Normalization for Fitch-style Modal Calculi (Draft)

Nachiappan Valliappan[1], Fabian Ruch, and Carlos Tomé Cortiñas[1]

[1] Chalmers University of Technology

23 November 2021

Abstract. Fitch-style modal lambda calculi enable programming with necessity modalities in a typed lambda calculus by extending the typing context with a delimiting operator that is denoted by a lock. The addition of locks simplifies the formulation of typing rules for calculi that incorporate different modal axioms, but each variant demands different, tedious and seemingly ad hoc syntactic lemmas to prove normalization. In this work, we take a semantic approach to normalization, called normalization by evaluation (NbE), by leveraging the possible-world semantics of Fitch-style calculi to yield a more modular approach to normalization. We show that NbE models can be constructed for calculi that incorporate the K, T and 4 axioms of modal logic, as suitable instantiations of the possible-world semantics. In addition to existing results that handle β-equivalence, our normalization result also considers η-equivalence for these calculi. Our key results have been mechanized in the proof assistant AGDA. Finally, we showcase several consequences of normalization for proving meta-theoretic properties of Fitch-style calculi as well as programming-language applications based on different interpretations of the necessity modality.

Keywords: Fitch-style lambda calculi · Possible-world semantics · Normalization by Evaluation.

1 Introduction

In type systems, a *modality* can be broadly construed as a unary type constructor with certain properties. Type systems with modalities have found a wide range of applications in programming languages to capture and specify properties of a program in its type. In this work, we study typed lambda calculi equipped with a *necessity* modality (denoted by $\Box$) formulated in the so-called Fitch style.

The necessity modality originates from modal logic, where the most basic intuitionistic modal logic IK (for "intuitionistic" and "Kripke") extends intuitionistic propositional logic with a unary connective $\Box$, the *necessitation rule* (if $\cdot \vdash A$ then $\Gamma \vdash \Box A$) and the K *axiom* ($\Box(A \Rightarrow B) \Rightarrow \Box A \Rightarrow \Box B$). With the addition of further modal axioms T ($\Box A \Rightarrow A$) and 4 ($\Box A \Rightarrow \Box\Box A$) to IK, we obtain richer logics IT (adding axiom T), IK4 (adding axiom 4), and IS4 (adding both T and 4). Type systems with necessity modalities based on IK and

IS4 have found applications in partial evaluation and staged computation [11], information-flow control [23], and recovering purity in an effectful language [8]. While type systems based on IT and IK4 do not seem to have any prior known programming applications, they are nevertheless interesting as objects of study that extend IK towards IS4.

Fitch-style modal lambda calculi [6, 9] feature necessity modalities in a typed lambda calculus by extending the typing context with a delimiting "lock" operator (denoted by 🔒). In this paper, we consider the family of Fitch-style modal lambda calculi that correspond to the logics IK, IT, IK4, and IS4. These calculi extend the simply-typed lambda calculus (STLC) with a type constructor $\square$, along with introduction and elimination rules for $\square$ types formulated using the 🔒 operator. For instance, the calculus λ_{IK}, which corresponds to the logic IK, extends STLC with Rules $\square$-Intro and $\square$-Elim/λ_{IK}, as summarized in Fig. 1. The rules for λ-abstraction and function application are formulated in the usual way—but note the modified variable rule Var!

$$\text{Ty} \quad A ::= \ldots \mid \square A \qquad\qquad \text{Ctx} \quad \Gamma ::= \cdot \mid \Gamma, x : A \mid \Gamma, \text{🔒}$$

$$\overset{\textsc{Var}}{\dfrac{}{\Gamma, x : A, \Gamma' \vdash x : A}}\ \text{🔒} \notin \Gamma' \qquad \overset{\square\text{-}\textsc{Intro}}{\dfrac{\Gamma, \text{🔒} \vdash t : A}{\Gamma \vdash \mathsf{box}\, t : \square A}} \qquad \overset{\square\text{-}\textsc{Elim}/\lambda_{\mathrm{IK}}}{\dfrac{\Gamma \vdash t : \square A}{\Gamma, \text{🔒}, \Gamma' \vdash \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, t : A}}\ \text{🔒} \notin \Gamma'$$

Fig. 1. Typing rules for λ_{IK} (omitting λ-abstraction and application)

The equivalence of terms in STLC is extended by Fitch-style calculi with the following rules for $\square$ types, where the former states the β- (or computational) equivalence, and the latter states a type-directed η- (or extensional) equivalence.

$$\overset{\square\text{-}\beta}{\mathsf{unbox}\,(\mathsf{box}\, t) \sim t} \qquad\qquad \overset{\square\text{-}\eta}{\dfrac{\Gamma \vdash t : \square A}{t \sim \mathsf{box}\,(\mathsf{unbox}\, t)}}$$

We are interested in the problem of normalizing terms with respect to these equivalences. Traditionally, terms in a calculus are normalized by rewriting them using rewrite rules formulated from these equivalences, and a term is said to be in *normal form* when it cannot be rewritten further. For example, we may formulate a rewrite rule $\mathsf{unbox}\,(\mathsf{box}\, t) \mapsto t$ by orienting the $\square$-β equivalence from left to right. This naive approach to formulating a rewrite rule, however, is insufficient for the $\square$-η rule since normalizing with a rewrite rule $t \mapsto \mathsf{box}\,(\mathsf{unbox}\, t)$ (for $\Gamma \vdash t : \square A$) does not terminate as it can be applied infinitely many times. It is presumably for this reason that existing normalization results [9] for some of these calculi only consider β-equivalence.

While it may be possible to carefully formulate a more complex set of rewrite rules that take the context of application into consideration to guarantee termination (as done, for example, by Jay and Ghani [17] for function and product types), the situation is further complicated for Fitch-style calculi by the fact that we must repeat such syntactic rewriting arguments separately for each calculus under consideration. The calculi λ_{IT}, λ_{IK4}, and λ_{IS4} differ from λ_{IK} only in the $\Box$-elimination rule, as summarized in Fig. 2. In spite of having identical syntax and

$$\Box\text{-}\textsc{Elim}/\lambda_{\mathrm{IT}}\quad \frac{\Gamma \vdash t : \Box A}{\Gamma, \Gamma' \vdash \mathsf{unbox}_{\lambda_{\mathrm{IT}}}\, t : A}\ \#_{🔒}(\Gamma') \leq 1 \qquad\qquad \Box\text{-}\textsc{Elim}/\lambda_{\mathrm{IK4}}\quad \frac{\Gamma \vdash t : \Box A}{\Gamma, 🔒, \Gamma' \vdash \mathsf{unbox}_{\lambda_{\mathrm{IK4}}}\, t : A}$$

$$\Box\text{-}\textsc{Elim}/\lambda_{\mathrm{IS4}}\quad \frac{\Gamma \vdash t : \Box A}{\Gamma, \Gamma' \vdash \mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t : A}$$

Fig. 2. $\Box$-elimination rules for λ_{IT}, λ_{IK4}, and λ_{IS4}

term equivalences, each calculus demands different, tedious and seemingly ad hoc syntactic renaming lemmas [9, Lemmas 4.1 and 5.1] to prove normalization.

In this paper, we take a semantic approach to normalization, called normalization by evaluation (NbE) [5]. NbE bypasses rewriting entirely, and instead normalizes terms by evaluating them in a suitable semantic model and then reifying values in the model as normal forms. For Fitch-style calculi, NbE can be developed by leveraging their possible-world semantics. To this end, we identify the parameters of the possible-world semantics for the calculi under consideration, and show that NbE models can be constructed by instantiating those parameters. The NbE approach exploits the semantic overlap of the Fitch-style calculi in the possible-world semantics and isolates their differences to a specific parameter that determines the modal fragment, thus enabling the reuse of the evaluation machinery and many lemmas proved in the process.

In Section 2, we begin by providing a brief overview of the main idea underlying this paper. We discuss the uniform interpretation of types for four Fitch-style calculi (λ_{IK}, λ_{IT}, λ_{IK4} and λ_{IS4}) in possible-world models and outline how NbE models can be constructed as instances. The *reification* mechanism that enables NbE is performed alike for all four calculi. In Section 3, we construct an NbE model for λ_{IK} that yields a correct normalization algorithm, and then show how NbE models can also be constructed for λ_{IS4}, and for λ_{IT} and λ_{IK4} by slightly varying the instantiation. The calculus λ_{IK} and its normalization algorithm have been implemented and verified correct in Agda [2], and we have also mechanized most of our results for λ_{IS4}.

NbE models and proofs of normalization more general have several useful consequences for term calculi. In Section 4, we show how NbE models and the accompanying normalization algorithm can be used to prove meta-theoretic properties of Fitch-style calculi including completeness, decidability, and some standard results in modal logic in a *constructive* manner. In Section 5, we discuss applications of our development to specific interpretations of the necessity modality in programming languages, and show how application-specific theorems that typically require semantic intervention can be proved syntactically. More specifically, we show that theorems in staging, information-flow control and imperative calculi can be proved syntactically by inspection of normal forms.

2 Main Idea

The main idea underlying this paper is that normalization can be achieved in a modular fashion for Fitch-style calculi by constructing NbE models as instances of their possible-world semantics. In this section, we observe that Fitch-style calculi can be interpreted in the possible-world semantics for intuitionistic modal logic with a minor refinement that accommodates the 🔒 operator, and give a brief overview of how we construct NbE models as instances.

Possible-World Semantics Possible-world semantics for intuitionistic modal logic [7] is parameterized by a *frame* F and a *valuation* V_ι. A frame F is a triple (W, R_i, R_m) that consists of a type W (for "worlds"), and two binary relations R_i (for "intuitionistic accessibility") and R_m (for "modal accessibility") on worlds that are required to satisfy certain conditions. An element $w : W$ can be thought of as a representation of the "knowledge state" about some "possible world" at a certain point in time, $w\ R_i\ w'$ as representing an increase in knowledge from w to w', and $w\ R_m\ v$ as specifying accessibility of worlds from one another. The valuation V_ι, on the other hand, interprets the base type ι at a given world w as a type $V_\iota(w)$, a value of which can be thought of as the evidence of a proposition ι at the world w.

Following the work by Božić and Došen [7], we interpret the types in all the Fitch-style calculi alike as follows, for a given world w:

$$\begin{aligned}
[\![\iota]\!]_w &= V_\iota(w) \\
[\![A \Rightarrow B]\!]_w &= \forall w'.\, w\ R_i\ w' \to [\![A]\!]_{w'} \to [\![B]\!]_{w'} \\
[\![\Box A]\!]_w &= \forall v.\, w\ R_m\ v \to [\![A]\!]_v
\end{aligned}$$

The base type ι is interpreted using the valuation V_ι, and the function type $A \Rightarrow B$ is interpreted as a function $[\![A]\!]_{w'} \to [\![B]\!]_{w'}$ for all worlds w' that represent an increase in knowledge. Note that the generalization to the worlds w' is required to give a sound interpretation of the function type in the model. As for the interpretation of the type $\Box A$, if we think of R_m as specifying progression in time, and thus a world v with $w\ R_m\ v$ to represent a possible future of w, the interpretation of $\Box A$ at a world w can then be understood as a statement about the future: for all possible future worlds v, we have $[\![A]\!]_v$.

To extend the possible-world semantics of intuitionistic modal logic to Fitch-style calculi, we must also provide an interpretation for contexts—in particular for contexts extended with the 🔒 operator, which is unique to Fitch-style calculi.

$$\begin{aligned}
[\![\cdot]\!]_w &= \top \\
[\![\Gamma, x : A]\!]_w &= [\![\Gamma]\!]_w \times [\![A]\!]_w \\
[\![\Gamma, \text{🔒}]\!]_w &= \textstyle\sum_u [\![\Gamma]\!]_u \times u \; R_m \; w
\end{aligned}$$

The empty context $\cdot$ is interpreted as usual by the unit type $\top$, and the extension of a context with a variable $\Gamma, x : A$ is interpreted by the product of the interpretations $[\![\Gamma]\!]_w$ and $[\![A]\!]_w$. While the interpretation of the type $\Box A$ can be understood as a statement about the future, the interpretation of the context $\Gamma, \text{🔒}$ can be understood as a statement about a past: we have $[\![\Gamma]\!]_u$ in some past world u—i.e. a world u such that $u \; R_m \; w$.

The interpretation of terms, also known as *evaluation*, in a possible-world model must be given by a function $[\![-]\!] : \Gamma \vdash A \to (\forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w)$. Clouston [9] shows that Fitch-style calculi can be soundly interpreted in a Cartesian closed category (CCC) equipped with an adjunction of endofunctors by interpreting $\Box$ by the right adjoint and 🔒 by the left adjoint. The possible-world interpretation of types in the simply-typed fragment of Fitch-calculi is an instance of the CCC interpretation, which means that evaluation for this fragment in a possible-world model is readily given by evaluation in a CCC. In specific, it can be shown that the possible-world interpretation has a presheaf structure, which is an instance of the CCC interpretation. The key idea here is that this observation also extends to the modal fragment: the interpretation of $\Box$ in a possible-world model has a left adjoint that is given by our interpretation of 🔒, and this gives us an adjunction suitable for evaluation in the possible-world interpretation. This means that we may reuse the *generic* interpreter given by Clouston [9] for evaluating Fitch-style calculi in a possible-world model.

Constructing NbE Models as Instances To construct an NbE model for Fitch-calculi, we must construct a possible-world model with a function quote : $(\forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w) \to \Gamma \vdash_{\mathrm{NF}} A$ that inverts the denotation of a term $(\forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w)$ to a derivation $\Gamma \vdash_{\mathrm{NF}} A$ in normal form. The normal forms for the modal fragment of λ_{IK} are defined below, where $\Gamma \vdash_{\mathrm{NE}} A$ denotes a special case of normal forms known as *neutral elements*.

$$\frac{\Gamma, \text{🔒} \vdash_{\mathrm{NF}} t : A}{\Gamma \vdash_{\mathrm{NF}} \mathsf{box}\, t : \Box A}\;\Box\text{-}\textsc{Intro-NF} \qquad\qquad \frac{\Gamma \vdash_{\mathrm{NE}} t : \Box A}{\Gamma, \text{🔒}, \Gamma' \vdash_{\mathrm{NE}} \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, t : A}\;\text{🔒} \notin \Gamma' \;\;\Box\text{-}\textsc{Elim-NE}/\lambda_{\mathrm{IK}}$$

The normal forms for λ_{IT}, λ_{IK4}, and λ_{IS4} are defined similarly by varying the elimination rule as in their term typing rules in Fig. 2.

Following the work on NbE for STLC with possible-world[3] models [10], we instantiate the parameters that define possible-world models for Fitch-style calculi as follows: we pick contexts for W, *order-preserving embeddings* (sometimes

[3] also called "Kripke" or "Kripke-style"

called "weakenings", defined in the next section) $\Gamma \leq \Gamma'$ for $\Gamma\ R_i\ \Gamma'$, and neutral derivations $\Gamma \vdash_{\mathrm{NE}} \iota$ as the valuation $V_\iota(\Gamma)$. It remains for us to instantiate the parameter R_m and show that this model supports the quote function.

The instantiation of the modal parameter R_m in the possible-world semantics varies for each calculus and captures the differences between them. Recollect that the syntax of the four calculi only differ in their elimination rule for $\Box$ types. When viewed through the lens of the possible-world semantics, this difference can be generalized as follows:

$$\textsc{$\Box$-Elim}\quad \frac{\Gamma \vdash t : \Box A}{\Delta \vdash \mathsf{unbox}\, t : A}\ (\Gamma \sqsubseteq \Delta)$$

We generalize the relationship between the context in the premise and the context in the conclusion using a generic modal accessibility relation $\sqsubseteq$ between contexts. When viewed as a candidate for instantiating the R_m relation, this rule states that if $\Box A$ is derivable in some past world Γ, then we may derive A in the current world Δ. The various $\Box$-elimination rules for Fitch-style calculi can be viewed as instances of this generalized rule, where we define $\sqsubseteq$ in accordance with $\Box$-elimination rule of the calculus under consideration. For example, for λ_{IK}, we observe that the context of the premise in Rule $\Box$-Elim/λ_{IK} is Γ and that of the conclusion is $\Gamma, \text{🔒}, \Gamma'$ such that $\text{🔒} \notin \Gamma'$, and thus define $\Gamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \Delta$ as $\exists \Gamma'.\, \text{🔒} \notin \Gamma' \wedge \Delta = \Gamma, \text{🔒}, \Gamma'$. Similarly, we define $\Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta$ as $\exists \Gamma'.\, \Delta = \Gamma, \Gamma'$ for λ_{IS4}, and follow this recipe for λ_{IT} and λ_{IK4}. Accordingly, we instantiate the R_m parameter in the NbE model with the corresponding definition of $\sqsubseteq$ in the calculus under consideration.

A key component of implementing the quote function in NbE models is *reification*, which is implemented by a family of functions $\mathrm{reify}_A : \forall \Gamma.\, [\![A]\!]_\Gamma \to \Gamma \vdash_{\mathrm{NF}} A$ indexed by a type A. While its implementation for the simply-typed fragment follows the standard, for the modal fragment we are required to give an implementation of $\mathrm{reify}_{\Box A} : \forall \Gamma.\, [\![\Box A]\!]_\Gamma \to \Gamma \vdash_{\mathrm{NF}} \Box A$. To reify a value of $[\![\Box A]\!]_\Gamma$, we first observe that $[\![\Box A]\!]_\Gamma = \forall \Delta.\, \Gamma \sqsubseteq \Delta \to [\![A]\!]_\Delta$ by definition of $[\![-]\!]$ and the instantiation of R_m with $\sqsubseteq$. By picking $\Gamma, \text{🔒}$ for Δ, we get $[\![A]\!]_{\Gamma, \text{🔒}}$ since it can be shown that the extension $\Gamma \sqsubseteq \Gamma, \text{🔒}$ is valid for the calculi under consideration. By reifying the value $[\![A]\!]_{\Gamma, \text{🔒}}$ recursively, we get a normal form $\Gamma, \text{🔒} \vdash_{\mathrm{NF}} n : A$, which can be used to construct the desired normal form $\Gamma \vdash_{\mathrm{NF}} \mathsf{box}\, n : \Box A$ using the Rule Nf/$\Box$-Intro.

3 Possible-World Semantics and NbE

In this section, we elaborate on the previous section by defining possible-world models and showing that Fitch-style calculi can be interpreted soundly in these models. Following this, we outline the details of constructing NbE models as instances. We begin with the calculus λ_{IK}, and then show how the same results can be achieved for the other calculi.

Before discussing a concrete calculus, we present some of their commonalities.

Types, Contexts and Order-Preserving Embeddings The grammar of types and typing contexts for Fitch-style is the following.

$$\text{Ty} \quad A ::= \iota \mid A \Rightarrow B \mid \Box A \qquad\qquad \text{Ctx} \quad \Gamma ::= \cdot \mid \Gamma, A \mid \Gamma, \text{🔒}$$

Types are generated by a base type ι, function types $A \Rightarrow B$, and modal types $\Box A$, and typing contexts are "snoc" lists of types and locks.

We define the relation of *order-preserving embeddings* (OPE) on typing contexts in Fig. 3. An OPE $\Gamma \leq \Gamma'$ embeds the context Γ into another context Γ' while preserving the order of types and the order and number of locks in Γ.

$$\mathsf{base} : \cdot \leq \cdot \qquad \frac{o : \Gamma \leq \Gamma'}{\mathsf{drop}\, o : \Gamma \leq \Gamma', A} \qquad \frac{o : \Gamma \leq \Gamma'}{\mathsf{keep}\, o : \Gamma, A \leq \Gamma', A} \qquad \frac{o : \Gamma \leq \Gamma'}{\mathsf{keep}_{\text{🔒}}\, o : \Gamma, \text{🔒} \leq \Gamma', \text{🔒}}$$

Fig. 3. Order-preserving embeddings

3.1 The Calculus λ_{IK}

Terms, Substitutions and Equational Theory To define the intrinsically-typed syntax and equational theory of λ_{IK}, we first define a modal accessibility relation on contexts $\Gamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \Delta$, which expresses that context Δ extends $\Gamma, \text{🔒}$ to the right without adding locks. Note that $\Gamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \Delta$ exactly when $\exists \Gamma'. \text{🔒} \notin \Gamma' \wedge \Delta = \Gamma, \text{🔒}, \Gamma'$.

$$\mathsf{nil} : \Gamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \Gamma, \text{🔒} \qquad\qquad \frac{e : \Gamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \Delta}{\mathsf{var}\, e : \Gamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \Delta, A}$$

Fig. 4. Modal accessibility relation on contexts (λ_{IK})

Fig. 5 presents the intrinsically-typed syntax of λ_{IK}. Instead of named variables as in Fig. 1, variables are defined using De Bruijn indices in a separate judgement $\Gamma \vdash_{\mathrm{VAR}} A$. The introduction and elimination rules for function types are like those in STLC, and the introduction rule for the type $\Box A$ is similar to that of Fig. 1. The elimination rule $\Box$-Elim/λ_{IK} is defined using the modal accessibility relation $\Delta \sqsubseteq_{\lambda_{\mathrm{IK}}} \Gamma$ which relates the contexts in the premise and the conclusion, respectively. This relation replaces the side condition ($\text{🔒} \notin \Gamma'$) in Fig. 1 and other $\Box$-elimination rules in Sections 1 and 2. Note that formulating the rule for the term $\mathsf{unbox}_{\lambda_{\mathrm{IK}}}$ with $e : \Delta \sqsubseteq_{\lambda_{\mathrm{IK}}} \Gamma$ as a second premise is in sharp contrast to Clouston [9, Fig. 1] where the relation is not mentioned in the term but formulated as the *side condition* $\Gamma = \Delta, \text{🔒}, \Gamma'$ for some lock-free Γ'.

$$\textsc{Var-Zero}\quad \varGamma, A \vdash_{\mathrm{VAR}} \mathsf{zero} : A \qquad \textsc{Var-Succ}\ \frac{\varGamma \vdash_{\mathrm{VAR}} x : A}{\varGamma, B \vdash_{\mathrm{VAR}} \mathsf{succ}\, x : A} \qquad \textsc{Var}\ \frac{\varGamma \vdash_{\mathrm{VAR}} x : A}{\varGamma \vdash \mathsf{var}\, x : A} \qquad \Rightarrow\text{-}\textsc{Intro}\ \frac{\varGamma, A \vdash t : B}{\varGamma \vdash \lambda t : A \to B}$$

$$\Rightarrow\text{-}\textsc{Elim}\ \frac{\varGamma \vdash t : A \to B \qquad \varGamma \vdash u : A}{\varGamma \vdash \mathsf{app}\, t\, u : B} \qquad \square\text{-}\textsc{Intro}\ \frac{\varGamma, \text{🔒} \vdash t : A}{\varGamma \vdash \mathsf{box}\, t : \square A} \qquad \square\text{-}\textsc{Elim}/\lambda_{\mathrm{IK}}\ \frac{\varDelta \vdash t : \square A \qquad e : \varDelta \sqsubseteq_{\lambda_{\mathrm{IK}}} \varGamma}{\varGamma \vdash \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, t\, e : A}$$

Fig. 5. Intrinsically-typed terms of λ_{IK}

A term $\varGamma \vdash t : A$ can be *weakened*, which is a special case of *renaming*, with an OPE (Fig. 3) using a function $\mathrm{wk} : \varGamma \leq \varGamma' \to \varGamma \vdash A \to \varGamma' \vdash A$. Given an OPE $o : \varGamma \leq \varGamma'$, renaming the term using wk yields a term $\varGamma' \vdash \mathrm{wk}\, o\, t : A$ in the weaker context $\varGamma'$. The unit element for wk is the identity OPE $\mathsf{id}_{\leq} : \varGamma \leq \varGamma$, i.e. $\mathrm{wk}\, \mathsf{id}_{\leq}\, t = t$. Renaming arises naturally when evaluating terms and in specifying the equational theory (e.g. in the η rule of function type).

$$\cdot \vdash_{\mathrm{S}} \mathsf{empty} : \cdot \qquad \frac{\varGamma \vdash_{\mathrm{S}} s : \varDelta \qquad \varGamma \vdash A}{\varGamma \vdash_{\mathrm{S}} \mathsf{ext}\, s\, t : \varDelta, A} \qquad \frac{\varGamma' \vdash_{\mathrm{S}} s : \varDelta \qquad e : \varGamma' \sqsubseteq_{\lambda_{\mathrm{IK}}} \varGamma}{\varGamma \vdash_{\mathrm{S}} \mathsf{ext}_{\text{🔒}}\, s\, e : \varDelta, \text{🔒}}$$

Fig. 6. Substitutions for λ_{IK}

Substitutions for λ_{IK} are inductively defined in Fig. 6. A judgment $\varGamma \vdash_{\mathrm{S}} s : \varDelta$ denotes a substitution for a context $\varDelta$ in the context $\varGamma$. Applying a substitution to a term $\varDelta \vdash t : A$, i.e. $\mathrm{subst}\, s\, t : \varGamma \vdash A$, yields a term in the context $\varGamma$. The substitution $\mathsf{id}_{\mathrm{S}} : \varGamma \vdash_{\mathrm{S}} \varGamma$ denotes the identity substitution, which exists for all $\varGamma$. As usual, it can be shown that terms are closed under the application of a substitution, and that it preserves the identity, i.e. $\mathrm{subst}\, \mathsf{id}_{\mathrm{S}}\, t = t$. Substitutions are also closed under renaming and this operation preserves the identity as well.

The equational theory for λ_{IK}, omitting congruence rules, is specified in Fig. 7. As discussed earlier, λ_{IK} extends the usual rules in STLC (Rules $\Rightarrow$-β and $\Rightarrow$-η) with rules for the $\square$ type (Rules $\square$-β and $\square$-η). The function factor : $\varGamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \varDelta \to \varGamma, \text{🔒} \leq \varDelta$, in Rule $\square$-β, maps an element of the modal accessibility relation $e : \varGamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \varDelta$ to an OPE $\varGamma, \text{🔒} \leq \varDelta$. This is possible because the context $\varDelta$ does not have any lock to the right of $\varGamma, \text{🔒}$.

Possible-World Semantics A possible-world model is defined using the notion of a possible-world frame as below. We work in a constructive type-theoretic metalanguage, and denote the universe of types in this language by Type.

$$\Rightarrow\text{-}\beta\ \frac{\Gamma, A \vdash t : B \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{app}\,(\lambda\, t)\, u \sim \mathrm{subst}\,(\mathsf{ext}\,\mathsf{id_s}\, t)\, t} \qquad \Rightarrow\text{-}\eta\ \frac{\Gamma \vdash t : A \Rightarrow B}{\Gamma \vdash t \sim \lambda\,(\mathsf{app}\,(\mathrm{wk}\,(\mathsf{drop}\,\mathsf{id}_{\leq})\, t)\, \mathsf{zero})}$$

$$\Box\text{-}\beta\ \frac{\Gamma', 🔒 \vdash t : A \qquad e : \Gamma' \sqsubseteq_{\lambda_{\mathrm{IK}}} \Gamma}{\Gamma \vdash \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\,(\mathsf{box}\, t)\, e \sim \mathrm{wk}\,(\mathrm{factor}\, e)\, t} \qquad \Box\text{-}\eta\ \frac{\Gamma \vdash t : \Box A}{\Gamma \vdash t \sim \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\,(\mathsf{box}\, t)\, \mathsf{nil}}$$

Fig. 7. Equational theory for λ_{IK}

Definition 1 (Possible-world frame). *A frame F is given by a triple (W, R_i, R_m) consisting of a type W : Set and two relations R_i and R_m on W such that the following conditions are satisfied:*

- *R_i is reflexive and transitive*
- *$R_i \,;\, R_m = R_m \,;\, R_i$*

where $R_1 \,;\, R_2$ denotes composition of relations, i.e. $x\ (R_1 \,;\, R_2)\ y = \sum_z x\ R_1\ z \times z\ R_2\ y$.

Definition 2 (Possible-world model). *A possible-world model $\mathcal{M}$ is given by a tuple (F, V) consisting of a frame F (see Definition 1) and a* valuation *$V_\iota : W \to \mathrm{Type}$ of the base type such that $\forall w, w'.\, w\ R_i\ w' \to V_\iota(w) \to V_\iota(w')$.*

The types in λ_{IK} are interpreted in a possible-world model as in Section 2. To evaluate terms, we must first prove following *monotonicity* lemma. This lemma is well-known as a requirement to give a sound interpretation of the function type in an arbitrary possible-world model, and can be thought of as the semantic generalization of renaming in terms.

Lemma 1 (Monotonicity). *In every possible-world model $\mathcal{M}$, for every type A and worlds w and w', we have a function $\mathrm{wk}_A : w\ R_i\ w' \to [\![A]\!]_w \to [\![A]\!]_{w'}$. And similarly, for every context Γ, a function $\mathrm{wk}_\Gamma : w\ R_i\ w' \to [\![\Gamma]\!]_w \to [\![\Gamma]\!]_{w'}$.*

We evaluate terms in λ_{IK} in a possible-world model as follows.

$$\begin{aligned}
[\![-]\!] : \Gamma \vdash A &\to (\forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w)\\
[\![\mathsf{var}\, x]\!]\ \gamma &= \mathrm{lookup}\, x\, \gamma\\
[\![\lambda\, t]\!]\ \gamma &= \lambda i.\, \lambda y.\, [\![t]\!]\ (\mathrm{wk}\, i\, \gamma, y)\\
[\![\mathsf{app}\, t\, u]\!]\ \gamma &= ([\![t]\!]\ \gamma)\ \mathsf{id}_{\leq}\ ([\![u]\!]\ \gamma)\\
[\![\mathsf{box}\, t]\!]\ \gamma &= \lambda m.\, [\![t]\!]\ (\gamma, m)\\
[\![\mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, t\, e]\!]\ \gamma &= [\![t]\!]\ \gamma'\ m\\
\text{where } (\gamma', m) &= \mathrm{trim}_{\lambda_{\mathrm{IK}}}\, \gamma\, e
\end{aligned}$$

The evaluation of terms in the simply-typed fragment is standard, and resembles the evaluator of STLC. Variables are interpreted by a lookup function that projects values from an environment, and λ-abstraction and application are evaluated using their semantic counterparts. To evaluate λ-abstraction, we must construct a semantic function $\forall w'.\, w\ R_i\ w' \to [\![A]\!]_{w'} \to [\![B]\!]_{w'}$ using the given term $\Gamma, A \vdash t : B$ and environment $\gamma : [\![\Gamma]\!]_w$. We achieve this by recursively evaluating t in an environment that extends γ appropriately using the semantic arguments $i : w\ R_i\ w'$ and $y : [\![A]\!]_{w'}$. We use the monotonicity lemma to "transport" $[\![\Gamma]\!]_w$ to $[\![\Gamma]\!]_{w'}$, and construct an environment of type $[\![\Gamma]\!]_{w'} \times [\![A]\!]_{w'}$ for recursively evaluating t, which produces the desired result of type $[\![B]\!]_{w'}$. Application is evaluated by simply recursively evaluating the applied terms and applying them in the semantics with a value $\mathsf{id}_{\leq} : w\ R_i\ w$, which is available since R_i is reflexive.

In the modal fragment, to evaluate the term $\Gamma \vdash \mathsf{box}\, t : \Box A$ with $\gamma : [\![\Gamma]\!]_w$, we must construct a function of type $\forall v.\, w\ R_m\ v \to [\![A]\!]_v$. Using the semantic argument $m : w\ R_m\ v$, we recursively evaluate the term $\Gamma, 🔒 \vdash t : A$ in the extended environment $(\gamma, m) : [\![\Gamma, 🔒]\!]_v$, since $[\![\Gamma, 🔒]\!]_v = \sum_w [\![\Gamma]\!]_w \times w\ R_m\ v$. On the other hand, the term $\Gamma \vdash \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, t\, e : A$ with $e : \Delta \sqsubseteq_{\lambda_{\mathrm{IK}}} \Gamma$ and $\Delta \vdash t : \Box A$, for some Δ, must be evaluated with an environment $\gamma : [\![\Gamma]\!]_w$. To recursively evaluate the term $\Delta \vdash t : \Box A$, we must first discard the part of the environment γ that substitutes the types in the extension of $\Delta, 🔒$. This is achieved using the function $\mathrm{trim}_{\lambda_{\mathrm{IK}}} : [\![\Gamma]\!]_w \to \Delta \sqsubseteq_{\lambda_{\mathrm{IK}}} \Gamma \to [\![\Delta, 🔒]\!]_w$ that projects γ to produce an environment $\gamma' : [\![\Gamma]\!]_v$ and a value $m : v\ R_m\ w$. We evaluate t with γ' and the apply the resulting function of type $\forall w.\, v\ R_m\ w \to [\![A]\!]_w$ with m to return the desired result.

We state the soundness of λ_{IK} with respect to the possible-world semantics before we instantiate it with the NbE model that we will construct in the next subsection.

Theorem 1. *Let $\mathcal{M}$ be any possible-world model (see Definition 2). If two terms t and $u : \Gamma \vdash A$ of λ_{IK} are equivalent (see Fig. 7) then $[\![t]\!]$ and $[\![u]\!] : \forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w$ are equal in $\mathcal{M}$.*

Proof. *Show that possible-world models, which are particular presheaf categories, are Cartesian closed and come equipped with an adjunction, then apply Clouston [9, Theorem 2.8 (with remark below)].*

NbE Model The normal forms of terms in λ_{IK} are defined along with neutral elements in a mutually recursive fashion by the judgements $\Gamma \vdash_{\mathrm{NF}} A$ and $\Gamma \vdash_{\mathrm{NE}} A$, respectively, in Fig. 8. Intuitively, a normal form may be thought of as a value, and a neutral element may be thought of as a "stuck" computation. We extend the standard definition of normal forms and neutral elements in STLC with Rules NF/$\Box$-INTRO and NE/$\Box$-ELIM/λ_{IK}.

Recall that an NbE model for a given calculus C is a particular kind of model $\mathcal{M}$ that comes equipped with a function $\mathrm{quote} : \mathcal{M}([\![\Gamma]\!], [\![A]\!]) \to \Gamma \vdash_{\mathrm{NF}} A$ satisfying $t \sim \mathrm{quote}\, [\![t]\!]$ for all terms $t : \Gamma \vdash A$ where $[\![-]\!]$ denotes the *generic* evaluation function for C.

Using the relations defined in Figs. 3 and 4, we construct an NbE model for λ_{IK} by instantiating the parameters that define a *possible-world* model as follows.

$$\textsc{Ne/Var}\ \frac{\Gamma \vdash_{\mathrm{VAR}} x : A}{\Gamma \vdash_{\mathrm{NE}} \mathsf{var}\, x : A} \qquad \textsc{Nf/Up}\ \frac{\Gamma \vdash_{\mathrm{NE}} n : \iota}{\Gamma \vdash_{\mathrm{NF}} \mathsf{up}\, n : \iota} \qquad \textsc{Nf/}{\Rightarrow}\textsc{-Intro}\ \frac{\Gamma, A \vdash_{\mathrm{NF}} n : B}{\Gamma \vdash_{\mathrm{NF}} \lambda\, n : A \Rightarrow B}$$

$$\textsc{Ne/}{\Rightarrow}\textsc{-Elim}\ \frac{\Gamma \vdash_{\mathrm{NE}} n : A \to B \qquad \Gamma \vdash_{\mathrm{NF}} m : A}{\Gamma \vdash_{\mathrm{NE}} \mathsf{app}\, n\, m : B} \qquad \textsc{Nf/}\Box\textsc{-Intro}\ \frac{\Gamma, \text{🔒} \vdash_{\mathrm{NF}} n : A}{\Gamma \vdash_{\mathrm{NF}} \mathsf{box}\, n : \Box A}$$

$$\textsc{Ne/}\Box\textsc{-Elim/}\lambda_{\mathrm{IK}}\ \frac{\Delta \vdash_{\mathrm{NE}} n : \Box A \qquad e : \Delta \sqsubseteq_{\lambda_{\mathrm{IK}}} \Gamma}{\Gamma \vdash_{\mathrm{NE}} \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, n\, e : A}$$

Fig. 8. Normal forms and neutral elements in λ_{IK}

- Worlds as contexts: $W = \mathrm{Ctx}$
- Relation R_i as order-preserving embeddings: $\Gamma\ R_i\ \Gamma' = \Gamma \leq \Gamma'$
- Relation R_m as extensions of a "locked" context: $\Gamma\ R_m\ \Delta = \Gamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \Delta$
- Valuation V_ι as neutral elements: $V_\iota(\Gamma) = \Gamma \vdash_{\mathrm{NE}} \iota$

The condition that the valuation must satisfy $\mathrm{wk}_A : \Gamma \leq \Gamma' \to \Gamma \vdash_{\mathrm{NE}} A \to \Gamma' \vdash_{\mathrm{NE}} A$, for all types A, can be shown by induction on the OPE $\Gamma \leq \Gamma'$. To show that this model is indeed a possible-world model, it remains for us to show that the frame conditions are satisfied.

The first frame condition states that OPEs must be reflexive and transitive, which can be shown by structural induction on the context and definition of OPEs, respectively. The second frame condition states that the types $\sum_{\Gamma'} \Gamma \leq \Gamma' \times \Gamma' \sqsubseteq_{\lambda_{\mathrm{IK}}} \Delta$ and $\sum_{\Delta'} \Gamma \sqsubseteq_{\lambda_{\mathrm{IK}}} \Delta' \times \Delta' \leq \Delta$ are isomorphic for all Γ, Δ : Ctx, which can be shown by constructing mutually inverse functions by simultaneous recursion on OPEs and the modal accessibility relation.

Observe that the instantiation of the monotonicity lemma in the NbE model states that we have the functions $\mathrm{wk}_A : \Gamma \leq \Gamma' \to [\![A]\!]_\Gamma \to [\![A]\!]_{\Gamma'}$ and $\mathrm{wk}_\Gamma : \Gamma \leq \Gamma' \to [\![\Gamma]\!]_\Gamma \to [\![\Gamma]\!]_{\Gamma'}$, which allow denotations of types and contexts to be renamed with respect to an OPE.

To implement the function quote, we first implement *reification* and *reflection*, using two functions $\mathrm{reify}_A : [\![A]\!]_\Gamma \to \Gamma \vdash_{\mathrm{NF}} A$ and $\mathrm{reflect}_A : \Gamma \vdash_{\mathrm{NE}} A \to [\![A]\!]_\Gamma$, respectively. Reification converts a semantic value to a normal form, while reflection converts a neutral element to a semantic value. They are implemented as follows by induction on the index type A.

$$
\begin{array}{ll}
\text{reify}_{A,\Gamma} : [\![A]\!]_\Gamma \to \Gamma \vdash_{\text{NF}} A & \\
\text{reify}_{\iota,\Gamma} \quad\;\; n & = \mathsf{up}\, n \\
\text{reify}_{A \Rightarrow B,\Gamma}\; f & = \lambda\, (\text{reify}_{B,(\Gamma,A)} (f\, (\mathsf{drop}\, \mathsf{id}_{\leq})\, \text{fresh}_{A,(\Gamma,A)})) \\
\text{reify}_{\Box A,\Gamma} \quad g & = \mathsf{box}\, (\text{reify}_{A,(\Gamma,\text{🔒})} (g\, \mathsf{nil}))
\end{array}
$$

$$
\begin{array}{ll}
\text{reflect}_{A,\Gamma} : \Gamma \vdash_{\text{NE}} A \to [\![A]\!]_\Gamma & \\
\text{reflect}_{\iota,\Gamma} \quad\;\; n & = n \\
\text{reflect}_{A \Rightarrow B,\Gamma}\; n & = \lambda o.\, \lambda x.\, \text{reflect}_{B,\Gamma}(\mathsf{app}\, (\text{wk}_{A \Rightarrow B}\, o\, n)\, (\text{reify}_{B,\Gamma'}\, x)) \\
\text{reflect}_{\Box A,\Gamma} \quad n & = \lambda (e : \Gamma \sqsubseteq_{\lambda_{\text{IK}}} \Delta).\, \text{reflect}_{A,\Delta}(\mathsf{unbox}_{\lambda_{\text{IK}}}\, n\, e)
\end{array}
$$

For the function type, we recursively reify the body of the λ-abstraction by applying the given semantic function f with suitable arguments, which are an OPE $\mathsf{drop}\,\mathsf{id}_{\leq} : \Gamma \leq \Gamma, A$ and a value $\text{fresh}_{A,(\Gamma,A)} = \text{reflect}_{A,(\Gamma,A)}\, (\mathsf{var}\,\mathsf{zero}) : [\![A]\!]_{\Gamma,A}$—which is the De Bruijn index equivalent of a fresh variable. Reflection, on the other hand, recursively reflects the application of a neutral $\Gamma \vdash_{\text{NE}} n : A \Rightarrow B$ to the reification of the semantic argument $x : [\![A]\!]_{\Gamma'}$ for an OPE $o : \Gamma \leq \Gamma'$. Similarly, for the $\Box$ type, we recursively reflect the body of box by applying the given semantic function $g : \forall \Delta.\, \Gamma \sqsubseteq_{\lambda_{\text{IK}}} \Delta \to [\![A]\!]_\Delta$ with a suitable argument, which is the empty context extension $\mathsf{nil} : \Gamma \sqsubseteq_{\lambda_{\text{IK}}} \Gamma, \text{🔒}$. Reflection also follows a similar pursuit by reflecting the application of the neutral $\Gamma \vdash_{\text{NE}} n : \Box A$ to the eliminator unbox.

Equipped with reification, we implement quote (as seen below), by applying the given denotation of a term, a function $f : \forall \Delta.\, [\![\Gamma]\!]_\Delta \to [\![A]\!]_\Delta$, to the identity environment $\text{freshEnv}_\Gamma : [\![\Gamma]\!]_\Gamma$, and then reifying the resulting value. The construction of the value freshEnv_Γ is the De Bruijn index equivalent of generating an environment with fresh variables.

$$
\begin{array}{l}
\text{quote} : (\forall \Delta.\, [\![\Gamma]\!]_\Delta \to [\![A]\!]_\Delta) \to \Gamma \vdash_{\text{NF}} A \\
\text{quote}\, f = \text{reify}_{A,\Gamma}\, (f\, \text{freshEnv}_\Gamma)
\end{array}
$$

$$
\begin{array}{ll}
\text{freshEnv}_\Gamma : [\![\Gamma]\!]_\Gamma & \\
\text{freshEnv}_\cdot & = () \\
\text{freshEnv}_{\Gamma,A} & = (\text{wk}\, (\mathsf{drop}\, \mathsf{id}_{\leq})\, \text{freshEnv}_\Gamma, \text{fresh}_{A,(\Gamma,A)}) \\
\text{freshEnv}_{\Gamma,\text{🔒}} & = (\text{freshEnv}_\Gamma, \mathsf{nil}\,)
\end{array}
$$

To prove that the function quote is indeed a retraction of evaluation, we follow the usual logical relations approach. As seen in Fig. 9, we define a relation L_A indexed by a type A that relates a term $\Gamma \vdash t : A$ to its denotation $x : [\![A]\!]_\Gamma$ as $\text{L}_A\, t\, x$. From a proof of $\text{L}_A\, t\, x$, it can be shown that $t \sim \text{reify}_A\, x$. This relation is extended to contexts as L_Δ, for some context Δ, which relates a substitution $\Gamma \vdash s : \Delta$ to its denotation $x : [\![\Delta]\!]_\Gamma$ as $\text{L}_A\, s\, x$.

For the logical relations, we then prove the so-called fundamental theorem.

$$
\begin{array}{ll}
\mathrm{L}_{A,\Gamma} : \Gamma \vdash A \to [\![A]\!]_{\Gamma} \to \mathrm{Type} & \\
\mathrm{L}_{\iota,\Gamma} & t\ x = t \sim \mathrm{quote}\, x \\
\mathrm{L}_{A \Rightarrow B,\Gamma} & t\ f = \forall \Gamma', e : \Gamma \leq \Gamma', u, x.\, \mathrm{L}_{A,\Gamma'}\, u\, x \to \mathrm{L}_{B,\Gamma'}\, (\mathsf{app}\, (\mathrm{wk}\, e\, t)\, u)\, (f\, e\, x) \\
\mathrm{L}_{\Box A,\Gamma} & t\ g = \sum_{u} \mathrm{L}_{A,(\Gamma,\text{🔒})}\, u\, (g\ \mathsf{nil}) \times t \sim \mathsf{box}\, u
\end{array}
$$

$$
\begin{array}{llll}
\mathrm{L}_{\Delta,\Gamma} : \Gamma \vdash_{\mathrm{S}} \Delta \to [\![\Delta]\!]_{\Gamma} \to \mathrm{Type} & & & \\
\mathrm{L}_{\cdot,\Gamma} & \mathsf{empty} & () & = \top \\
\mathrm{L}_{(\Delta,A),\Gamma} & (\mathsf{ext}\, s\, t) & (x, y) & = \mathrm{L}_{\Delta,\Gamma}\, s\, x \times \mathrm{L}_{A,\Gamma}\, t\, y \\
\mathrm{L}_{(\Delta,\text{🔒}),\Gamma} & (\mathsf{ext}_{\text{🔒}}\, s\, (e : \Gamma' \sqsubseteq_{\lambda_{\mathrm{IK}}} \Gamma)) & (x, e) & = \mathrm{L}_{\Delta,\Gamma'}\, s\, x
\end{array}
$$

Fig. 9. Logical relations for λ_{IK}

Proposition 1 (Fundamental theorem). *Given a term $\Delta \vdash t : A$, a substitution $\Gamma \vdash_{\mathrm{S}} s : \Delta$ and a value $x : [\![\Delta]\!]_{\Gamma}$, if $\mathrm{L}\, s\, x$ then $\mathrm{L}\, (\mathrm{subst}\, s\, t)\, ([\![t]\!]\, x)$.*

We conclude this subsection by stating the normalization theorem for λ_{IK}.

Proposition 1 entails that $\mathrm{L}\, (\mathrm{subst}\, \mathsf{id}_{\mathrm{S}}\, t)\, ([\![t]\!]\, \mathrm{freshEnv}_{\Delta})$ for any term t, if we pick s as the identity substitution $\mathsf{id}_{\mathrm{S}} : \Delta \vdash_{\mathrm{S}} \Delta$, and x as $\mathrm{freshEnv}_{\Delta} : [\![\Delta]\!]_{\Delta}$, since they can be shown to be related as $\mathrm{L}\, \mathsf{id}_{\mathrm{S}}\, \mathrm{freshEnv}$. From this it follows that $\mathrm{subst}\, \mathsf{id}_{\mathrm{S}}\, t \sim \mathrm{reify}_{A}\, ([\![t]\!]\, \mathrm{freshEnv})$, and further that $t \sim \mathrm{quote}\, [\![t]\!]$ from the definition of quote and the fact that $\mathrm{subst}\, \mathsf{id}_{\mathrm{S}}\, t = t$. As a result, the composite $\mathrm{norm} = \mathrm{quote} \circ [\![-]\!]$ is *adequate*, i.e. $\mathrm{norm}\, t = \mathrm{norm}\, t'$ implies $t \sim t'$.

The soundness of λ_{IK} with respect to possible-world models (see Theorem 1) directly entails $\mathrm{quote}\, [\![t]\!] = \mathrm{quote}\, [\![u]\!] : \Gamma \vdash_{\mathrm{NF}} A$ for all terms t, $u : \Gamma \vdash A$ such that $\Gamma \vdash t \sim u : A$, which means that $\mathrm{norm} = \mathrm{quote} \circ [\![-]\!]$ is *complete*. Note that this terminology might be slightly confusing because it is the *soundness* of $[\![-]\!]$ that implies the *completeness* of norm.

Theorem 2. *Let $\mathcal{M}$ denote the possible-world model over the frame given by the relations $\Gamma \leq \Gamma'$ and $\Gamma \sqsubseteq_{\lambda_{IK}} \Delta$ and the valuation $V_{\iota,\Gamma} = \Gamma \vdash_{\mathrm{NE}} \iota$.*

There is a function $\mathrm{quote} : \mathcal{M}([\![\Gamma]\!], [\![A]\!]) \to \Gamma \vdash_{\mathrm{NF}} A$ such that the composite $\mathrm{norm} = \mathrm{quote} \circ [\![-]\!] : \Gamma \vdash A \to \Gamma \vdash_{\mathrm{NF}} A$ from terms to normal forms of λ_{IK} is complete and adequate.

3.2 Extending to the Calculus λ_{IS4}

Terms, Substitutions and Equational Theory To define the intrinsically-typed syntax of λ_{IS4}, we first define the modal accessibility relation on contexts in Fig. 10.

If $\Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta$ then Δ is an extension of Γ with as many locks as needed. Note that, in contrast to λ_{IK}, the modal accessibility relation is both reflexive and transitive. This corresponds to the conditions on the accessibility relation for the logic IS4.

Fig. 11 presents the changes of λ_{IK} that yield λ_{IS4}. The terms are the same as λ_{IK} with the exception of Rule $\Box$-ELIM/λ_{IK} which now includes the modal

$$\mathsf{nil} : \Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Gamma \qquad \frac{e : \Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta}{\mathsf{var}\, e : \Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta, A} \qquad \frac{e : \Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta}{\mathsf{lock}\, e : \Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta, \text{🔒}}$$

Fig. 10. Modal accessibility relation on contexts (λ_{IS4})

accessibility relation for λ_{IS4}. Similarly, the substitution rule for contexts with locks now refers to $\sqsubseteq_{\lambda_{\mathrm{IS4}}}$.

$$\square\text{-}\textsc{Elim}/\lambda_{\mathrm{IS4}}\; \frac{\Delta \vdash t : \square A \qquad e : \Delta \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Gamma}{\Gamma \vdash \mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t\, e : A} \qquad \frac{\Gamma' \vdash s : \Delta \qquad e : \Gamma' \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Gamma}{\Gamma \vdash_{\mathrm{s}} \mathsf{ext}_{\text{🔒}}\, s\, e : \Delta, \text{🔒}}$$

Fig. 11. Intrinsically-typed terms and substitutions of λ_{IS4}

Fig. 12 presents the equational theory of the modal fragment of λ_{IS4}. This is a slightly modified version of λ_{IK} (cf. Fig. 7) that accommodates the changes to the Rule $\square$-Elim/λ_{IS4}. Unlike before, Rule $\square$-β now performs a substitution to modify the term $\Gamma', \text{🔒} \vdash t : A$ to a term of type $\Gamma \vdash A$. Note that the result of such a substitution need not yield the same term since substitution may change the context extension of some subterm.

$$\square\text{-}\beta\; \frac{\Gamma', \text{🔒} \vdash t : A \qquad e : \Gamma' \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Gamma}{\Gamma \vdash \mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, (\mathsf{box}\, t)\, e \sim \mathrm{subst}\, (\mathsf{ext}_{\text{🔒}}\, \mathsf{id}_{\mathrm{s}}\, e)\, t} \qquad \square\text{-}\eta\; \frac{\Gamma \vdash t : \square A}{\Gamma \vdash t \sim \mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, (\mathsf{box}\, t)\, (\mathsf{lock}\, \mathsf{nil})}$$

Fig. 12. Equational theory for λ_{IS4}

Possible-World Semantics Giving possible-world semantics for λ_{IS4} requires an additional frame condition on the relation R_m: it must be reflexive and transitive. Evaluation proceeds as before, where we use a function $\mathrm{trim}_{\lambda_{\mathrm{IS4}}} : \forall w.\, [\![\Gamma]\!]_w \to \Gamma' \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Gamma \to [\![\Gamma', \text{🔒}]\!]_w$ to manipulate the environment for evaluating $\mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t\, e$, as seen below.

$$\begin{aligned} [\![\mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t\, e]\!]\; \gamma &= [\![t]\!]\; \gamma'\; m \\ \text{where } (\gamma', m) &= \mathrm{trim}_{\lambda_{\mathrm{IS4}}}\; \gamma\; e \end{aligned}$$

The additional frame requirements ensures that the function $\mathrm{trim}_{\lambda_{\mathrm{IS4}}}$ can be implemented. For example, consider implementing the case of $\mathrm{trim}_{\lambda_{\mathrm{IS4}}}$ for some argument of type $[\![\Gamma]\!]_w$ and the extension $\mathsf{nil} : \Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Gamma$ adds zero locks. The desired result is of type $[\![\Gamma, 🔒]\!]_w$, which is defined as $\sum_v [\![\Gamma]\!]_v \times v\ R_m\ w$. We construct such a result using the argument of $[\![\Gamma]\!]_w$ by picking v as w itself, and using the reflexivity of R_m to construct a value of type $w\ R_m\ w$. Similarly, the transitivity of R_m is required when the context extension adds more than one lock.

Analogously to Theorem 1, we state the soundness of λ_{IS4} with respect to *reflexive and transitive* possible-world models before we instantiate it with the NbE model that we will construct in the next subsection.

Proposition 2. *Let $\mathcal{C}$ be a Cartesian closed category equipped with a comonad $\Box$ that has a left adjoint $🔒 \dashv \Box$, then equivalent terms t and $u : \Gamma \vdash A$ denote equal morphisms in $\mathcal{C}$.*

*Proof. A version of Clouston [9, Theorem 4.8] for λ_{IS4} where the side condition of Rule $\Box$-*Elim*/λ_{IS4} appears as an argument to the term former* unbox *and hence idempotency is not imposed on the comonad $\Box$.*

Theorem 3. *Let $\mathcal{M}$ be a possible-world model (see Definition 2) such that the modal accessibility relation R_m is reflexive and transitive. If two terms t and $u : \Gamma \vdash A$ of λ_{IS4} are equivalent (see Fig. 12) then $[\![t]\!]$ and $[\![u]\!] : \forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w$ are equal in $\mathcal{M}$.*

Proof. Show that reflexive and transitive possible-world models, which are particular presheaf categories, are Cartesian closed and come equipped with a comonad that has a left adjoint, then apply Proposition 2.

NbE Model The normal forms of λ_{IS4} are defined as before, with the following rule that replaces the neutral Rule Ne/$\Box$-Elim/λ_{IK}.

$$\textsc{Ne}/\Box\text{-}\textsc{Elim}/\lambda_{\mathrm{IS4}}\quad \frac{\Delta \vdash_{\mathrm{NE}} n : \Box A \qquad e : \Delta \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Gamma}{\Gamma \vdash_{\mathrm{NE}} \mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t\, e : A}$$

The NbE model construction also proceeds in the same way, where we now pick the relation R_m as arbitrary extensions of a context: $\Gamma\ R_m\ \Delta = \Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta$. The modal fragment for reify and reflect are now implemented as follows:

$$\begin{aligned}
\mathrm{reify}_{\Box A,\Gamma}\ \ g &= \mathsf{box}\,(\mathrm{reify}_{A,(\Gamma,🔒)}\,(g\,(\mathsf{lock}\,\mathsf{nil})))\\
\mathrm{reflect}_{\Box A,\Gamma}\ n &= \lambda(e : \Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta).\, \mathrm{reflect}_{A,\Delta}\,(\mathsf{unbox}\, n\, e)
\end{aligned}$$

Conjecture 1. Let $\mathcal{M}$ denote the possible-world model over the reflexive and transitive frame given by the relations $\Gamma \leq \Gamma'$ and $\Gamma \sqsubseteq_{\lambda_{\mathrm{IS4}}} \Delta$ and the valuation $V_{\iota,\Gamma} = \Gamma \vdash_{\mathrm{NE}} \iota$.

There is a function quote : $\mathcal{M}([\![\Gamma]\!], [\![A]\!]) \to \Gamma \vdash_{\mathrm{NF}} A$ such that the composite norm $=$ quote $\circ$ $[\![-]\!] : \Gamma \vdash A \to \Gamma \vdash_{\mathrm{NF}} A$ from terms to normal forms of λ_{IS4} is complete and adequate.

3.3 Extending to the Calculi λ_{IT} and λ_{IK4}

The NbE model construction for λ_{IT} and λ_{IK4} follows a similar pursuit as λ_{IS4}. We define suitable modal accessibility relations $\sqsubseteq_{\lambda_{\mathrm{IT}}}$ and $\sqsubseteq_{\lambda_{\mathrm{IK4}}}$ as extensions that allow the addition of at most one 🔒, and at least one lock 🔒, respectively. To give possible-world semantics, we require an additional frame condition that the relation R_m be reflexive for λ_{IT} and transitive for λ_{IK4}. For evaluation, we use a function $\mathrm{trim}_{\lambda_{\mathrm{IT}}} : [\![\Gamma]\!]_w \to \Delta \sqsubseteq_{\lambda_{\mathrm{IT}}} \Gamma \to [\![\Delta, \text{🔒}]\!]_w$ for λ_{IT}, and similarly $\mathrm{trim}_{\lambda_{\mathrm{IK4}}}$ for λ_{IK4}. The modification to the neutral Rule NE/$\Box$-ELIM/λ_{IK} is achieved as before in λ_{IS4} using the corresponding modal accessibility relations. Unsurprisingly, reification and reflection can also be implemented, thus yielding normalization functions for both λ_{IT} and λ_{IK4}.

4 Completeness, Decidability and Logical Applications

In this section we record some immediate consequences of the model constructions we presented in the previous section.

Completeness of the Equational Theory As a corollary of the adequacy of an NbE model $\mathcal{N}$, i.e. $\Gamma \vdash t \sim u : A$ whenever $[\![t]\!] = [\![u]\!] : \mathcal{N}([\![\Gamma]\!], [\![A]\!])$, we obtain completeness of the equational theory with respect to the class of models that the respective NbE model belongs to. Given the NbE models constructed in subsections 3.1 and 3.2 this means that the equational theories of λ_{IK} and λ_{IS4} (cf. Fig. 7) are (sound and) complete with respect to the class of Cartesian closed categories equipped with an adjunction and a right-adjoint comonad, respectively.

Theorem 4. *Let t, $u : \Gamma \vdash A$ be two terms of λ_{IK}. If for all Cartesian closed categories $\mathcal{M}$ equipped with an adjunction it is the case that $[\![t]\!] = [\![u]\!] : \mathcal{M}([\![\Gamma]\!], [\![A]\!])$ then $\Gamma \vdash t \sim u : A$.*

Proof. Let $\mathcal{M}_0$ be the model we constructed in subsection 3.1. Since $\mathcal{M}_0$ is a Cartesian closed category equipped with an adjunction, by assumption we have $[\![t]\!]_{\mathcal{M}_0} = [\![u]\!]_{\mathcal{M}_0}$. And lastly, since $\mathcal{M}_0$ is an NbE model, we have $\Gamma \vdash t \sim \mathrm{quote}([\![t]\!]_{\mathcal{M}_0}) = \mathrm{quote}([\![u]\!]_{\mathcal{M}_0}) \sim u : A$.

Note that this statement corresponds to Clouston [9, Theorem 3.2] but there it is obtained via a term model construction and for the term model to be equipped with an adjunction the calculus needs to be first extended with an internalization of the operation 🔒 on contexts as an operation $\blacklozenge$ on types.

Theorem 5. *Let t, $u : \Gamma \vdash A$ be two terms of λ_{IS4}. If for all Cartesian closed categories $\mathcal{M}$ equipped with a right-adjoint comonad it is the case that $[\![t]\!] = [\![u]\!] : \mathcal{M}([\![\Gamma]\!], [\![A]\!])$ then $\Gamma \vdash t \sim u : A$.*

Proof. As for Theorem 4.

This statement corresponds to Clouston [9, Section 4.4] but there it is proved for an equational theory that identifies terms up to differences in the accessibility proofs and with respect to the class of models where the comonad is *idempotent.*

Completeness of the Deductive Theory Using the quotation function of an NbE model $\mathcal{N}$, i.e. quote : $\mathcal{N}([\![\Gamma]\!], [\![A]\!]) \to \Gamma \vdash A$, we obtain completeness of the deductive theory with respect to the class of models that the respective NbE model belongs to. Given the NbE models constructed in subsections 3.1 and 3.2 this means that the deductive theories of λ_{IK} and λ_{IS4} (cf. Figs. 2 and 5) are (sound and) complete with respect to the class of possible-world models with an arbitrary frame and a reflexive–transitive frame, respectively.

Theorem 6. *Let Γ :* Ctx *be a context and A :* Ty *a type. If for all possible-world models $\mathcal{M}$ it is the case that $\mathcal{M}([\![\Gamma]\!], [\![A]\!])$ is inhabited then there is a term $t : \Gamma \vdash A$ of λ_{IK}.*

Proof. Let $\mathcal{M}_0$ be the model we constructed in subsection 3.1. Since $\mathcal{M}_0$ is a possible-world model, by assumption we have a morphism $p : \mathcal{M}_0([\![\Gamma]\!], [\![A]\!])$. And lastly, since $\mathcal{M}_0$ is an NbE model, we have the term $\text{quote}(p) : \Gamma \vdash A$.

Theorem 7. *Let Γ :* Ctx *be a context and A :* Ty *a type. If for all possible-world models $\mathcal{M}$ with a reflexive–transitive frame it is the case that $\mathcal{M}([\![\Gamma]\!], [\![A]\!])$ is inhabited then there is a term $t : \Gamma \vdash A$ of λ_{IS4}.*

Proof. As for Theorem 6.

Note that the proofs of Theorems 6 and 7 are constructive.

Decidability As a corollary of the completeness and adequacy of an NbE model $\mathcal{N}$, i.e. $\Gamma \vdash t \sim u : A$ if and only if $[\![t]\!] = [\![u]\!] : \mathcal{N}([\![\Gamma]\!], [\![A]\!])$, we obtain decidability of the equational theory from decidability of the equality of normal forms n, m : $\Gamma \vdash_{\text{NF}} A$. Given the NbE models constructed in subsections 3.1 and 3.2 this means that the equational theories of λ_{IK} and λ_{IS4} (cf. Fig. 7) are decidable.

To show that any of the following decision problems $P(x)$ is decidable we give a *constructive* proof of the proposition $\forall x. P(x) \vee \neg P(x)$. Such a proof can be understood as the construction of an algorithm d that takes as input an x and produces as output a Boolean $d(x)$, alongside a correctness proof that $d(x)$ is `true` if and only if $P(x)$ holds.

Theorem 8. *For any two terms t, $u : \Gamma \vdash A$ of λ_{IK} the problem whether $t \sim u$ is decidable.*

Proof. We first observe that for any two normal forms n, $m : \Gamma \vdash_{\text{NF}} A$ of λ_{IK} the problem whether $n = m$ is decidable by proving $\forall n, m. n = m \vee n \neq m$ constructively. All the cases of an simultaneous induction on n, $m : \Gamma \vdash_{\text{NF}} A$ are immediate.

Let $\mathcal{N}$ be the NbE model we constructed in subsection 3.1. Completeness and adequacy of $\mathcal{N}$ imply that we have $t \sim u$ if and only if $\text{norm}\, t = \text{norm}\, u$ *for the function* $\text{norm} : \Gamma \vdash A \to \Gamma \vdash_{\text{NF}} A, t \mapsto \text{quote}\,[\![t]\!]$. *Now, $t \sim u$ is decidable because* $\text{norm}\, t = \text{norm}\, u$ *is decidable by the observation we started with.*

Theorem 9. *For any two terms t, $u : \Gamma \vdash A$ of λ_{IS4} the problem whether $t \sim u$ is decidable.*

Proof. As for Theorem 8.

Denecessitation The last of the consequences of the NbE model constructions we record is of a less generic flavour than the other three, namely it is an application of normal forms to a basic proof-theoretic result in modal logic.

Using invariance of truth in possible-world models under bisimulation[4] it can be shown that $\Box A$ is a valid formula of IK (or IS4) if and only if A is. A completeness theorem then implies the same for provability of $\Box A$ and A. The statement for proofs in λ_{IK} (and λ_{IS4}) can also be shown by inspection of normal forms as follows.

Firstly, we note that while deduction is not closed under arbitrary context extensions (including locks) it is closed under extensions (including locks) *on the left*:

Lemma 2 (cf. Clouston [9, Lemma A.1]). *Let* Δ, Γ : Ctx *be arbitrary contexts and* A : Ty *an arbitrary type. There is an operation* $\Gamma \vdash A \to \Delta, \Gamma \vdash A$ *on terms of* λ_{IK} *(and* λ_{IS4}*), where* Δ, Γ *denotes context concatenation.*

Proof. By recursion on terms.

And, secondly, we note that also a converse of this lemma holds by inspection of normal forms:

Lemma 3. *Let* Δ, Γ : Ctx *be arbitrary contexts,* A : Ty *an arbitrary type and* $t : \Delta, \Gamma \vdash A$ *a term of* λ_{IK} *(or* λ_{IS4}*) in the concatenated context* Δ, Γ *that does* not *mention any variables from* Δ*, then there is a term* $t' : \Gamma \vdash A$ *of* λ_{IK} *(or* λ_{IS4}*, respectively).*

Proof. Since normalization (Theorem 2 and Conjecture 1) does not introduce new free variables it suffices to prove the statement for terms in normal form. We do so by induction on normal forms $n : \Delta, \Gamma \vdash_{\mathrm{NF}} A$ *(see Fig. 8). The only nonimmediate step is for* n *of the form* $\mathsf{unbox}\, n'\, e$ *for some neutral element* $n' : \Delta' \vdash_{\mathrm{NE}} \Box A$ *and* $\Delta' \sqsubseteq \Delta \leq \Delta, \Gamma$*. But in that case the induction hypothesis says that we have a neutral element* $n'' : \cdot \vdash_{\mathrm{NE}} \Box A$*, which is impossible.*

Theorem 10. *Let* A : Ty *be an arbitrary type. There is a term* $t : \cdot \vdash A$ *of* λ_{IK} *(or* λ_{IS4}*) if and only if there is a term* $u : \cdot \vdash \Box A$ *of* λ_{IK} *(or* λ_{IS4}*, respectively), where* $\cdot$: Ctx *denotes the* empty *context.*

Proof. From a term $t : \cdot \vdash A$ *we can construct a term* $t' : \cdot, \text{🔒} \vdash A$ *using Lemma 2 and thus the term* $u = \mathsf{box}\, t' : \cdot \vdash \Box A$*.*

In the other direction, from a term $u : \cdot \vdash \Box A$ *we obtain a normal form* $u' = \mathrm{norm}\, u : \cdot \vdash_{\mathrm{NF}} \Box A$ *using Theorem 2 and Conjecture 1. By inspection of normal forms (see Fig. 8) we know that* u' *must be of the form* $\mathsf{box}\, v$ *for some normal form* $v : \cdot, \text{🔒} \vdash_{\mathrm{NF}} A$*, from which we obtain a term* $t : \cdot \vdash A$ *using Lemma 3 since the context* $\cdot, \text{🔒}$ *does not declare any variables that could have been mentioned in* v*.*

This concludes this section on some consequences of the model constructions presented in this paper.

[4] Invariance of truth under bisimulation says that if w and v are two bisimilar worlds in two possible-world models $\mathcal{M}_0$ and $\mathcal{M}_1$, respectively, then for all formulas A it is the case that $[\![A]\!]_w$ holds in $\mathcal{M}_0$ if and only if $[\![A]\!]_v$ does in $\mathcal{M}_1$.

5 Programming-Language Applications

In this section, we discuss some applications of NbE for Fitch-calculi in the context of programming languages. Concretely, we use NbE to show meta-theoretic results, e.g. noninterference, and normalization as a tool, e.g. for staged computations.

5.1 Escaping Computational Effects

Choudhury and Krishnaswami [8] present a modal type system based on IS4 for a programming language with implicit effects in the style of the computational lambda calculus [24]. In their language, programs need access to capabilities in order to produce computational effects. For example, the primitive print takes two arguments: the string to print and a capability of type Cap. Then, a program of type $\square A$ is a program of type A that does not access any capability in context; the program is safe.

Even languages where computational effects are explicit on the type level, e.g. Haskell with the IO monad, can benefit from a mechanism to safely escape computations. For example, currently Haskell programmers have to use the primitive *unsafePerformIO* to escape IO, which under many circumstances is not safe (as its name clearly indicates). If the IO computation is known to not produce effects, e.g. printing because it does not access capabilities, then the type $\square$ could provide a safe escape hatch from IO.

Motivated by this, we study purity in the context of a language that extends Moggi's monadic metalanguage with λ_{IS4}. In this system there is a type T of computations, that is, values of type $\mathsf{T}\,A$ are computations that might produce side-effects and finally return a value of type A, and programs perform printing effects via a primitive $\mathsf{print} : \mathsf{Cap} \Rightarrow \mathsf{String} \Rightarrow \mathsf{T\,Unit}$. A computation denied of access to capabilities does not print:

Definition 3 (Purity). *A program* $c : \mathsf{Cap} \vdash f : \square(\mathsf{T\,Unit})$ *is* pure *if* $c : \mathsf{Cap} \vdash f \sim \mathsf{box}\,(\mathsf{return\,unit})$.

Building on normalization results for the metalanguage [21], we use NbE to show that all programs are pure:

Theorem 11. *Any program* $c : \mathsf{Cap} \vdash f : \square(\mathsf{T\,Unit})$ *is pure.*

Theorem 11 follows by correctness of normalization and inspection of the normal forms of that type: that is, the normal form $c : \mathsf{Cap} \vdash_{\mathrm{NF}} f : \square(\mathsf{T\,Unit})$ is (syntactically) equal to $c : \mathsf{Cap} \vdash_{\mathrm{NF}} \mathsf{box}\,(\mathsf{return\,unit})$.

Fitch-style λ_{IS4} is a natural fit for purity and effects. To validate their intuition, Choudhury and Krishnaswami construct the *capability spaces* model where $\square A$ removes all elements of type A with access to capabilities. This functor has a left adjoint 🔒 that denies access to capabilities.

5.2 Information-Flow Control

Information-flow control (IFC) [26] protects confidentiality of data by controlling how information is allowed to flow in programs. In static IFC (e.g. [1, 27]), types serve as the security specification, i.e. to specify what inputs of a program are *secret* or *public*. Then, the type system enforces that information flows according to the given security policy, i.e. *secret* inputs should not interfere with *public* outputs. This is usually formalized as some sort of *noninterference* property [13].

Tomé Cortiñas and Valliappan [28] illustrate the use of normalization for proving that an IFC language, based on Moggi's monadic metalanguage [24], is secure. Their idea is simple: reason about the behaviour of classes of programs by looking at their normal forms, which are syntactically simpler. Correctness of normalization ensures that the normal form of a program $\Gamma \vdash t : A$ is semantically equivalent to t, i.e. $\mathrm{norm}\, t \sim t$.

Without committing to a particular Fitch-style calculus, we identify $\Box\mathsf{Bool}$ as the type of secret Booleans and Bool as the type of public Booleans. We state *noninterference* for Fitch-style as follows.

Definition 4 (Noninterference). *A program* $\cdot \vdash f : \Box\mathsf{Bool} \to \mathsf{Bool}$ *is* noninterferent*, if for any two secrets* $\cdot \vdash s_1, s_2 : \Box\mathsf{Bool}$*, it is the case that* $\cdot \vdash \mathsf{app}\, f\, s_1 \sim \mathsf{app}\, f\, s_2 : \mathsf{Bool}$.

A calculus satisfies noninterference *if all programs* $\cdot \vdash f : \Box\mathsf{Bool} \to \mathsf{Bool}$ *are noninterferent.*

Note that some of the Fitch-style calculi that we study satisfy noninterference, e.g. λ_{IK}, while others do not, e.g. λ_{IS4}.

The Calculus λ_{IK} Satisfies Noninterference

Theorem 12. *All λ_{IK} programs $\cdot \vdash f : \Box\mathsf{Bool} \to \mathsf{Bool}$ are noninterferent.*

Proof. We show that such a program is equivalent to a constant function, which is clearly noninterferent.

The normal forms of type $\cdot \vdash_{\mathrm{NF}} \Box\mathsf{Bool} \to \mathsf{Bool}$ are of the shape $\cdot \vdash_{\mathrm{NF}} \lambda\, x.\, \_$ where the hole has type $\Box\mathsf{Bool} \vdash_{\mathrm{NF}} \_ : \mathsf{Bool}$. If the hole is true *or* false *we are done. If it is not then it cannot be of the shape* $\mathsf{unbox}\, \_$ *because the context does not contain 🔒. The remaining cases follow a similar argument.* $\Box$

The Calculus λ_{IS4} does not Satisfy Noninterference Axiom 4, i.e. $\vdash \lambda\, x.\, \mathsf{unbox}\, x : \Box A \Rightarrow A$, is an example of a program that is interferent.

5.3 Staged Computation

Davies and Pfenning [11, 12] introduce a modal type system based on IS4 and argue that it is suitable for staged computation. In their system, the type $\Box A$ represents *code* of type A and the axioms of IS4 correspond to operations for code manipulation: $\mathrm{K} : \Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B)$ for substituting code in code, T : $\Box A \Rightarrow A$ for evaluating code, and 4 : $\Box A \Rightarrow \Box\Box A$ for making code available on subsequent stages.

One of their desiderata is stage separation, namely, code should *only* depend on code. As an example, they show that the term $\lambda\, x.\, \mathsf{box}\, x$, which violates stage separation, is not typeable.

We use NbE to prove a type-based stage separation theorem for λ_{IS4}:

Theorem 13 (Stage separation). *It is not the case that for all types A there is a term $\cdot \vdash A \Rightarrow \Box A$.*

Proof. Let $A = \iota$ be the witness. By inspection, in context $x : \iota$ there are no normal forms of type $\Box\iota$, and hence, there are no terms of type $\cdot \vdash \iota \Rightarrow \Box\iota$. □

Note that, for example, for some types like Bool there are nontrivial terms of type $\cdot \vdash \mathsf{Bool} \Rightarrow \Box\mathsf{Bool}$ [19, Section 4.3]:

$$\cdot \vdash \lambda\, x.\, \mathsf{ifte}\, x\, (\mathsf{box}\, \mathsf{true})\, (\mathsf{box}\, \mathsf{false}) : \mathsf{Bool} \Rightarrow \Box\mathsf{Bool}$$

Staged Evaluation of Programs with Free Variables Davies and Pfenning connect the logic IS4 to staged evaluation by defining a reduction relation on closed terms that applies β reduction everywhere but leaves "uninterpreted code", i.e. terms of the form $\mathsf{box}\, t$, unchanged.

Different from Davies and Pfenning, who disable the congruence rule for the box term, we can reuse the normalization function for λ_{IS4} (subsection 3.2) to normalize open terms at every stage. Note that normalization performs both β reductions and η expansions.

To finalize, we prove a theorem analogous to Davies and Pfenning [12, Theorem 5] but adapted to NbE for Fitch-style λ_{IS4}. This theorem states that after program staging the resulting term is "irreducible", that is, at least the first stage has been completely eliminated.

Theorem 14 (Eliminability). *If $\Gamma \vdash t : \Box A$ then* $\mathrm{norm}\, t$ *is of the form $\Gamma \vdash \mathsf{box}\, t' : \Box A$ for some t'.*

6 Related and Further Work

Fitch-style Calculi, Normalization and Their Semantics Fitch-style modal type systems [6, 22] adapt the proof methods of Fitch-style natural deduction systems for modal logic. In a Fitch-style natural deduction system, to eliminate a formula of type $\Box A$, we open a so-called strict subordinate proof and apply an "import" rule to produce a formula A. Fitch-style lambda calculi achieve a similar effect, for example in λ_{IK}, by adding a 🔒 to the context. To introduce a formula of type $\Box A$, on the other hand, we close a strict subordinate proof, and apply an "export" rule to a formula of type A—which corresponds to removing a 🔒 from the context. In the possible-world reading, adding a 🔒 corresponds to travelling to a future world, and removing it corresponds to returning to the original world.

The Fitch-style calculus λ_{IK} was presented for the logic IK by Borghuis [6], and later investigated further by Clouston [9]. Clouston showed that 🔒 can be interpreted as the left adjoint of $\Box$, and proves a completeness result for a term calculus that extends λ_{IK} with a type former $\blacklozenge$ that internalizes 🔒. The extended term calculus is, however, somewhat unsatisfactory since the normal forms do not enjoy the *subformula property*. Normalization was also considered by Clouston, but only with Rule $\Box$-β and not Rule $\Box$-η. The normalization result presented here considers both rules, and the corresponding completeness result achieved using the NbE model does not require the extension of λ_{IK} with $\blacklozenge$. The decidability result that follows for the complete equational theory of λ_{IK} also appears to have been an open problem prior to our work.

For the logic IS4, there appear to be several possible formulations of a Fitch-style calculus, where the difference has to do with the definition of the rule $\Box$-Elim/λ_{IS4}. One possibility is to define unbox by explicitly recording the context extension as a part of the term former. Davies and Pfenning [11] present such a system where they annotate the term former unbox as unbox_n to denote the number of 🔒s[5] added to the resulting context. Another possibility is to define unbox without any explicit annotations, thus leaving it ambiguous and to be inferred from a specific typing derivation. Such a system is presented by Clouston [9], and also discussed by Davies and Pfenning. The primary difference lies in their semantic interpretation: in the latter option, Clouston shows that $\Box$ can be interpreted as an idempotent comonad, i.e. $\Box\Box A \cong \Box A$, while this is not the case with the former—although it can be shown that $\Box\Box A \leftrightarrow \Box A$. The λ_{IS4} calculus presented here falls under the former category, where we record the extension explicitly using a premise instead of an annotation.

Gratzer, Sterling, and Birkedal [16] present yet another possibility that reformulates the system for IS4 in Clouston [9]. They further extend it with dependent types, and also prove a normalization result with respect to an equational theory that includes both Rules $\Box$-β and $\Box$-η using NbE. Although their approach is semantic in the sense of using NbE, their semantic domain has a very syntactic flavour [16, Section 3.2] that obscures the elegant possible-world interpretation. For example, it is unclear as to how their NbE algorithm can be adapted to minor variations in the syntax such as in λ_{IK}, λ_{IK4} and λ_{IT}—a solution to which is at the very core of our pursuit. This difference also has to do with the fact that they are interested in NbE for type-checking (also called "untyped" or "defunctionalized" NbE), while we are interested in NbE for well-typed terms (and thus "typed" NbE), which is better suited for studying the underlying models. Furthermore, we also avoid several complications that arise in accommodating dependent types in a Fitch-style calculus, which is the main goal of their work.

Possible-World semantics Given that Fitch-style natural deduction for modal logic has itself been motivated by possible-world semantics, it is only natural that Fitch-style calculi can also be given possible-world semantics. It appears to be roughly understood that the 🔒 operator models some notion of a past world, but this has not been—to the best of our knowledge—made precise with a

[5] Precisely, the number of *stack frames*, since their presentation uses a stack of contexts, as opposed to a single context with a first-class delimiting operator 🔒

concrete definition that is supported by a soundness and completeness result. As noted earlier, this requires a minor refinement of the frame conditions that define possible-world models for intuitionistic modal logic given by Božić and Došen [7].

Dual-Context Calculi Dual-context calculi [25, 12, 18] provide an alternative approach to programming with the necessity modality using judgements of the form $\Delta; \Gamma \vdash A$ where Δ is thought of as the modal context and Γ as the usual (or "local") one. As opposed to a "direct" eliminator as in Fitch-style calculi, dual-context calculi feature a pattern-matching eliminator formulated as a let-construct. The let-construct allows a type $\Box A$ to be eliminated into an arbitrary type C, which induces an array of commuting conversions in the equational theory to attain normal forms that obey the subformula property. Furthermore, the inclusion of an η-law for the $\Box$ type former complicates the ability to produce a unique normal form. Normalization (and, more specifically, NbE) for a pattern-matching eliminator—while certainly achievable—is a much more tedious endeavour, as evident from the work on normalizing sum types [4, 20, 3], which suffer from a similar problem. Presumably for this reason, none of the existing normalization results for dual-context calculi consider the η-law. The possible-world semantics of dual-context calculi is also less apparent, and it is unclear how NbE models can be constructed as instances of that semantics.

Multimodal Type Theory (MTT) Gratzer et al. [15] present a multimodal dependent type theory that for every choice of mode theory yields a dependent type theory with multiple interacting modalities. In contrast to Fitch-style calculi, their system features a variable rule that controls the use of variables of modal type in context. Further, the elimination rule for modal types is formulated in the style of the let-construct for dual-context calculi. In a recent result, Gratzer [14] proves normalization for multimodal type theory. In spite of the generality of multimodal type theory, it is worth noting that the normalization problem for Fitch-style calculi, when considering the full equational theory, is not a special case of normalization for multimodal type theory.

Further Modal Axioms The possible-world semantics and NbE models presented here only consider the logics IK, IT, IK4 and IS4. We wonder if it would be possible to extend the ideas presented here to further modal axioms such as $\mathrm{R} : A \to \Box A$ and $\mathrm{GL} : \Box(\Box A \to A) \to \Box A$, especially considering that the calculi may differ in more than just the elimination rule for the $\Box$ type.

References

1. Abadi, M., Banerjee, A., Heintze, N., and Riecke, J.G.: A Core Calculus of Dependency. In: Appel, A.W., and Aiken, A. (eds.) POPL '99, Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, TX, USA, January 20-22, 1999, pp. 147–160. ACM (1999). DOI: 10.1145/292540.292555. https://doi.org/10.1145/292540.292555

2. Abel, A., Allais, G., Cockx, J., Danielsson, N.A., Hausmann, P., Nordvall Forsberg, F., Norell, U., López Juan, V., Sicard-Ramírez, A., and Vezzosi, A., *Agda 2* version 2.6.1.3, 2005–2021. Chalmers University of Technology and Gothenburg University. LIC: BSD3. URL: `https://wiki.portal.chalmers.se/agda/pmwiki.php`, VCS: `https://github.com/agda/agda`.
3. Abel, A., and Sattler, C.: Normalization by Evaluation for Call-By-Push-Value and Polarized Lambda Calculus. In: Komendantskaya, E. (ed.) Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019, 3:1–3:12. ACM (2019). DOI: `10.1145/3354166.3354168`. `https://doi.org/10.1145/3354166.3354168`
4. Altenkirch, T., Dybjer, P., Hofmann, M., and Scott, P.J.: Normalization by Evaluation for Typed Lambda Calculus with Coproducts. In: 16th Annual IEEE Symposium on Logic in Computer Science, Boston, Massachusetts, USA, June 16-19, 2001, Proceedings, pp. 303–310. IEEE Computer Society (2001). DOI: `10.1109/LICS.2001.932506`. `https://doi.org/10.1109/LICS.2001.932506`
5. Berger, U., and Schwichtenberg, H.: An Inverse of the Evaluation Functional for Typed lambda-calculus. In: Proceedings of the Sixth Annual Symposium on Logic in Computer Science (LICS '91), Amsterdam, The Netherlands, July 15-18, 1991, pp. 203–211. IEEE Computer Society (1991). DOI: `10.1109/LICS.1991.151645`. `https://doi.org/10.1109/LICS.1991.151645`
6. Borghuis, V.A.J.: Coming to terms with modal logic. Technische Universiteit Eindhoven, Eindhoven (1994). On the interpretation of modalities in typed λ-calculus, Dissertation, Technische Universiteit Eindhoven, Eindhoven, 1994
7. Božić, M., and Došen, K.: Models for normal intuitionistic modal logics. Studia Logica **43**(3), 217–245 (1984). DOI: `10.1007/BF02429840`. `https://doi.org/10.1007/BF02429840`
8. Choudhury, V., and Krishnaswami, N.: Recovering purity with comonads and capabilities. Proc. ACM Program. Lang. **4**(ICFP), 111:1–111:28 (2020). DOI: `10.1145/3408993`. `https://doi.org/10.1145/3408993`
9. Clouston, R.: Fitch-Style Modal Lambda Calculi. In: Baier, C., and Lago, U.D. (eds.) Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings. LNCS, vol. 10803, pp. 258–275. Springer, Heidelberg (2018). DOI: `10.1007/978-3-319-89366-2\_14`. `https://doi.org/10.1007/978-3-319-89366-2%5C_14`
10. Coquand, C.: A Formalised Proof of the Soundness and Completeness of a Simply Typed Lambda-Calculus with Explicit Substitutions. High. Order Symb. Comput. **15**(1), 57–90 (2002). DOI: `10.1023/A:1019964114625`. `https://doi.org/10.1023/A:1019964114625`
11. Davies, R., and Pfenning, F.: A Modal Analysis of Staged Computation. In: Boehm, H., and Jr., G.L.S. (eds.) Conference Record of POPL'96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Papers Presented at the Symposium, St. Petersburg Beach, Florida, USA, January 21-24, 1996, pp. 258–270. ACM Press (1996). DOI: `10.1145/237721.237788`. `https://doi.org/10.1145/237721.237788`
12. Davies, R., and Pfenning, F.: A modal analysis of staged computation. J. ACM **48**(3), 555–604 (2001). DOI: `10.1145/382780.382785`. `https://doi.org/10.1145/382780.382785`

13. Goguen, J.A., and Meseguer, J.: Security Policies and Security Models. In: 1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982, pp. 11–20. IEEE Computer Society (1982). DOI: 10.1109/SP.1982.10014. https://doi.org/10.1109/SP.1982.10014
14. Gratzer, D.: Normalization for multimodal type theory. arXiv preprint arXiv:2106.01414 (2021)
15. Gratzer, D., Kavvos, G.A., Nuyts, A., and Birkedal, L.: Multimodal Dependent Type Theory. In: Hermanns, H., Zhang, L., Kobayashi, N., and Miller, D. (eds.) LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020, pp. 492–506. ACM (2020). DOI: 10.1145/3373718.3394736. https://doi.org/10.1145/3373718.3394736
16. Gratzer, D., Sterling, J., and Birkedal, L.: Implementing a modal dependent type theory. Proc. ACM Program. Lang. **3**(ICFP), 107:1–107:29 (2019). DOI: 10.1145/3341711. https://doi.org/10.1145/3341711
17. Jay, C.B., and Ghani, N.: The Virtues of Eta-Expansion. J. Funct. Program. **5**(2), 135–154 (1995). DOI: 10.1017/S0956796800001301. https://doi.org/10.1017/S0956796800001301
18. Kavvos, G.A.: Dual-Context Calculi for Modal Logic. Log. Methods Comput. Sci. **16**(3) (2020). https://lmcs.episciences.org/6722
19. Kavvos, G.A.: Modalities, cohesion, and information flow. Proc. ACM Program. Lang. **3**(POPL), 20:1–20:29 (2019). DOI: 10.1145/3290333. https://doi.org/10.1145/3290333
20. Lindley, S.: Extensional Rewriting with Sums. In: Rocca, S.R.D. (ed.) Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings. LNCS, vol. 4583, pp. 255–271. Springer, Heidelberg (2007). DOI: 10.1007/978-3-540-73228-0_19. https://doi.org/10.1007/978-3-540-73228-0%5C_19
21. Lindley, S.: Normalisation by evaluation in the compilation of typed functional programming languages. University of Edinburgh, UK (2005)
22. Martini, S., and Masini, A.: A computational interpretation of modal proofs. In: Proof theory of modal logic, pp. 213–241. Springer (1996)
23. Miyamoto, K., and Igarashi, A.: A modal foundation for secure information flow. In: In Proceedings of IEEE Foundations of Computer Security (FCS), pp. 187–203 (2004)
24. Moggi, E.: Notions of Computation and Monads. Inf. Comput. **93**(1), 55–92 (1991). DOI: 10.1016/0890-5401(91)90052-4. https://doi.org/10.1016/0890-5401(91)90052-4
25. Pfenning, F., and Davies, R.: A judgmental reconstruction of modal logic. Math. Struct. Comput. Sci. **11**(4), 511–540 (2001). DOI: 10.1017/S0960129501003322. https://doi.org/10.1017/S0960129501003322
26. Sabelfeld, A., and Myers, A.C.: Language-based information-flow security. IEEE J. Sel. Areas Commun. **21**(1), 5–19 (2003). DOI: 10.1109/JSAC.2002.806121. https://doi.org/10.1109/JSAC.2002.806121
27. Shikuma, N., and Igarashi, A.: Proving Noninterference by a Fully Complete Translation to the Simply Typed Lambda-Calculus. Log. Methods Comput. Sci. **4**(3) (2008). DOI: 10.2168/LMCS-4(3:10)2008. https://doi.org/10.2168/LMCS-4(3:10)2008
28. Tomé Cortiñas, C., and Valliappan, N.: Simple Noninterference by Normalization. In: Proceedings of the 14th ACM SIGSAC Workshop on Programming Languages and Analysis for Security. PLAS'19, pp. 61–72. Association for Computing Machinery,

London, United Kingdom (2019). DOI: `10.1145/3338504.3357342`. `https://doi.org/10.1145/3338504.3357342`