_{\llbracket A_n \rrbracket} \circ \llbracket M_n \rrbracket \rangle = \\
& \quad \text{By the property of a pair of morphisms} \\
& \mathcal{K}(\llbracket N \rrbracket) \circ *_{\llbracket A_1 \rrbracket, \dots, \llbracket A_n \rrbracket} \circ (\eta_{\llbracket A_1 \rrbracket} \times \dots \times \eta_{\llbracket A_n \rrbracket}) \circ \langle \llbracket M_1 \rrbracket, \dots, \llbracket M_n \rrbracket \rangle = \\
& \quad \text{Associativity of composition} \\
& \mathcal{K}(\llbracket N \rrbracket) \circ (*_{\llbracket A_1 \rrbracket, \dots, \llbracket A_n \rrbracket} \circ (\eta_{\llbracket A_1 \rrbracket} \times \dots \times \eta_{\llbracket A_n \rrbracket})) \circ \langle \llbracket M_1 \rrbracket, \dots, \llbracket M_n \rrbracket \rangle = \\
& \quad \text{By the definition of an applicative functor} \\
& \mathcal{K}(\llbracket N \rrbracket) \circ \eta_{\llbracket A_1 \rrbracket \times \dots \times \llbracket A_n \rrbracket} \circ \langle \llbracket M_1 \rrbracket, \dots, \llbracket M_n \rrbracket \rangle = \\
& \quad \text{Naturality of } \eta \\
& \eta_{\llbracket B \rrbracket} \circ \llbracket N \rrbracket \circ \langle \llbracket M_1 \rrbracket, \dots, \llbracket M_n \rrbracket \rangle = \\
& \quad \text{Associativity of composition} \\
& \eta_{\llbracket B \rrbracket} \circ (\llbracket N \rrbracket \circ \langle \llbracket M_1 \rrbracket, \dots, \llbracket M_n \rrbracket \rangle) = \\
& \quad \text{Simultaneous substitution lemma} \\
& \eta_{\llbracket B \rrbracket} \circ \llbracket N[\vec{x} := \vec{M}] \rrbracket \\
& \quad \text{By interpretation} \\
& \llbracket \Gamma \vdash \text{pure } (N[\vec{x} := \vec{M}]) : \mathbf{KB} \rrbracket
\end{aligned}$$

2)

If  $\Gamma \vdash M : A$  and  $M \rightarrow_\beta N$ , then  $\llbracket \Gamma \vdash \text{pure } M : \mathbf{KA} \rrbracket = \llbracket \Gamma \vdash \text{pure } N : \mathbf{KA} \rrbracket$ .

If  $\Gamma \vdash M : A$  and  $M \rightarrow_\beta N$ , then  $\Gamma \vdash N : A$  by subject reduction.

By assumption  $\llbracket \Gamma \vdash M : A \rrbracket = \llbracket \Gamma \vdash N : A \rrbracket$ .

So  $\eta_{\llbracket A \rrbracket} \circ \llbracket \Gamma \vdash M : A \rrbracket = \eta_{\llbracket A \rrbracket} \circ \llbracket \Gamma \vdash N : A \rrbracket$ .

Hence  $\llbracket \Gamma \vdash \text{pure } M : \mathbf{KA} \rrbracket = \llbracket \Gamma \vdash \text{pure } N : \mathbf{KA} \rrbracket$ .

ii) For  $\eta$ -reduction.

1)  $\llbracket \vdash \text{let pure } \_ = \_ \text{ in } N : \mathbf{KA} \rrbracket = \llbracket \vdash \text{pure } N : \mathbf{KA} \rrbracket$ .

$$\begin{aligned}
& \llbracket \vdash \text{let pure } \_ = \_ \text{ in } N : \mathbf{KA} \rrbracket = & \text{By interpretation} \\
& \mathcal{K}(\llbracket N \rrbracket) \circ \eta_{\mathbf{1}} = & \text{Naturality for } \eta \\
& \eta_{\llbracket A \rrbracket} \circ \llbracket N \rrbracket = & \text{By interpretation} \\
& \llbracket \vdash \text{pure } N : \mathbf{KA} \rrbracket
\end{aligned}$$

2)

If  $\Gamma \vdash M : A$  and  $M \rightarrow_\eta N$ , then  $\llbracket \Gamma \vdash \text{pure } M : \mathbf{KA} \rrbracket = \llbracket \Gamma \vdash \text{pure } N : \mathbf{KA} \rrbracket$ .

Similar to case with  $\beta$ -reduction.

□

□

## 6.2 Completeness

## 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.