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

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

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

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

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

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[1] for calculations with container types like `Functor` and `Applicative` [2]:

---

[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]

```
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.

λK 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 $\to$ and $\wedge$ (we call that calculus $\mathrm{NIEL}^-_{\wedge,\to}$):

Definition 2. *Natural deduction $NIEL^-_{\wedge,\to}$ for IEL^- with $\to$ and $\wedge$:*

$$\frac{}{\Gamma, \alpha \vdash A} \; ax$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to_i \qquad\qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B} \to_i$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_i \qquad\qquad \frac{\Gamma \vdash A_1 \wedge A_2}{\Gamma \vdash A_i} \wedge_e, i \in \{1, 2\}$$

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

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

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

Proof. Induction on the derivation.

Let us consider cases with modality.

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

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

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

$$
\begin{array}{lll}
(1) & \bigwedge \Gamma \to \bigwedge_{i=1}^{n} \mathbf{K}A_i & \text{assumption} \\[2mm]
(2) & \bigwedge_{i=1}^{n} \mathbf{K}A_i \to \mathbf{K} \bigwedge_{i=1}^{n} A_i & \text{IEL theorem} \\[2mm]
(3) & \bigwedge \Gamma \to \mathbf{K} \bigwedge_{i=1}^{n} A_i & \text{from (1), (2) and transitivity} \\[2mm]
(4) & \bigwedge_{i=1}^{n} A_i \to B & \text{assumption} \\[2mm]
(5) & (\bigwedge_{i=1}^{n} A_i \to B) \to \mathbf{K}(\bigwedge_{i=1}^{n} A_i \to B) & \text{co-reflection} \\[2mm]
(6) & \mathbf{K}(\bigwedge_{i=1}^{n} A_i \to B) & \text{from (2), (3) and MP} \\[2mm]
(7) & \mathbf{K} \bigwedge_{i=1}^{n} A_i \to \mathbf{K}B & \text{from (6) and normality} \\[2mm]
(8) & \bigwedge \Gamma \to \mathbf{K}B & \text{from (3), (7) and transitivity}
\end{array}
$$

$\square$

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

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

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

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

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

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

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

$$\frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \to \beta} \to_i$$

$$\frac{\Gamma \vdash x : \alpha \qquad \Gamma \vdash y : \beta}{\Gamma \vdash (x, y) : \alpha \times \beta} \times_i$$

$$\frac{\Gamma \vdash x : \alpha}{\Gamma \vdash \mathbf{pure}\ x : \boldsymbol{K}\alpha} \boldsymbol{K}_I$$

$$\frac{\Gamma \vdash f : \alpha \to \beta \qquad \Gamma \vdash x : \alpha}{\Gamma \vdash fx : \beta} \to_e$$

$$\frac{\Gamma \vdash p : \alpha_1 \times \alpha_2}{\Gamma \vdash \pi_i p : \alpha_i} \times_e,\ i \in \{1, 2\}$$

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

$\mathbf{K}_I$-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.

Here are some examples of derivation trees.

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

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

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

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

Definition 6. *Set $FV(M)$ of free variables for arbitrary term M:*
  1) $FV(x) = \{x\}$;
  2) $FV(\lambda x.M) = FV(M) \setminus \{x\}$;
  3) $FV(MN) = FV(M) \cup FV(N)$;

4) $FV((M, N)) = FV(M) \cup FV(N)$;

5) $FV(\pi_i p) \subseteq FV(p)$, $i \in \{1, 2\}$;

6) $FV(pure\ M) = FV(M)$;

7) $FV(\textbf{let pure } \vec{N} = \vec{M} \textbf{ in } M) = \bigcup\limits_{i=1}^{n} FV(M)$, *where* $n = |\vec{M}|$.

Definition 7. *Substitution:*

1) $x[x := N] = N$, $x[y := N] = x$;

2) $(MN)[x := N] = M[x := N]N[x := N]$;

3) $(\lambda x.M)[x := N] = \lambda x.M[x := N]$;

4) $(M, N)[x := P] = (M[x := P], N[x := P])$;

5) $(\pi_i M)[x := P] = \pi_i(M[x := P])$, $i \in \{1, 2\}$;

6) $(\textbf{pure } M)[x := P] = \textbf{pure }(M[x := P])$;

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

In $\lambda\textbf{K}$ we have the following computational rules. We will define them for terms with **pure** or $\star$. Reduction rules for $(,)$ and π_i are described, for example, in [13].

Definition 8. *β-reduction rules for $\lambda\textbf{K}$.*

3 Basic lemmas

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

Definition 9. *The domain of a context Γ:*

Let $\Gamma = \{x_1 : \alpha_1, ..., x_n : \alpha_n\}$. *Then the domain of Γ, or $dom(\Gamma)$, is a set* $\{x_1, ..., x_n\}$.

Lemma 2. *If $\Gamma \vdash M : \alpha$, then $FV(M) \subseteq dom(\Gamma)$*

Proof. Induction on the derivation of $\Gamma \vdash M : \alpha$.

$\square$

Lemma 3. *Generation for $\lambda\textbf{K}$.*

1) $\Gamma \vdash \textbf{pure } M : \textbf{K}\alpha$ *implies that* $\Gamma \vdash M : \alpha$;

2) $\Gamma \vdash \textbf{let pure } \vec{N} = \vec{M} \textbf{ in } M : \textbf{K}B$ *implies that* $\Gamma \vdash \vec{M} : \textbf{K}\vec{A}$ *and* $\vec{N} : \vec{A} \vdash M : B$.

Proof.

Induction on the derivation of $\Gamma \vdash \textbf{pure } M : \textbf{K}\alpha$ and $\Gamma \vdash \textbf{let pure } \vec{N} = \vec{M} \textbf{ in } M : \textbf{K}B$ respectively. $\square$

The next one lemma allows that weakening structural rule is admissable.

Lemma 4. *Weakening for $\lambda\textbf{K}$.*

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

---

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

Proof.

Induction on derivation of $\Gamma \vdash M : \alpha$. Let us assume $\Gamma \subseteq \Delta$.

1) Let $\Gamma \vdash x : \alpha$, such that $\Gamma = \Delta, x : \alpha$ and $\Theta \subseteq \Gamma$. Let $\Sigma = \Theta \setminus \Gamma$, or, which is the same, $\Sigma = \Theta \setminus \Delta, x : \alpha$, then $\Sigma, \Delta, x : \alpha \vdash x : \alpha$, or, $\Theta \vdash x : \alpha$.

2) Let $\Gamma \vdash \mathbf{pure}\, M : \mathbf{K}\alpha$ and $\Gamma \subseteq \Theta$.

If $\Gamma \vdash \mathbf{pure}\, M : \mathbf{K}\alpha$, then $\Gamma \vdash M : \alpha$ by generation and, by hypothesis, $\Theta \vdash M : \alpha$, so $\Theta \vdash \mathbf{pure}\, M : \mathbf{K}\alpha$ by applying $\mathbf{K}_I$-rule.

3) Let $\Gamma \vdash M \star N : \mathbf{K}\beta$. So $\Gamma \vdash M : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \vdash N : \mathbf{K}\alpha$. By hypothesis $\Delta \vdash M : \mathbf{K}(\alpha \to \beta)$ and $\Delta \vdash N : \mathbf{K}\alpha$. Then $\Delta \vdash M \star N : \mathbf{K}\beta$.

$\square$

Lemma 5. *Considering for $\lambda\mathbf{K}$.*

If $\Gamma \vdash M : \alpha$, then $\Gamma \uparrow FV(M) \vdash M : \alpha$, where $\Gamma \uparrow FV(M)$ is a subcontext of Γ, such that $dom(\Gamma \uparrow FV(M)) = dom(\Gamma) \cap FV(M)$.

Proof. Induction by derivation. We consider the base of induction and the case with $\mathbf{K}_{app}$. The rest cases are proven by the same way.

1) Let $\Gamma \vdash x : \alpha$, where $\Gamma = \Delta, x : \alpha$, $x \in \mathbb{V}$.

$FV(x) = \{x\}$, then $dom(\Gamma) \cap \{x\} = \{x\}$. So $(\Delta, x : \alpha) \uparrow FV(x) = \{x : \alpha\}$, then $x : \alpha \vdash x : \alpha$ by axiom.

2) Let $\Gamma \vdash M \star N : \mathbf{K}\beta$.

By generation $\Gamma \vdash M : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \vdash N : \mathbf{K}\alpha$.

By induction hypothesis $\Gamma \uparrow FV(M) \vdash M : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \uparrow FV(N) \vdash N : \mathbf{K}\beta$.

But $\Gamma \uparrow FV(M) \subseteq \Gamma \uparrow FV(M \star N)$ and $\Gamma \uparrow FV(N) \subseteq \Gamma \uparrow FV(M \star N)$, so $\Gamma \uparrow FV(M \star N) \vdash M : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \uparrow FV(M \star N) \vdash N : \mathbf{K}\alpha$ by weakening.

Then $\Gamma \uparrow FV(M \star N) \vdash M \star N : \mathbf{K}\beta$ by $\mathbf{K}_{app}$.

$\square$

Lemma 6. *If $\Gamma, x : \alpha \vdash M : \beta$ and $\Gamma \vdash N : \alpha$, then $\Gamma \vdash (M[x := N]) : \beta$*

Proof.

1) Let $\Gamma, x : \alpha \vdash \mathbf{pure}\, M : \mathbf{K}\beta$ and $\Gamma \vdash N : \alpha$.

If $\Gamma, x : \alpha \vdash \mathbf{pure}\, M : \mathbf{K}\beta$, then, by generation, $\Gamma, x : A \vdash M : \beta$. So, by induction hypothesis, $\Gamma \vdash (M[x := N]) : \beta$, then $\Gamma \vdash \mathbf{pure}\, (M[x := N]) : \mathbf{K}\beta$ by $\mathbf{K}_I$, but $\mathbf{pure}\, (M[x := N]) = (\mathbf{pure}\, M)(M[x := N])$ by substitution definition, so $\Gamma \vdash (\mathbf{pure}\, M)(M[x := N]) : \mathbf{K}\beta$

2) Let $\Gamma, x : \gamma \vdash M \star N : \mathbf{K}\beta$, and $\Gamma \vdash y : \gamma$.

So, by generation, $\Gamma, x : \gamma \vdash M : \mathbf{K}(\alpha \to \beta)$ and $\Gamma, x : \gamma \vdash N : \mathbf{K}\alpha$.

Hence $\Gamma, x : \gamma \vdash (M[x := y]) : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \vdash (N[x := y]) : \mathbf{K}\alpha$ by hypothesis.

So $\Gamma \vdash (M[x := y]) \star (N[x := y]) : \mathbf{K}\beta$, or, $\Gamma \vdash (M \star N)([x := y]) : \mathbf{K}\beta$.

$\square$

Theorem 1. *Subject reduction*

Let $\Gamma \vdash M : \alpha$ and $M \twoheadrightarrow_\beta N$, then $\Gamma \vdash N : \alpha$

We consider cases with reduction rules which are applicative laws. The general statement for $\twoheadrightarrow_\beta$ follows from transitivity of multi-step β-reduction.

Proof.

1) Let $\Gamma \vdash \mathbf{pure}\,(\lambda x.x) \star M : \mathbf{K}\alpha$. Then $\Gamma \vdash \mathbf{pure}\,(\lambda x.x) : \mathbf{K}(\alpha \to \alpha)$ and $\Gamma \vdash M : \mathbf{K}\alpha$ by generation. Then $\Gamma \vdash M : \mathbf{K}\alpha$ trivially.

2) Let $\Gamma \vdash \mathbf{pure}\,(\lambda fgx.f(gx)) \star M \star N \star P : \mathbf{K}\gamma$.
Then $\Gamma \vdash \mathbf{pure}\,(\lambda fgx.f(gx)) : \mathbf{K}((\beta \to \gamma) \to (\alpha \to \beta) \to \alpha \to \gamma)$, $\Gamma \vdash M : \mathbf{K}(\beta \to \gamma)$, $\Gamma \vdash N : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \vdash P : \mathbf{K}\alpha$ by generation.
If $\Gamma \vdash N : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \vdash P : \mathbf{K}\alpha$, then $\Gamma \vdash N \star P : \mathbf{K}\beta$ by $\mathbf{K}_{app}$.
Hence, if $\Gamma \vdash M : \mathbf{K}(\beta \to \gamma)$, then $\Gamma \vdash M \star (N \star P) : \mathbf{K}\gamma$ by $\mathbf{K}_{app}$.

3) Let $\Gamma \vdash (\mathbf{pure}\,M) \star (\mathbf{pure}\,N) : \mathbf{K}\beta$. Then $\Gamma \vdash \mathbf{pure}\,M : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \vdash \mathbf{pure}\,N : \mathbf{K}\alpha$ by generation. Moreover, $\Gamma \vdash M : \alpha \to \beta$ and $\Gamma \vdash N : \alpha$.
Then $\Gamma \vdash MN : \beta$ by application.
Hence, $\Gamma \vdash \mathbf{pure}\,(MN) : \mathbf{K}\beta$ by K_I.

4) Let $\Gamma \vdash M \star (\mathbf{pure}\,N) : \mathbf{K}\beta$.
Then $\Gamma \vdash M : \mathbf{K}(\alpha \to \beta)$ and $\Gamma \vdash \mathbf{pure}\,N : \mathbf{K}\alpha$.
Moreover, $\Gamma \vdash N : \alpha$ by generation.
Let $\Gamma, f : \alpha \to \beta \vdash f : \alpha \to \beta$ and $\Gamma, f : \alpha \to \beta \vdash N : \alpha$ by weakening.
So $\Gamma, f : \alpha \to \beta \vdash fN : \beta$ by application, so $\Gamma \vdash \lambda f.fN : (\alpha \to \beta) \to \beta$ by abstraction.
Then $\Gamma \vdash \mathbf{pure}\,(\lambda f.fN) : \mathbf{K}((\alpha \to \beta) \to \beta)$ by $\mathbf{K}_I$.
Hence, $\Gamma \vdash \mathbf{pure}\,(\lambda f.fN) \star M : \mathbf{K}\beta$.

$\square$

4 Strong normalization

We modify and apply Tait's technique of logical relation for modalities. Strong normalization proof with Tait's method for simply typed lambda calculus is described here [13].

Theorem 2. *Let $M \in \Lambda_{\mathbf{K}}$, then any sequence of reduction $M \to_\beta M_1 \ldots$ terminates.*

Proof. We build the smallest of subset of strongly normalazing terms of modal types and show that an arbitrary term belongs to this subset.

Definition 10. *The set of strongly computable terms of type $\phi \in \mathbb{T}_{\mathbf{K}}$, SC_ϕ:*

- *Let $\phi = \mathbf{K}\alpha$ and $\alpha \in \mathbb{T}$, then:*

$$SC_{\mathbf{K}\alpha} = \{M : \mathbf{K}\alpha \mid M \text{ is strongly normalizing}\} \qquad (2)$$

- *Let $\phi = \mathbf{K}(\tau \to \psi)$ and $\tau, \psi \in \mathbb{T}_{\mathbf{K}}$, then:*

$$SC_{\mathbf{K}(\tau \to \psi)} = \{M : \mathbf{K}(\tau \to \psi) \mid \forall N \in SC_{\mathbf{K}\tau}, M \star N \in SC_{\mathbf{K}\psi}\} \qquad (3)$$

- Let $\phi = \boldsymbol{K}(\tau_1 \times \tau_2)$ and $\tau_1, \tau_2 \in \mathbb{T}_{\boldsymbol{K}}$, then:

$$SC_{\boldsymbol{K}(\tau_1 \times \tau_2)} = \{P : \boldsymbol{K}(\tau_1 \times \tau_2) \,|\, \textbf{pure}\,(\lambda x.\pi_i x) \star P \in SC_{\boldsymbol{K}\tau_i}, i \in \{1,2\}\} \quad (4)$$

Lemma 7.

  If $M \in SC_\alpha$, then M is strongly normalizing.

Proof.

  1) If $M \in SC_{\boldsymbol{K}\alpha}$ and $\alpha \in \mathbb{T}$, then M is strongly normalizing by the definition of $SC_{\boldsymbol{K}\alpha}$.

  2) Let $M \in SC_{\boldsymbol{K}(\tau \to \psi)}$, so by every $N \in SC_{\boldsymbol{K}\tau}$, $M \star N \in SC_{\boldsymbol{K}\psi}$, which is strongly normalizing by hypothesis. So M is strongly normalizing.

  3) Let $M \in SC_{\boldsymbol{K}(\tau_1 \times \tau_2)}$, so $\textbf{pure}\,(\lambda x.\pi_i x) \star M \in SC_{\boldsymbol{K}\tau_i}$, $i \in \{1,2\}$, which are strongly normalizing. So M is strongly normalizing.

$\square$

Lemma 8.

  Let $M \to_\beta M'$ and $M \in SC_\alpha$, then $M' \in SC_\alpha$.

Proof.

  1) Let $M \to_\beta M'$ and $M \in SC_{\boldsymbol{K}\alpha}$, where $\alpha \in \mathbb{T}$.

  M has the longest reduction path (which we denote as $p(M)$). So $p(M') < p(M)$, then $M' \in SC_{\boldsymbol{K}\alpha}$.

  2) Let $M \in SC_{\boldsymbol{K}(\alpha \to \beta)}$ and $M \to_\beta M'$. Let $N \in SC_{\boldsymbol{K}\alpha}$. So $M \star N \in SC_{\boldsymbol{K}\beta}$.

  If $M \to_\beta M'$, then $M \star N \to_\beta M' \star N$ by reduction rule, so $M' \star N \in SC_{\boldsymbol{K}\beta}$ and $M' \in SC_{\boldsymbol{K}(\alpha \to \beta)}$ by hypothesis.

  3) Let $M \in SC_{\boldsymbol{K}(\tau_1 \times \tau_2)}$ and $M \to_\beta M'$.

  So $\textbf{pure}\,(\lambda x.\pi_i x) \star M \to_\beta \textbf{pure}\,(\lambda x.\pi_i x) \star M'$, $i \in \{1,2\}$ by reduction rule. So $\textbf{pure}\,(\lambda x.\pi_i x) \star M' \in SC_{\boldsymbol{K}\tau_i}$ and $M' \in SC_{\boldsymbol{K}(\tau_1 \times \tau_2)}$.

$\square$

Definition 11. *Neutral term:*

  We define a term M to be neutral if it has of the next forms:

  1) $M = x$, where $x \in \mathbb{V}$;

  2) $M = (PQ)$;

  3) $M = \pi_i M$, $i \in \{1,2\}$;

  4) $M = P \star Q$;

  5) If M is a neutral, then $\textbf{pure}\,M$ is a neutral.

Lemma 9. *Let $M \to_\beta M'$ and $M' \in SC_\alpha$ for every one-step reduction. So if M' is a neutral, then $M \in SC_\alpha$.*

Proof.

  Simple induction on the structure of M'.

$\square$

Lemma 10.

  Let $x_1 : \phi_1, \ldots, x_n : \phi_n \vdash M : \phi$ and for all $i \in \{1, \ldots, n\}$, $N_i \in SC_{\phi_i}$, then $(M[x_1 := N_1, \ldots, x_n := N_n]) \in SC_\phi$.

Proof.

1) If ϕ is an atomic and M is a variable, then this condition holds straightforwardly.

2) Let $\Gamma = \{x_1 : \phi_1, \ldots, x_n : \phi_n\}$, $\Gamma \vdash \mathbf{pure}\, M : \mathbf{K}\alpha$ and for all $i \in \{1, \ldots, n\}, N_i \in SC_{\phi_i}$.

Then by $\Gamma \vdash M : \alpha$ by generation and $(M[x_1 := N_1, \ldots, x_n := N_n]) \in SC_\alpha$ by induction hypothesis.

Hence, $\Gamma \vdash \mathbf{pure}\, M : \mathbf{K}\alpha$ and $(\mathbf{pure}\, M([x_1 := N_1, \ldots, x_n := N_n])) \in SC_{\mathbf{K}\alpha}$ by definition of $SC_{\mathbf{K}\alpha}$.

3) Let $\Gamma = \{x_1 : \phi_1, \ldots, x_n : \phi_n\}$, $\Gamma : \phi_n \vdash M \star P : \mathbf{K}\beta$ and forall $i \in \{1, \ldots, n\}, N_i \in SC_{\phi_i}$.

Then $\Gamma \vdash M : \mathbf{K}(\alpha \to \beta)$, $\Gamma \vdash P : \mathbf{K}\alpha$ by generation.

But by induction hypothesis $M[x_1 := N_1, \ldots, x_n := N_n] \in SC_{\mathbf{K}(\alpha \to \beta)}$ and $P[x_1 := N_1, \ldots, x_n := N_n] \in SC_{\mathbf{K}\alpha}$.

Then, by definition of $SC_{\mathbf{K}\beta}$, $((M[x_1 := N_1, \ldots, x_n := N_n]) \star (P[x_1 := N_1, \ldots, x_n := N_n])) \in SC_{\mathbf{K}\beta}$, i.e. $(M \star N([x_1 := N_1, \ldots, x_n := N_n])) \in SC_{\mathbf{K}\beta}$.
□

Corollary 1.
If $\vdash M : \alpha$, then M is strongly normalizing.

Proof. $M \in SC_\alpha$ by Lemma 10, so M is strongly normalizing.

□

□

5 Confluence

In the confluence proof (below) we treat the cases with **pure** and $\star$ similar to [15] [18].

Definition 12. *Alphabet for the labelled terms:*
variables: $x, y, z, x_1, y_1, z_1, \ldots$;
lambdas: $\lambda, \lambda_0, \lambda_1, \lambda_2, \ldots$;
constructors for an applicative functor: **pure** *, $\star$;*
parentheses (,).

Definition 13. *The set of labelled terms $\Lambda'_{\mathbf{K}}$ inductively defined as a set of words on the alphabet described above:*
1) $x \in \Lambda'$;
2) If $M \in \Lambda'_{\mathbf{K}}$, then $(\lambda x.M) \in \Lambda'_{\mathbf{K}}$;
3) If $M, N \in \Lambda'_{\mathbf{K}}$, then $(MN) \in \Lambda'_{\mathbf{K}}$;
4) If $M \in \Lambda'_{\mathbf{K}}$, then $\mathbf{pure}\, M \in \Lambda'_{\mathbf{K}}$;
5) If $M, N \in \Lambda'_{\mathbf{K}}$, then $M \star N \in \Lambda'_{\mathbf{K}}$;
6) If $M, N \in \Lambda'_{\mathbf{K}}$, then for all $i \in \mathbb{N}$, $((\lambda_i x.M)N) \in \Lambda'_{\mathbf{K}}$.

Definition 14. *Erasing map*
Erasing map is a map $|.| : \Lambda'_{\mathbf{K}} \to \Lambda_{\mathbf{K}}$, such that:
1) $|x| = x$;
2) $|(\lambda x.M)| = \lambda x.|M|$;
3) $|(MN)| = |M||N|$;

9

4) $|(\mathbf{pure}\ M)| = \mathbf{pure}\ |M|;$

5) $|M \star N| = |M| \star |N|;$

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

Example 1.

$|\mathbf{pure}\ ((\lambda_i x.M)N) \star P| = \mathbf{pure}\ (\lambda x.|M|)|N|) \star |P|$

Definition 15. *Substitution for* $\Lambda'_{\boldsymbol{K}}$:

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

5) $(M \star N)[x := P] = (M[x := P]) \star (N[x := P]);$

6) $(\lambda_i x.M)N[y := P] = (\lambda_i x.M[y := P])(N[y := P]).$

Definition 16. *One-step reduction* $\to_{\beta'}$ *for* $\Lambda'_{\boldsymbol{K}}$:

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

2) $\mathbf{pure}\ (\lambda x.x) \star M \to_{\beta'} M;$

3) $\mathbf{pure}\ (\lambda fgx.f(gx)) \star M \star N \star P \to_{\beta'} M \star (N \star P);$

4) $(\mathbf{pure}\ M) \star (\mathbf{pure}\ N) \to_{\beta'} \mathbf{pure}\ (MN);$

5) $M \star (\mathbf{pure}\ N) \to_{\beta'} \mathbf{pure}\ (\lambda f.fN) \star M;$

6) $(\lambda_i x.M)N \to_{\beta'} M[x := N].$

Multi-step reduction $\twoheadrightarrow_{\beta'}$ *is a reflexive-transitive closure of* $\to_{\beta'}$.

Definition 17. *Let us define a map* $\phi : \Lambda'_{\boldsymbol{K}} \to \Lambda_{\boldsymbol{K}}$ *inductively as follows:*

1) $\phi(x) = x;$

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

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

4) $\phi(\mathbf{pure}\ M) = \mathbf{pure}\ (\phi(M));$

5) $\phi(M \star N) = \phi(M) \star \phi(N);$

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

Example 2.

$\phi(\mathbf{pure}\ ((\lambda_i x.M)N) \star P) = \mathbf{pure}\ (\phi(M)[x := \phi(N)]) \star \phi(P)$

Lemma 11.

1) Let $M, N \in \Lambda'_{\boldsymbol{K}}$ *and* $|M| \twoheadrightarrow_{\beta} |N|$, *then* $M \twoheadrightarrow_{\beta'} N$.

2) Let $M, N \in \Lambda'_{\boldsymbol{K}}$ *and* $M \twoheadrightarrow_{\beta'} N$, *then* $|M| \twoheadrightarrow_{\beta} |N|$.

Proof.

Induction on the generation of $\twoheadrightarrow_{\beta}$ $(\twoheadrightarrow_{\beta'})$.

1) Let us consider homomorphism rule. The rest applicative reduction rules are considered similary.

Let $(\mathbf{pure}\ M') \star (\mathbf{pure}\ N'), \mathbf{pure}\ (M'N') \in \Lambda'_{\mathbf{K}}$.

So $|(\mathbf{pure}\ M') \star (\mathbf{pure}\ N')| = (\mathbf{pure}\ |M'|) \star (\mathbf{pure}\ |N'|)$ and $|\mathbf{pure}\ (M'N')| = \mathbf{pure}\ (|M'||N'|)$.

By reduction rule, $(\mathbf{pure}\ |M'|) \star (\mathbf{pure}\ |N'|) \to_{\beta} \mathbf{pure}\ (|M'||N'|)$.

But $(\mathbf{pure}\ M') \star (\mathbf{pure}\ N') \to_{\beta'} \mathbf{pure}\ (M'N')$ by reduction rule for $\to_{\beta'}$.

2) Let us consider interchange rule.

Let $M \star (\mathbf{pure}\ N), \mathbf{pure}\ (\lambda f.fN) \star M \in \Lambda'_{\mathbf{K}}$ and $M \star (\mathbf{pure}\ N) \to_{\beta'} \mathbf{pure}\ (\lambda f.fN) \star M$.

But $|M \star (\textbf{pure } N)| = |M| \star (\textbf{pure } |N|)$ and $|\textbf{pure } (\lambda f.fN) \star M| = \textbf{pure } (\lambda f.f|N|) \star |M|$.

So $|M| \star (\textbf{pure } |N|) \to_\beta \textbf{pure } (\lambda f.f|N|) \star |M|$ by β-reduction rule.

It is easy to see, that the statement for $\twoheadrightarrow_{\beta'}$ and $\twoheadrightarrow_\beta$ immedeatly follows from transitivity of multi-step rediction for labelled terms and for usual terms respectively.

$\square$

Lemma 12.
$\phi(M[x := N]) = \phi(M)[x := \phi(N)]$.

Proof. Induction on M.

1) Let $M = x$. Then $\phi(x[x := N]) = \phi(N)$.
On the other hand, $\phi(x)[x := \phi(N)] = x[x := \phi(N)] = \phi(N)$.
So $\phi(x[x := N]) = \phi(x)[x := \phi(N)]$.

2) Let $M = y$ and $y \neq x$. Then $\phi(y[x := N]) = \phi(y) = y$.
But $\phi(y)[x := \phi(N)] = y[x := \phi(N)] = y$.
Therefore $\phi(y[x := N]) = \phi(y)[x := \phi(N)]$.

3) Let $M = \textbf{pure } M'$. Then $\phi(\textbf{pure } M'[x := N]) = \textbf{pure } \phi(M'[x := N])$.
By hypothesis, $\textbf{pure } (\phi(M'[x := N])) = \textbf{pure } (\phi(M')[x := \phi(N)])$, which is $(\textbf{pure } \phi(M'))[x := \phi(N)]$ by substitution definition.

4) Let $M = M' \star N'$. So $\phi((M' \star N')[x := N])) = \phi(M'[x := N] \star N'[x := N])$.
By definition of ϕ,
$\phi(M'[x := N] \star N'[x := N]) = \phi(M'[x := N]) \star \phi(N'[x := N])$.
But by induction hypothesis,
$\phi(M'[x := N]) = \phi(M')[x := \phi(N)]$ and
$\phi(N'[x := N]) = \phi(N')[x := \phi(N)]$.
Hence,
$\phi(M'[x := N]) \star \phi(N'[x := N]) = \phi(M')[x := \phi(N)] \star \phi(N')[x := \phi(N)]$.
So,
$\phi(M')[x := \phi(N)] \star \phi(N')[x := \phi(N)] = (\phi(M') \star \phi(N'))[x := \phi(N)]$.
And by definition of ϕ, $(\phi(M') \star \phi(N'))[x := \phi(N)] = \phi(M' \star N')[x := \phi(N)]$.

$\square$

Lemma 13.
Let $M, N \in \Lambda_{\textbf{K}}'$ and $M \twoheadrightarrow_{\beta'} N$, then $\phi(M) \twoheadrightarrow_\beta \phi(N)$.

Proof.

1) Let $\textbf{pure } (\lambda x.x) \star M, M \in \Lambda_{\textbf{K}}'$ and $\textbf{pure } (\lambda x.x) \star M \to_{\beta'} M$.
But $\phi(\textbf{pure } (\lambda x.x) \star M) = \textbf{pure } (\lambda x.x) \star \phi(M)$.
So $\textbf{pure } (\lambda x.x) \star \phi(M) \to_\beta \phi(M)$ by β-reduction rule.

2) Let $\textbf{pure } (\lambda fgx.f(gx)) \star M \star N \star P, M \star (N \star P) \in \Lambda_{\textbf{K}}'$ and $\textbf{pure } (\lambda fgx.f(gx)) \star M \star N \star P \to_{\beta'} M \star (N \star P)$.
By the definition of ϕ:
$\phi(\textbf{pure } (\lambda fgx.f(gx)) \star M \star N \star P) = \textbf{pure } (\lambda fgx.f(gx)) \star \phi(M) \star \phi(N) \star \phi(P)$;
$M \star (N \star P) = \phi(M) \star (\phi(N) \star \phi(P))$.

Hence, **pure** $(\lambda fgx.f(gx)) \star \phi(M) \star \phi(N) \star \phi(P) \to_\beta \phi(M) \star (\phi(N) \star \phi(P))$ by β-reduction rule.

3) Let $(\textbf{pure } M)\star(\textbf{pure } N), \textbf{pure } (MN) \in \Lambda_{\textbf{K}}^{'}$ and $(\textbf{pure } M)\star(\textbf{pure } N) \to_\beta$ **pure** (MN).

By the definition of ϕ:

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

$\phi(\textbf{pure } (MN)) = \textbf{pure } (\phi(M)\phi(N))$.

So, by reduction rule, $(\textbf{pure } \phi(M)) \star (\textbf{pure } \phi(N)) \to_\beta \textbf{pure } (\phi(M)\phi(N))$.

4) Let $M \star (\textbf{pure }), \textbf{pure } (\lambda f.fN)\star M$ and $M \star (\textbf{pure } N) \to_{\beta'} (\lambda f.fN)\star M$.

$\phi(M \star (\textbf{pure } N)) = \phi(M) \star (\textbf{pure } \phi(N))$

$\phi((\lambda f.fN) \star M) = (\lambda f.f\phi(N)) \star \phi(M)$.

So, $\phi(M) \star (\textbf{pure } \phi(N)) \to_\beta \textbf{pure } (\lambda f.f\phi(N)) \star \phi(M)$. $\qquad\square$
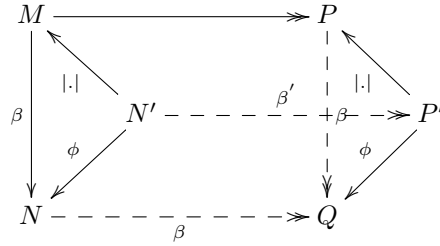
Lemma 14.

Let $M \in \Lambda_{\textbf{K}}^{'}$. Then $|M| \twoheadrightarrow_\beta \phi(M)$.

Proof. Induction on the structure of M.

$\qquad\square$

Lemma 15. *Strip lemma.*

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

Proof. Proof is similar to [15] [18]. We build the following diagram

$\qquad\square$

which is commutes by lemmas 11 – 14.

Theorem 3. *Confluence.*

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.

By unfolding $M \twoheadrightarrow_\beta N$ as the sequence of one-step reductions $M \to_\beta M_1 \to_\beta \ldots \to_\beta M_n \to_\beta N$ and applying strip lemma on every step.

$\qquad\square$

6   Acknowledgement.

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.