Normalization for Fitch-style Modal Calculi (Draft)*

NACHIAPPAN VALLIAPPAN, Chalmers University of Technology, Sweden
FABIAN RUCH, Unaffiliated, Sweden
CARLOS TOMÉ CORTIÑAS, Chalmers University of Technology, Sweden

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.

Additional Key Words and Phrases: 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 [Davies and Pfenning 1996], information-flow control [Miyamoto and Igarashi 2004], and recovering purity in an effectful language [Choudhury and Krishnaswami 2020]. 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 [Borghuis 1994; Clouston 2018; Martini and Masini 1996] 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 $\Box$, along with introduction and elimination rules for $\Box$ types formulated using the 🔒 operator. For instance, the calculus λ_{IK}, which corresponds to the logic IK, extends STLC with Rules $\Box$-Intro and $\lambda_{\text{IK}}/\Box$-Elim, 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!

*Updated 17 June 2022

$$\mathit{Ty} \quad A ::= \ldots \mid \Box A \qquad\qquad \mathit{Ctx} \quad \Gamma ::= \cdot \mid \Gamma, x : A \mid \Gamma, \blacksquare$$

$$\textsc{Var}\ \frac{}{\Gamma, x : A, \Gamma' \vdash x : A}\ \blacksquare \notin \Gamma' \qquad \Box\textsc{-Intro}\ \frac{\Gamma, \blacksquare \vdash t : A}{\Gamma \vdash \mathsf{box}\, t : \Box A} \qquad \lambda_{\mathrm{IK}}/\Box\textsc{-Elim}\ \frac{\Gamma \vdash t : \Box A}{\Gamma, \blacksquare, \Gamma' \vdash \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, t : A}\ \blacksquare \notin \Gamma'$$

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 $\Box$ types, where the former states the β- (or computational) equivalence, and the latter states a type-directed η- (or extensional) equivalence.

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

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 $\Box$-β equivalence from left to right. This naive approach to formulating a rewrite rule, however, is insufficient for the $\Box$-η rule since normalizing with a rewrite rule $t \mapsto \mathsf{box}\,(\mathsf{unbox}\, t)$ (for $\Gamma \vdash t : \Box A$) does not terminate as it can be applied infinitely many times. It is presumably for this reason that existing normalization results [Clouston 2018] 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 [1995] 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 term equivalences, each

$$\lambda_{\mathrm{IT}}/\Box\textsc{-Elim}\ \frac{\Gamma \vdash t : \Box A}{\Gamma, \Gamma' \vdash \mathsf{unbox}_{\lambda_{\mathrm{IT}}}\, t : A}\ \#_{\blacksquare}(\Gamma') \leq 1 \qquad \lambda_{\mathrm{IK4}}/\Box\textsc{-Elim}\ \frac{\Gamma \vdash t : \Box A}{\Gamma, \blacksquare, \Gamma' \vdash \mathsf{unbox}_{\lambda_{\mathrm{IK4}}}\, t : A} \qquad \lambda_{\mathrm{IS4}}/\Box\textsc{-Elim}\ \frac{\Gamma \vdash t : \Box A}{\Gamma, \Gamma' \vdash \mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t : A}$$

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

calculus demands different, tedious and seemingly ad hoc syntactic renaming lemmas [Clouston 2018, Lemmas 4.1 and 5.1] to prove normalization.

In this paper, we take a semantic approach to normalization, called normalization by evaluation (NbE) [Berger and Schwichtenberg 1991]. 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 calculi λ_{IK} and λ_{IS4} and their normalization algorithms have been implemented and verified correct in the proof assistant Agda [Abel, Allais, et al. 2005–2021].

NbE models and proofs of normalization in 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 properties that typically require semantic intervention can be proved syntactically. We show that properties similar to capability safety, noninterference, and binding-time correctness can be proved syntactically using normal forms of terms.

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. The possible-world semantics for intuitionistic modal logic [Božić and Došen 1984] 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 of *worlds* along with two binary *accessibility* relations R_i (for "intuitionistic") and R_m (for "modal") 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. Then, $w\ R_i\ w'$ represents an increase in knowledge from w to w', and $w\ R_m\ v$ represents a possible passage from w to v. A valuation V_ι, on the other hand, is a family of types $V_{\iota,w}$ indexed by $w : W$ along with functions $wk_{\iota,w,w'} : V_{\iota,w} \to V_{\iota,w'}$ whenever $w\ R_i\ w'$. An element $p : V_{\iota,w}$ can be thought of as "evidence" for (the knowledge of) the truth of the *atomic* proposition ι at the world w. The requirement for functions $wk_{\iota,w,w'}$ enforces that the knowledge of the truth of ι at w is preserved as time moves on to w', and is neither forgotten nor contradicted by any new evidence learned at w'. There are no such requirements on a valuation V_ι with respect to the modal accessibility relation R_m.

Given a frame (W, R_i, R_m) and a valuation V_ι, we interpret (object) types A in *any* Fitch-style calculus as families of (meta) types $[\![A]\!]_w$ indexed by worlds $w : W$, following the work by Ewald [1986], Fischer-Servi [1981], Plotkin and Stirling [1986], and Simpson [1994] as below:

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

The nonmodal type formers are interpreted as in the Kripke semantics for intuitionistic propositional logic: The base type ι is interpreted using the valuation V_ι, and function types $A \Rightarrow B$ at $w : W$ are interpreted as *families* of functions $[\![A]\!]_{w'} \to [\![B]\!]_{w'}$ indexed by $w' : W$ such that $w \; R_i \; w'$. Recall that the generalization to families is necessary for the interpretation of function types to be sound.

As for the interpretation of modal types, at $w : W$ the types $\Box A$ are interpreted by families of elements $[\![A]\!]_v$ indexed by those $v : W$ that are accessible from w via some $w' : W$ such that $w \; R_i \; w'$ and $w' \; R_m \; v$. In other words, $\Box A$ is true at a world w if A is necessarily true in "the future", whichever concrete possibility this may turn out to be. We remark that the interpretation of $\Box A$ as $\forall v.\, w \; R_m \; v \to [\![A]\!]_v$, as in classical modal logic without the first quantifier $\forall w'.\, w \; R_i \; w'$, requires additional conditions [Božić and Došen 1984; Simpson 1994] on frames that (some of) the NbE models we construct do not satisfy.

In order to extend the possible-world semantics of intuitionistic modal logic to Fitch-style calculi, we must also provide an interpretation of contexts and the 🔒 operator, which is unique to the Fitch style, in particular:

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

The empty context $\cdot$ and the context extension Γ, A of a context Γ with a type A are interpreted as in the Kripke semantics for STLC by the terminal family and the Cartesian product of the families $[\![\Gamma]\!]$ and $[\![A]\!]$, respectively. While the interpretation of types $\Box A$ can be understood as a statement about the future, the interpretation of contexts $\Gamma, \text{🔒}$ can be understood as a dual statement about the past: $\Gamma, \text{🔒}$ is true at a world w if Γ is true at *some* world u for which w is a possibility, i.e. $u \; R_m \; w$.

With the interpretation of contexts Γ and types A as (W, R_i)-indexed families $[\![\Gamma]\!]$ and $[\![A]\!]$ at hand, the interpretation of terms $t : \Gamma \vdash A$, also known as *evaluation*, in a possible-world model is given by a function $[\![-]\!] : \Gamma \vdash A \to (\forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w)$ as follows. Clouston [2018] shows that the interpretation of STLC in Cartesian closed categories (CCCs) extends to an interpretation of Fitch-style calculi in any CCC equipped with an adjunction by interpreting $\Box$ and 🔒 by the right and left adjoint as well as box and unbox using the right and left adjuncts, respectively. The key idea here is that, correspondingly, the interpretation of terms in the nonmodal fragment of Fitch-style calculi using the familiar CCC structure on (W, R_i)-indexed families extends to the modal fragment: the interpretation of $\Box$ in a possible-world model has a left adjoint that is denoted by our interpretation of 🔒. In summary, the possible-world interpretation of Fitch-style calculi can be given by instantiation of Clouston's *generic* interpretation in CCCs equipped with an adjunction.

Constructing NbE Models as Instances. To construct an NbE model for Fitch-style calculi, we must construct a possible-world model with a function $\mathit{quote} : (\forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w) \to \Gamma \vdash_{\mathrm{NF}} A$ that inverts the denotation $(\forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w)$ of a term 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.*

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

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[1] models [Coquand 2002], we instantiate the parameters that define possible-world models for Fitch-style calculi as follows: we pick contexts for W, *order-preserving embeddings* (sometimes 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 difference between them. Recall that the syntaxes of the four calculi only differ in their elimination rules for $\Box$ types. When viewed through the lens of the possible-world semantics, this difference can be generalized as follows:

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

We generalize the relationship between the context in the premise and the context in the conclusion using a generic modal accessibility relation $\lhd$ 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 Δ, then we may derive A in the current world Γ. The various $\Box$-elimination rules for Fitch-style calculi can be viewed as instances of this generalized rule, where we define $\lhd$ 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 λ_{IK}/$\Box$-Elim is Γ and that of the conclusion is $\Gamma, \text{🔒}, \Gamma'$ such that $\text{🔒} \notin \Gamma'$, and thus define $\Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma$ as $\exists \Delta'. \text{🔒} \notin \Delta' \wedge \Gamma = \Delta, \text{🔒}, \Delta'$. Similarly, we define $\Delta \lhd_{\lambda_{\mathrm{IS4}}} \Gamma$ as $\exists \Delta'. \Gamma = \Delta, \Delta'$ 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 $\lhd$ 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 $\mathit{reify}_A : \forall \Gamma.\, [\![A]\!]_\Gamma \to \Gamma \vdash_{\mathrm{NF}} A$ 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 $\mathit{reify}_{\Box A} : \forall \Gamma.\, [\![\Box A]\!]_\Gamma \to \Gamma \vdash_{\mathrm{NF}} \Box A$. To reify a value of $[\![\Box A]\!]_\Gamma$, we first observe that $[\![\Box A]\!]_\Gamma = \forall \Gamma'. \Gamma \leq \Gamma' \to \forall \Delta. \Gamma' \lhd \Delta \to [\![A]\!]_\Delta$ by definition of $[\![-]\!]$ and the instantiations of R_i with $\leq$ and R_m with $\lhd$. By picking Γ for Γ' and $\Gamma, \text{🔒}$ for Δ, we get $[\![A]\!]_{\Gamma,\text{🔒}}$ since $\leq$ is reflexive and it can be shown that $\Gamma \lhd \Gamma, \text{🔒}$ holds 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.

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

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

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

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}

3.1.1 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 $\Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma$, which expresses that context Γ extends $\Delta, \text{🔒}$ to the right without adding locks. Note that $\Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma$ exactly when $\exists \Delta'. \text{🔒} \notin \Delta' \wedge \Gamma = \Delta, \text{🔒}, \Delta'$.

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

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

$$\textsc{Var-Zero}\ \ \Gamma, A \vdash_{\mathrm{VAR}} \mathsf{zero} : A \qquad \textsc{Var-Succ}\ \frac{\Gamma \vdash_{\mathrm{VAR}} v : A}{\Gamma, B \vdash_{\mathrm{VAR}} \mathsf{succ}\, v : A} \qquad \textsc{Var}\ \frac{\Gamma \vdash_{\mathrm{VAR}} v : A}{\Gamma \vdash \mathsf{var}\, v : A} \qquad {\Rightarrow}\textsc{-Intro}\ \frac{\Gamma, A \vdash t : B}{\Gamma \vdash \lambda\, t : A \Rightarrow B}$$

$${\Rightarrow}\textsc{-Elim}\ \frac{\Gamma \vdash t : A \Rightarrow B \qquad \Gamma \vdash u : A}{\Gamma \vdash \mathsf{app}\, t\, u : B} \qquad \Box\textsc{-Intro}\ \frac{\Gamma, \text{🔒} \vdash t : A}{\Gamma \vdash \mathsf{box}\, t : \Box A} \qquad \lambda_{\mathrm{IK}}/\Box\textsc{-Elim}\ \frac{\Delta \vdash t : \Box A \qquad e : \Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma}{\Gamma \vdash \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, t\, e : A}$$

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

Fig. 5 presents the intrinsically-typed syntax of λ_{IK}. We will use both $\Gamma \vdash t : A$ and $t : \Gamma \vdash A$ to say that t denotes an (intrinsically-typed) term of type A in context Γ, and similarly for substitutions. 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 $\lambda_{\mathrm{IK}}/\Box$-Elim is defined using the modal accessibility relation $\Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma$ 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 $e : \Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma$ as a second premise is in sharp contrast to Clouston [2018, 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'.

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

$$\Gamma \vdash_{\mathrm{s}} \mathrm{empty} : \cdot \qquad \frac{\Gamma \vdash_{\mathrm{s}} s : \Delta \qquad \Gamma \vdash t : A}{\Gamma \vdash_{\mathrm{s}} \mathrm{ext}\ s\ t : \Delta, A} \qquad \frac{\Theta \vdash_{\mathrm{s}} s : \Delta \qquad e : \Theta \lhd_{\lambda_{\mathrm{IK}}} \Gamma}{\Gamma \vdash_{\mathrm{s}} \mathrm{ext}_{\text{🔒}}\ s\ e : \Delta, \text{🔒}}$$

Fig. 6. Substitutions for λ_{IK}

Substitutions for λ_{IK} are inductively defined in Fig. 6. A judgement $\Gamma \vdash_{\mathrm{s}} s : \Delta$ denotes a substitution for a context Δ in the context Γ. Applying a substitution to a term $\Delta \vdash t : A$, i.e. $\mathit{subst}\ s\ t : \Gamma \vdash A$, yields a term in the context Γ. The substitution $\mathrm{id}_{\mathrm{s}} : \Gamma \vdash_{\mathrm{s}} \Gamma$ denotes the identity substitution, which exists for all Γ. As usual, it can be shown that terms are closed under the application of a substitution, and that it preserves the identity, i.e. $\mathit{subst}\ \mathrm{id}_{\mathrm{s}}\ t = t$. 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 $\Box$ type (Rules $\Box$-β and $\Box$-η). The function $\mathit{factor} : \Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma \to \Delta, \text{🔒} \leq \Gamma$, in Rule $\Box$-β, maps an element of the modal accessibility relation $e : \Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma$ to an OPE $\Delta, \text{🔒} \leq \Gamma$. This is possible because the context Γ does not have any lock to the right of $\Delta, \text{🔒}$.

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

$$\Box\text{-}\beta\ \frac{\Delta, \text{🔒} \vdash t : A \qquad e : \Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma}{\Gamma \vdash \mathrm{unbox}_{\lambda_{\mathrm{IK}}}\ (\mathrm{box}\ t)\ e \sim \mathit{wk}\ (\mathit{factor}\ e)\ t} \qquad \Box\text{-}\eta\ \frac{\Gamma \vdash t : \Box A}{\Gamma \vdash t \sim \mathrm{box}\ (\mathrm{unbox}_{\lambda_{\mathrm{IK}}}\ t\ \mathrm{nil})}$$

Fig. 7. Equational theory for λ_{IK}

3.1.2 *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*.

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

- R_i is *reflexive* and *transitive*
- if $w\ R_m\ v$ and $v\ R_i\ v'$ then there exists some $w' : W$ such that $w\ R_i\ w'$ and $w'\ R_m\ v'$; this *factorization* condition can be pictured as an implication $R_m ; R_i \subseteq R_i ; R_m$ or diagrammatically as follows:

$$
\begin{tikzcd}
w' \arrow[r, dashed, "R_m"] & v' \\
w \arrow[u, dashed, "R_i"] \arrow[r, "R_m"] & v \arrow[u, "R_i"']
\end{tikzcd}
$$

(note that neither w' nor the proofs of relatedness are required to be unique, nor will they all be in the frames that we will consider)

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 W-indexed family $V_\iota : W \longrightarrow \mathit{Type}$ (called the *valuation* of the base type) such that $\forall w, w'.\, w\ R_i\ w' \longrightarrow V_{\iota,w} \longrightarrow V_{\iota,w'}$.

The types and typing contexts in λ_{IK} are interpreted in a possible-world model via the interpretation functions $[\![-]\!]$ defined in Section 2. To evaluate terms, we must first prove the 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 $wk_A : w\ R_i\ w' \longrightarrow [\![A]\!]_w \longrightarrow [\![A]\!]_{w'}$. And similarly, for every context Γ, a function $wk_\Gamma : w\ R_i\ w' \longrightarrow [\![\Gamma]\!]_w \longrightarrow [\![\Gamma]\!]_{w'}$.*

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

$$
\begin{array}{l}
[\![-]\!] : \Gamma \vdash A \longrightarrow (\forall w.\, [\![\Gamma]\!]_w \longrightarrow [\![A]\!]_w) \\
[\![\mathsf{var}\ v]\!]\ \gamma = \mathit{lookup}\ v\ \gamma \\
[\![\lambda\, t]\!]\ \gamma = \lambda i.\, \lambda a.\, [\![t]\!]\ (\mathit{wk}\ i\ \gamma, a) \\
[\![\mathsf{app}\ t\ u]\!]\ \gamma = ([\![t]\!]\ \gamma)\ \mathsf{id}_{\leq}\ ([\![u]\!]\ \gamma) \\
[\![\mathsf{box}\ t]\!]\ \gamma = \lambda i.\, \lambda m.\, [\![t]\!]\ (\mathit{wk}\ i\ \gamma, m) \\
[\![\mathsf{unbox}_{\lambda_{\text{IK}}}\ t\ e]\!]\ \gamma = [\![t]\!]\ \delta\ \mathsf{id}_{\leq}\ m \\
\qquad \text{where } (\delta, m) = \mathit{trim}_{\lambda_{\text{IK}}}\ \gamma\ e
\end{array}
$$

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 $\forall w'.\, w\ R_i\ w' \longrightarrow [\![A]\!]_{w'} \longrightarrow [\![B]\!]_{w'}$ 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 $a : [\![A]\!]_{w'}$. 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 $\forall w'.\, w\ R_i\ w' \longrightarrow \forall v.\, w'\ R_m\ v \longrightarrow [\![A]\!]_v$. Using the semantic arguments $i : w\ R_i\ w'$ and $m : w'\ R_m\ v$, we recursively evaluate the term $\Gamma, 🔒 \vdash t : A$ in the extended environment $(\mathit{wk}\ i\ \gamma, m) : [\![\Gamma, 🔒]\!]_v$, since $[\![\Gamma, 🔒]\!]_v = \sum_{w'} [\![\Gamma]\!]_{w'} \times w'\ R_m\ v$. On the other hand, the term $\Gamma \vdash \mathsf{unbox}_{\lambda_{\text{IK}}}\, t\, e : A$ with $e : \Delta \lhd_{\lambda_{\text{IK}}} \Gamma$ 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 $\mathit{trim}_{\lambda_{\text{IK}}} : [\![\Gamma]\!]_w \longrightarrow \Delta \lhd_{\lambda_{\text{IK}}} \Gamma \longrightarrow [\![\Delta, 🔒]\!]_w$ that projects γ to produce an environment $\delta : [\![\Delta]\!]_{v'}$ and a value $m : v' \; R_m \; w$. We evaluate t with δ and apply the resulting function of type $\forall v.\, v \; R_i \; v' \to \forall w.\, v' \; R_m \; w \to [\![A]\!]_w$ to $\mathrm{id}_{\leq}$ and 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. We note that the soundness proof relies on the possible-world models to satisfy coherence conditions that we have omitted from Definitions 1 and 2 but that will be satisfied by the NbE models. Specifically, W and R_i together with the transitivity and reflexivity proofs trans_i and refl_i for R_i need to form a category $\mathscr{W}$, i.e. trans_i needs to be associative and refl_i needs to be a unit for trans_i; the proofs of the factorization condition need to satisfy the functoriality laws $\mathit{factor}_i \, m \, (\mathit{refl}_i \, v) = \mathit{refl}_i \, w$, $\mathit{factor}_m \, m \, (\mathit{refl}_i \, v) = m$, $\mathit{factor}_i \, m \, (\mathit{trans}_i \, i \, j) = \mathit{trans}_i \, (\mathit{factor}_i \, m \, i) \, (\mathit{factor}_i \, m' \, j)$ and $\mathit{factor}_m \, m \, (\mathit{trans}_i \, i \, j) = \mathit{factor}_m \, m' \, j$ where $m' \coloneqq \mathit{factor}_m \, m \, i : w' \; R_m \; v'$ denotes the modal accessibility proof produced by the first factorization of $m : w \; R_m \; v$ and $i : v \; R_i \; v'$; and V_ι together with the monotonicity proof wk_ι needs to form a functor on the category $\mathscr{W}$, i.e. $\mathit{wk}_\iota \, (\mathit{refl}_i \, w)$ needs to be equal to the identity function on $V_{\iota,w}$ and $\mathit{wk}_\iota \, (\mathit{trans}_i \, i \, j)$ needs to be equal to the composite $\mathit{wk}_\iota \, j \circ \mathit{wk}_\iota \, i$.

Theorem 2. *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 the functions $[\![t]\!]$ and $[\![u]\!] : \forall w.\, [\![\Gamma]\!]_w \to [\![A]\!]_w$ as determined by $\mathcal{M}$ are equal.*

Proof. The underlying frame (W, R_i, R_m) of any possible-world model $\mathcal{M}$ can be seen as determining an adjunction on the category of presheaves indexed by the category whose objects are worlds $w : W$ and whose morphisms are proofs $i : w \; R_i \; w'$, and the interpretation $[\![-]\!]$ determined by $\mathcal{M}$ can be seen as factoring through that adjunction and the familiar Cartesian closed structure on presheaves as described in Clouston [2018, Section 2.3]. We can therefore conclude by applying Clouston [2018, Theorem 2.8 (with remark below)]. ◻

3.1.3 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/□-Intro and λ_{IK}/Ne/□-Elim.

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

$$\frac{\Gamma, \text{🔒} \vdash_{\mathrm{NF}} n : A}{\Gamma \vdash_{\mathrm{NF}} \mathsf{box}\, n : \Box A}\;\textsc{Nf/}\Box\textsc{-Intro} \qquad \frac{\Delta \vdash_{\mathrm{NE}} n : \Box A \qquad e : \Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma}{\Gamma \vdash_{\mathrm{NE}} \mathsf{unbox}_{\lambda_{\mathrm{IK}}}\, n\, e : A}\;\lambda_{\mathrm{IK}}\textsc{/Ne/}\Box\textsc{-Elim}$$

Fig. 8. Normal forms and neutral elements in 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 $\mathit{quote} : \mathcal{M}([\![\Gamma]\!], [\![A]\!]) \to \Gamma \vdash_{\mathrm{NF}} A$ satisfying $t \sim \mathit{quote}\, [\![t]\!]$ 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.

- Worlds as contexts: $W = \mathit{Ctx}$

- Relation R_i as order-preserving embeddings: $\Gamma\ R_i\ \Gamma' = \Gamma \leq \Gamma'$
- Relation R_m as extensions of a "locked" context: $\Delta\ R_m\ \Gamma = \Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma$
- Valuation V_ι as neutral elements: $V_{\iota,\Gamma} = \Gamma \vdash_{\mathrm{NE}} \iota$

The condition that the valuation must satisfy $\mathit{wk}_A : \Gamma \leq \Gamma' \rightarrow \Gamma \vdash_{\mathrm{NE}} A \rightarrow \Gamma' \vdash_{\mathrm{NE}} A$, for all types A, can be shown by induction on the neutral term $\Gamma \vdash_{\mathrm{NE}} A$. 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 given $\Delta \lhd_{\lambda_{\mathrm{IK}}} \Gamma$ and $\Gamma \leq \Gamma'$ there is a $\Delta' : \mathit{Ctx}$ such that $\Delta \leq \Delta'$ and $\Delta' \lhd_{\lambda_{\mathrm{IK}}} \Gamma'$,

$$\begin{tikzcd}
\Delta' \arrow[r, dashed, "\lhd_{\lambda_{\mathrm{IK}}}"] & \Gamma' \\
\Delta \arrow[u, dashed, "\leq"] \arrow[r, "\lhd_{\lambda_{\mathrm{IK}}}"] & \Gamma \arrow[u, "\leq"']
\end{tikzcd}$$

which can be shown by constructing a function 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 $\mathit{wk}_A : \Gamma \leq \Gamma' \rightarrow [\![A]\!]_\Gamma \rightarrow [\![A]\!]_{\Gamma'}$ and $\mathit{wk}_\Delta : \Gamma \leq \Gamma' \rightarrow [\![\Delta]\!]_\Gamma \rightarrow [\![\Delta]\!]_{\Gamma'}$, 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 $\mathit{reify}_A : [\![A]\!]_\Gamma \rightarrow \Gamma \vdash_{\mathrm{NF}} A$ and $\mathit{reflect}_A : \Gamma \vdash_{\mathrm{NE}} A \rightarrow [\![A]\!]_\Gamma$, 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.

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

$$\begin{array}{ll}
\mathit{reflect}_{A,\Gamma} : \Gamma \vdash_{\mathrm{NE}} A \rightarrow [\![A]\!]_\Gamma & \\
\mathit{reflect}_{\iota,\Gamma} & n = n \\
\mathit{reflect}_{A \Rightarrow B,\Gamma} & n = \lambda(o : \Gamma \leq \Gamma').\,\lambda a.\ \mathit{reflect}_{B,\Gamma}(\mathsf{app}\ (\mathit{wk}_{A \Rightarrow B}\ o\ n)\ (\mathit{reify}_{A,\Gamma'}\ a)) \\
\mathit{reflect}_{\Box A,\Gamma} & n = \lambda(o : \Gamma \leq \Gamma').\,\lambda(e : \Gamma' \lhd_{\lambda_{\mathrm{IK}}} \Delta).\ \mathit{reflect}_{A,\Delta}(\mathsf{unbox}_{\lambda_{\mathrm{IK}}}\ (\mathit{wk}_{\Box A}\ o\ n)\ e)
\end{array}$$

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

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

$$
\begin{aligned}
&\mathit{quote} : (\forall \Delta.\, [\![\Gamma]\!]_\Delta \longrightarrow [\![A]\!]_\Delta) \longrightarrow \Gamma \vdash_{\mathrm{NF}} A \\
&\mathit{quote}\, f = \mathit{reify}_{A,\Gamma}\, (f\ \mathit{freshEnv}_\Gamma)
\end{aligned}
$$

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

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 $a : [\![A]\!]_\Gamma$ as $L_A\ t\ a$. From a proof of $L_A\ t\ a$, it can be shown that $t \sim \mathit{reify}_A\ a$. This relation is extended to contexts as L_Δ, for some context Δ, which relates a substitution $\Gamma \vdash s : \Delta$ to its denotation $\delta : [\![\Delta]\!]_\Gamma$ as $L_\Delta\ s\ \delta$.

$$
\begin{aligned}
&L_{A,\Gamma} : \Gamma \vdash A \longrightarrow [\![A]\!]_\Gamma \longrightarrow \mathit{Type} \\
&L_{\iota,\Gamma} \qquad t\ n = t \sim \mathit{quote}\ n \\
&L_{A\Rightarrow B,\Gamma}\ t\ f = \forall \Gamma', o : \Gamma \leq \Gamma', u, a.\ L_{A,\Gamma'}\ u\ a \longrightarrow L_{B,\Gamma'}\ (\mathrm{app}\ (\mathit{wk}\ o\ t)\ u)\ (f\ o\ a) \\
&L_{\Box A,\Gamma} \quad t\ b = \forall \Gamma', o : \Gamma \leq \Gamma', e : \Gamma' \lhd_{\lambda_{\mathrm{IK}}} \Delta.\ L_{A,\Delta}\ (\mathrm{unbox}_{\lambda_{\mathrm{IK}}}\ (\mathit{wk}\ o\ t)\ e)\ (b\ o\ e)
\end{aligned}
$$

$$
\begin{aligned}
&L_{\Delta,\Gamma} : \Gamma \vdash_{\mathrm{s}} \Delta \longrightarrow [\![\Delta]\!]_\Gamma \longrightarrow \mathit{Type} \\
&L_{\cdot,\Gamma} \qquad \mathrm{empty} \qquad\qquad\qquad\quad () \quad\ = \top \\
&L_{(\Delta,A),\Gamma}\ (\mathrm{ext}\ s\ t) \qquad\qquad\quad\ (\delta, a) = L_{\Delta,\Gamma}\ s\ \delta \times L_{A,\Gamma}\ t\ a \\
&L_{(\Delta,\text{🔒}),\Gamma}\ (\mathrm{ext}_{\text{🔒}}\ s\ (e : \Theta \lhd_{\lambda_{\mathrm{IK}}} \Gamma))\ (\delta, e) = L_{\Delta,\Theta}\ s\ \delta
\end{aligned}
$$

Fig. 9. Logical relations for λ_{IK}

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

Proposition 3 (Fundamental theorem)**.** *Given a term $\Delta \vdash t : A$, a substitution $\Gamma \vdash_s s : \Delta$ and a value $\delta : [\![\Delta]\!]_\Gamma$, if $L_{\Delta,\Gamma}\ s\ \delta$ then $L_{A,\Gamma}\ (\mathit{subst}\ s\ t)\ ([\![t]\!]\ \delta)$.*

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

Proposition 3 entails that $L_{A,\Delta}\ (\mathit{subst}\ \mathrm{id_s}\ t)\ ([\![t]\!]\ \mathit{freshEnv}_\Delta)$ for any term t, if we pick s as the identity substitution $\mathrm{id_s} : \Delta \vdash_{\mathrm{s}} \Delta$, and δ as $\mathit{freshEnv}_\Delta : [\![\Delta]\!]_\Delta$, since they can be shown to be related as $L_{\Delta,\Delta}\ \mathrm{id_s}\ \mathit{freshEnv}_\Delta$. From this it follows that $\mathit{subst}\ \mathrm{id_s}\ t \sim \mathit{reify}_A\ ([\![t]\!]\ \mathit{freshEnv}_\Delta)$, and further that $t \sim \mathit{quote}\ [\![t]\!]$ from the definition of *quote* and the fact that $\mathit{subst}\ \mathrm{id_s}\ t = t$. As a result, the composite $\mathit{norm} = \mathit{quote} \circ [\![-]\!]$ is *adequate*, i.e. $\mathit{norm}\ t = \mathit{norm}\ t'$ implies $t \sim t'$.

The soundness of λ_{IK} with respect to possible-world models (see Theorem 2) directly entails $\mathit{quote}\ [\![t]\!] = \mathit{quote}\ [\![u]\!] : \Gamma \vdash_{\mathrm{NF}} A$ for all terms $t, u : \Gamma \vdash A$ such that $\Gamma \vdash t \sim u : A$, which means that $\mathit{norm} = \mathit{quote} \circ [\![-]\!]$ is *complete*. Note that this terminology might be slightly confusing because it is the *soundness* of $[\![-]\!]$ that implies the *completeness* of *norm*.

Theorem 4. *Let $\mathcal{M}$ denote the possible-world model over the frame given by the relations $\Gamma \leq \Gamma'$ and $\Delta \lhd_{\lambda_{IK}} \Gamma$ and the valuation $V_{\iota,\Gamma} = \Gamma \vdash_{NE} \iota$.*

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

3.2 Extending to the Calculus λ_{IS4}

3.2.1 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.

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

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

If $\Delta \lhd_{\lambda_{\mathrm{IS4}}} \Gamma$ then Γ is an extension of Δ 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 $\lambda_{\mathrm{IK}}/\Box$-Elim which now includes the modal accessibility relation for λ_{IS4}. Similarly, the substitution rule for contexts with locks now refers to $\lhd_{\lambda_{\mathrm{IS4}}}$.

$$\lambda_{\mathrm{IS4}}/\Box\text{-Elim}\ \frac{\Delta \vdash t : \Box A \qquad e : \Delta \lhd_{\lambda_{\mathrm{IS4}}} \Gamma}{\Gamma \vdash \mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t\, e : A} \qquad \frac{\Theta \vdash s : \Delta \qquad e : \Theta \lhd_{\lambda_{\mathrm{IS4}}} \Gamma}{\Gamma \vdash_{\mathrm{s}} \mathsf{ext}_{🔒}\, s\, e : \Delta, 🔒}$$

Fig. 11. Intrinsically-typed terms and substitutions of λ_{IS4} (omitting the unchanged rules of Fig. 5)

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 $\lambda_{\mathrm{IS4}}/\Box$-Elim. Unlike before, Rule $\Box$-β now performs a substitution to modify the term $\Delta, 🔒 \vdash t : A$ 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.

$$\Box\text{-}\beta\ \frac{\Delta, 🔒 \vdash t : A \qquad e : \Delta \lhd_{\lambda_{\mathrm{IS4}}} \Gamma}{\Gamma \vdash \mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, (\mathsf{box}\, t)\, e \sim \mathit{subst}\, (\mathsf{ext}_{🔒}\, \mathsf{id}_{\mathrm{s}}\, e)\, t} \qquad \Box\text{-}\eta\ \frac{\Gamma \vdash t : \Box A}{\Gamma \vdash t \sim \mathsf{box}\, (\mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t\, (\mathsf{lock}\, \mathsf{nil}))}$$

Fig. 12. Equational theory for λ_{IS4} (omitting the unchanged rules of Fig. 7)

3.2.2 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 $\mathit{trim}_{\lambda_{\mathrm{IS4}}} : \forall w.\, [\![\Gamma]\!]_w \to \Delta \lhd_{\lambda_{\mathrm{IS4}}} \Gamma \to [\![\Delta, 🔒]\!]_w$ to manipulate the environment for evaluating $\mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t\, e$, as seen below.

$$\begin{aligned} [\![\mathsf{unbox}_{\lambda_{\mathrm{IS4}}}\, t\, e]\!]\, \gamma &= [\![t]\!]\, \delta\, \mathsf{id}_{\leq}\, m \\ \text{where } (\delta, m) &= \mathit{trim}_{\lambda_{\mathrm{IS4}}}\, \gamma\, e \end{aligned}$$

The additional frame requirements ensures that the function $\mathit{trim}_{\lambda_{\mathrm{IS4}}}$ can be implemented. For example, consider implementing the case of $\mathit{trim}_{\lambda_{\mathrm{IS4}}}$ for some argument of type $[\![\Gamma]\!]_w$ and the extension $\mathsf{nil} : \Gamma \lhd_{\lambda_{\mathrm{IS4}}} \Gamma$ that 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 2, 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. In addition to the coherence conditions stated before Theorem 2 the soundness proof for λ_{IS4} relies on coherence conditions involving the additional proofs refl_m and trans_m that a reflexive and transitive modal accessibility relation R_m must come equipped with. Specifically, trans_m also needs to be associative, refl_m also needs to be a unit for trans_m, and the proofs of the factorization condition also need to satisfy the functoriality laws in the modal accessibility argument, i.e. $\mathit{factor}_i\ (\mathit{refl}_m\ w)\ i\ =\ i$, $\mathit{factor}_m\ (\mathit{refl}_m\ w)\ i\ =\ \mathit{refl}_m\ w'$, $\mathit{factor}_i\ (\mathit{trans}_m\ n\ m)\ i\ =\ \mathit{factor}_i\ n\ i'$ and $\mathit{factor}_m\ (\mathit{trans}_m\ n\ m)\ i = \mathit{trans}_m\ (\mathit{factor}_m\ n\ i')\ (\mathit{factor}_m\ m\ i)$ where $i' := \mathit{factor}_i\ m\ i : w\ R_i\ w'$.

Proposition 5. *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. This is a version of Clouston [2018, Theorem 4.8] for λ_{IS4} where the side condition of Rule $\lambda_{\text{IS4}}/\Box$-Elim appears as an argument to the term former unbox and hence idempotency is not imposed on the comonad $\Box$. ◻

Theorem 6. *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 the functions $[\![t]\!]$ and $[\![u]\!] : \forall w.\ [\![\Gamma]\!]_w \to [\![A]\!]_w$ as determined by $\mathcal{M}$ are equal.*

Proof. The right adjoint determined by a reflexive and transitive frame has a comonad structure so that we can conclude by applying Proposition 5. ◻

3.2.3 NbE Model. The normal forms of λ_{IS4} are defined as before, except for the following rule replacing the neutral rule $\lambda_{\text{IK}}/\text{Ne}/\Box$-Elim.

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

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

$$\begin{aligned} \mathit{reify}_{\Box A,\Gamma}\quad b &= \mathsf{box}\ (\mathit{reify}_{A,(\Gamma,🔒)}\ (b\ \mathsf{id}_{\leq}\ (\mathsf{lock}\ \mathsf{nil}))) \\ \mathit{reflect}_{\Box A,\Gamma}\ n &= \lambda(o : \Gamma \leq \Gamma').\ \lambda(e : \Gamma' \lhd_{\lambda_{\text{IS4}}} \Delta).\ \mathit{reflect}_{A,\Delta}\ (\mathsf{unbox}\ (\mathit{wk}\ o\ n)\ e) \end{aligned}$$

Theorem 7. *Let $\mathcal{M}$ denote the possible-world model over the reflexive and transitive frame given by the relations $\Gamma \leq \Gamma'$ and $\Delta \lhd_{\lambda_{IS4}} \Gamma$ and the valuation $V_{\iota,\Gamma} = \Gamma \vdash_{NE} \iota$.*

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

The proof of this theorem requires us to identify terms by extending the equational theory of λ_{IS4} with an additional rule. To understand the need for it, consider unboxing a term $\Gamma \vdash t : \Box A$ into an extended context Γ, B in λ_{IS4}. We may first weaken t as $\Gamma, B\ \vdash\ \mathit{wk}\ (\mathsf{drop}\ \mathsf{id}_{\leq})\ t\ :\ \Box A$ and then apply unbox as $\Gamma, B\ \vdash\ \mathsf{unbox}\ (\mathit{wk}\ (\mathsf{drop}\ \mathsf{id}_{\leq})\ t)\ \mathsf{nil}\ :\ A$. However, we may also apply unbox on t as $\Gamma, B \vdash \mathsf{unbox}\ t\ (\mathsf{var}\ \mathsf{nil})\ :\ A$. This weakens the term "explicitly" in the sense that the weakening with B is recorded in the term by the proof $\mathsf{var}\ \mathsf{nil}$ of the modal accessibility relation $\Gamma \lhd_{\lambda_{\text{IS4}}} \Gamma, B$. The two ways of unboxing $\Gamma \vdash t : \Box A$ into the extended context Γ, B result in two terms with the same denotation in the possible-world semantics but *distinct* typing derivations.

We wish the two typing derivations unbox t (var nil) and unbox (wk (drop $\mathrm{id}_{\leq}$) t) nil to be identified. For this reason, we extend the equational theory of λ_{IS4} with the rule unbox t ($\mathit{trans}_m\, e\, e'$) $\sim$ unbox (wk ($\mathit{toOPE}\, e$) t) e' for any *lock-free* extension e, which can be converted to a sequence of drops using the function toOPE. Explicit weakening can also be avoided by, instead of extending the equational theory, changing the definition of the modal accessibility relation such that $\Delta \lhd_{\lambda_{\mathrm{IS4}}} \Gamma$ holds only if $\Gamma = \Delta$ or $\Gamma = \Delta, 🔒, \Gamma'$ for some Γ'. Note that the modal accessibility relation for λ_{IK}, where the issue of explicit weakening does not occur, satisfies this property.

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 $\lhd_{\lambda_{\mathrm{IT}}}$ and $\lhd_{\lambda_{\mathrm{IK4}}}$ 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 $\mathit{trim}_{\lambda_{\mathrm{IT}}} : [\![\Gamma]\!]_w \to \Delta \lhd_{\lambda_{\mathrm{IT}}} \Gamma \to [\![\Delta, 🔒]\!]_w$ for λ_{IT}, and similarly $\mathit{trim}_{\lambda_{\mathrm{IK4}}}$ for λ_{IK4}. The modification to the neutral rule λ_{IK}/Ne/$\Box$-Elim 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 3.1.3 and 3.2.3 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 8. *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 3.1.3. 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 $\Gamma \vdash t \sim \mathit{quote}([\![t]\!]_{\mathcal{M}_0}) = \mathit{quote}([\![u]\!]_{\mathcal{M}_0}) \sim u : A$. □

Note that this statement corresponds to Clouston [2018, 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 ⧫ on types.

Theorem 9. *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 8. □

This statement corresponds to Clouston [2018, 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*, to which the model of Subsection 3.2.3 does not belong.

Completeness of the Deductive Theory. Using the quotation function of an NbE model $\mathcal{N}$, i.e. $\mathit{quote} : \mathcal{N}(\llbracket \Gamma \rrbracket, \llbracket A \rrbracket) \rightarrow \Gamma \vdash A$, 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 3.1.3 and 3.2.3 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 10. *Let $\Gamma : Ctx$ be a context and $A : Ty$ a type. If for all possible-world models $\mathcal{M}$ it is the case that $\mathcal{M}(\llbracket \Gamma \rrbracket, \llbracket A \rrbracket)$ 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 3.1.3. Since $\mathcal{M}_0$ is a possible-world model, by assumption we have a morphism $p : \mathcal{M}_0(\llbracket \Gamma \rrbracket, \llbracket A \rrbracket)$. And lastly, since $\mathcal{M}_0$ is an NbE model, we have the term $\mathit{quote}(p) : \Gamma \vdash A$. □

Theorem 11. *Let $\Gamma : Ctx$ 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 $\mathcal{M}(\llbracket \Gamma \rrbracket, \llbracket A \rrbracket)$ is inhabited then there is a term $t : \Gamma \vdash A$ of λ_{IS4}.*

Proof. As for Theorem 10. □

Note that the proofs of Theorems 10 and 11 are constructive.

Decidability of the Equational Theory. 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 $\llbracket t \rrbracket = \llbracket u \rrbracket : \mathcal{N}(\llbracket \Gamma \rrbracket, \llbracket A \rrbracket)$, we obtain decidability of the equational theory from decidability of the equality of normal forms $n, m : \Gamma \vdash_{\mathrm{NF}} A$. Given the NbE models constructed in Subsections 3.1.3 and 3.2.3 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 12. *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 $n, m : \Gamma \vdash_{\mathrm{NF}} A$ 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 $n, m : \Gamma \vdash_{\mathrm{NF}} A$ are immediate.

Let $\mathcal{N}$ be the NbE model we constructed in Subsection 3.1.3. Completeness and adequacy of $\mathcal{N}$ imply that we have $t \sim u$ if and only if $\mathit{norm}\, t = \mathit{norm}\, u$ for the function $\mathit{norm} : \Gamma \vdash A \longrightarrow \Gamma \vdash_{\mathrm{NF}} A, t \mapsto \mathit{quote}\, \llbracket t \rrbracket$. Now, $t \sim u$ is decidable because $\mathit{norm}\, t = \mathit{norm}\, u$ is decidable by the observation we started with. □

Theorem 13. *For any two terms $t, u : \Gamma \vdash A$ of λ_{IS4} the problem whether $t \sim u$ is decidable.*

Proof. As for Theorem 12. □

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[2] 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

[2]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 $\llbracket A \rrbracket_w$ holds in $\mathcal{M}_0$ if and only if $\llbracket A \rrbracket_v$ does in $\mathcal{M}_1$.

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 14 (cf. Clouston [2018, Lemma A.1])**.** *Let Δ, Γ : Ctx be arbitrary contexts, both possibly containing locks, and A : Ty an arbitrary type. There is an operation $\Gamma \vdash A \longrightarrow \Delta, \Gamma \vdash A$ 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 15. *Let Δ, Γ : Ctx be arbitrary contexts, both possibly containing locks, 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 (Theorems 4 and 7) 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 $\Delta' \lhd \Delta \leq \Delta, \Gamma$. But in that case the induction hypothesis says that we have a neutral element $n'' : \cdot \vdash_{\mathrm{NE}} \Box A$, which is impossible. □

Note that some form of normalization seems to be needed in the proof of Lemma 15. More specifically, the "strengthening" of a term of the form $\mathsf{unbox}\, t\, e$ from the context $\cdot, 🔒, \cdot$ to the empty context $\cdot$ cannot possibly result in a term of the form $\mathsf{unbox}\, t'\, e'$ because there is *no* context Γ such that $\Gamma \lhd \cdot$ in λ_{IK}. As an example, consider the term $\mathsf{unbox}\, (\mathsf{box}\, (\lambda\, x.\, x))\, \mathsf{nil}$, which needs to be strengthened to $\lambda\, x.\, x$.

With these two lemmas at hand we are ready to prove denecessitation through normalization:

Theorem 16. *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 $t' : \cdot, 🔒 \vdash A$ using Lemma 14 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 $u' = \mathit{norm}\, u : \cdot \vdash_{\mathrm{NF}} \Box A$ using Theorems 4 and 7. By inspection of normal forms (see Fig. 8) we know that u' must be of the form $\mathsf{box}\, v$ for some normal form $v : \cdot, 🔒 \vdash_{\mathrm{NF}} A$, from which we obtain a term $t : \cdot \vdash A$ using Lemma 15 since the context $\cdot, 🔒$ 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. Note that the consequences we recorded are completely independent of the concrete model construction. To wit, the two completeness theorems follow from the mere existence of an NbE model, and the decidability and denecessitation theorems follow from the mere existence of a normalization function.

5 PROGRAMMING-LANGUAGE APPLICATIONS

In this section, we discuss some implications of normalization for Fitch-style calculi for specific interpretations of the necessity modality in the context of programming languages. In particular, we show how normalization can be used to prove properties about program calculi by leveraging the shape of normal forms of terms. We extend the term calculi presented earlier with application-specific primitives, ensure that the extended calculi are in fact normalizing, and then use this result to prove properties similar to capability safety, noninterference, and binding-time correctness. Note that we do not actually prove the general properties for these calculi, and only illustrate special cases. Although possible, proving the general properties requires further technical development that obscures the main idea underlying the use of normal forms for simplifying these proofs.

5.1 Capability Safety

Choudhury and Krishnaswami [2020] present a modal type system based on IS4 for a programming language with implicit effects in the style of ML [Milner et al. 1990] and the computational lambda calculus [Moggi 1989]. In this language, programs need access to capabilities to perform effects. For instance, a primitive for printing a string requires a capability as an argument in addition to the string to be printed. Crucially, capabilities cannot be introduced within the language, and must be obtained either from the global context (called *ambient* capabilities) or as a function argument.

Let us denote the type of capabilities by Cap. Passing a printing capability c to a function of type $\mathsf{Cap} \Rightarrow \mathsf{Unit}$ in a language that uses capabilities to print yields a program that either (1) does not print, (2) prints using only the capability c, or (3) prints using ambient capabilities (and possibly c). A program that at most uses the capabilities that it is passed explicitly, as in the cases 1 and 2, is said to be *capability safe*. To identify such programs, Choudhury and Krishnaswami [2020] introduce a comonadic modality $\square$ to capture capability safety. Their type system is loosely based on the dual-context calculus for IS4 [Kavvos 2020; Pfenning and Davies 2001]. A term of type $\square A$ is enforced to be capability safe by making the introduction rule for $\square$ "brutally" remove all capabilities from the typing context. As a result, programs with the type $\square(\mathsf{Cap} \Rightarrow \mathsf{Unit})$ are denied ambient capabilities and thus guaranteed to behave like the cases 1 and 2.

Capability safety can be characterized precisely using the *capability space* model of Choudhury and Krishnaswami [2020]. A capability space (X, w_X) is a set X and a weight relation w_X that assigns sets of capabilities to every member in X. In this model, it is possible to define a comonad that restricts the underlying set of a capability space to those elements that are only related to the empty set of capabilities. This comonad has a left adjoint that replaces the weight relation of a capability space by the relation that relates every element to the empty set of capabilities. This adjunction suggests that capability spaces are a model of λ_{IS4} and we may thus use λ_{IS4} to write programs that support reasoning about capability safety.

In this subsection, we present a calculus λ_{IS4}+Moggi$^{\mathsf{Cap}}$ that extends λ_{IS4} with a capability type and a monad for printing effects. We extend the normalization algorithm for λ_{IS4} to λ_{IS4}+Moggi$^{\mathsf{Cap}}$ and show that the resulting normal forms can be used to prove a kind of capability safety. In contrast to the language presented by Choudhury and Krishnaswami [2020], λ_{IS4}+Moggi$^{\mathsf{Cap}}$ models a language where effects are explicit in the type of a term. Languages with explicit effects, such as Haskell [Augustsson et al. 1990] (with the IO monad) or PureScript [Freeman 2013] (with the `Effect` monad), can also benefit from a mechanism for capability safety, and we begin with an example in a hypothetical extension of PureScript to illustrate this.

Example in PureScript. Let us consider web development in PureScript. A web application may consist of a mashup of several components, e.g. social media, news feed, or chat, provided by untrusted sources. A component is a function of type

```
type Component = Element -> Effect Unit
```

that takes as a parameter the DOM element where the component will be rendered. For the correct functioning of the web application, it is important that components do not interfere with each other

$$\textit{Ty} \quad A, B ::= \ldots \mid \mathsf{T}\,A \mid \mathsf{Cap} \mid \mathsf{String} \mid \mathsf{Unit} \qquad\qquad \textit{Ctx} \quad \Gamma ::= \ldots$$

$$\textsc{T-Intro}\ \frac{\Gamma \vdash t : A}{\Gamma \vdash \mathsf{return}\, t : \mathsf{T}\,A} \qquad \textsc{T-Elim}\ \frac{\Gamma \vdash t : \mathsf{T}\,A \qquad \Gamma, A \vdash u : \mathsf{T}\,B}{\Gamma \vdash \mathsf{let}\, t\, u : \mathsf{T}\,B}$$

$$\textsc{Unit-Intro}\ \Gamma \vdash \mathsf{unit} : \mathsf{Unit} \qquad \textsc{String-Lit}\ \frac{}{\Gamma \vdash \mathsf{str}_s : \mathsf{String}}\ s \in \textit{String}$$

$$\textsc{T-Print}\ \frac{\Gamma \vdash c : \mathsf{Cap} \qquad \Gamma \vdash s : \mathsf{String}}{\Gamma \vdash \mathsf{print}\, c\, s : \mathsf{T}\,\mathsf{Unit}}$$

Fig. 13. Types, contexts and terms of λ_{IS4}+Moggi$^{\text{Cap}}$ (omitting the unchanged rules of Figs. 5 and 11)

in malicious ways. For example, a malicious component (of Bob) could illegitimately overwrite a DOM element (of Alice):

```
evilBob :: Component
evilBob e = do w <- window
               doc <- document w
               aliceE <- getElementById "alice.app" doc
               setTextContent "Alice has been hacked!" aliceE
```

The issue here is that Bob has unrestricted access to the function `window :: Effect Window`, and is able to obtain the DOM using `document :: Window -> Effect DOM` and overwrite an element that belongs to Alice. Capabilities can be leveraged to restrict the access to `window`. We can achieve this by extending PureScript with a type `WindowCap`, a type constructor `Box` that works similarly to Choudhury and Krishnaswami's $\Box$, and replacing the function `window` with a function `window' :: WindowCap -> Effect Window` that requires an additional capability argument. By making `WindowCap` an ambient capability that is available globally, all existing programs retain their unrestricted access to retrieve a window as before. The difference now, however, is that we can selectively restrict some programs and limit their access to `WindowCap` using `Box`. We can define a variant of the type `Component` as:

```
type Component' = Box (Element -> Effect Unit)
```

By requiring Bob to write a component of the type `Component'`, we are ensured that Bob cannot overwrite an element that belongs to Alice. This is because the `Box` type constructor used to define `Component'` disallows access to all ambient capabilities (including `WindowCap`), and thus restricts Bob to only using the given `Element` argument. In particular, the program `evilBob` cannot be reproduced with the type `Component'` since the substitute function `window'` requires a capability that is neither available as an argument nor as an ambient capability.

Extension with a Capability and a Monad. We extend λ_{IS4} with a monad for printing based on Moggi's monadic metalanguage [Moggi 1991]. We introduce a type $\mathsf{T}\,A$ that denotes a monadic computation that can print before returning a value of type A, a type Cap for capabilities, and a type String for strings. Fig. 13 summarizes the terms that correspond to this extension. The term construct print is used for printing. The equational theory of λ_{IS4}+Moggi$^{\text{Cap}}$ and the corresponding normal forms are summarized in Fig. 14 and Fig. 15, respectively.

$$\text{T-}\beta\quad \frac{\Gamma \vdash t : A \qquad \Gamma, A \vdash u : \mathrm{T}\,B}{\Gamma \vdash \mathsf{let}\,(\mathsf{return}\, t)\, u \sim \mathit{subst}\,(\mathsf{ext}\,\mathsf{id_s}\, t)\, u} \qquad \text{T-}\eta\quad \frac{\Gamma \vdash t : \mathrm{T}\,A}{\Gamma \vdash t \sim \mathsf{let}\, t\, (\mathsf{return}\,(\mathsf{var\,zero}))}$$

$$\text{T-}\gamma\quad \frac{\Gamma \vdash t_1 : A \qquad \Gamma, A \vdash t_2 : \mathrm{T}\,B \qquad \Gamma, B \vdash t_3 : \mathrm{T}\,C}{\Gamma \vdash \mathsf{let}\,(\mathsf{let}\, t_1\, t_2)\, t_3 \sim \mathsf{let}\, t_1\,(\mathsf{let}\, t_2\,(\mathit{wk}\,(\mathsf{keep}\,(\mathsf{drop}\,\mathsf{id}_{\leq}))\, t_3))}$$

Fig. 14. Equational theory for λ_{IS4}+Moggi$^{\mathrm{Cap}}$ (omitting the unchanged rules of Figs. 7 and 12)

$$\textsc{Nf/T-Intro}\quad \frac{\Gamma \vdash_{\mathrm{NF}} m : A}{\Gamma \vdash_{\mathrm{NF}} \mathsf{return}\, m : \mathrm{T}\,A} \qquad \textsc{Nf/T-Elim}\quad \frac{\Gamma \vdash_{\mathrm{NE}} n : \mathrm{T}\,A \qquad \Gamma, A \vdash_{\mathrm{NF}} m : \mathrm{T}\,B}{\Gamma \vdash_{\mathrm{NF}} \mathsf{let}\, n\, m : \mathrm{T}\,B}$$

$$\textsc{Nf/Unit-Intro}\quad \Gamma \vdash_{\mathrm{NF}} \mathsf{unit} : \mathrm{Unit} \qquad \textsc{Nf/Up-Cap}\quad \frac{\Gamma \vdash_{\mathrm{NE}} n : \mathrm{Cap}}{\Gamma \vdash_{\mathrm{NF}} \mathsf{up}\, n : \mathrm{Cap}} \qquad \textsc{Nf/Up-String}\quad \frac{\Gamma \vdash_{\mathrm{NE}} n : \mathrm{String}}{\Gamma \vdash_{\mathrm{NF}} \mathsf{up}\, n : \mathrm{String}} \qquad \textsc{Nf/String-Lit}\quad \frac{}{\Gamma \vdash_{\mathrm{NF}} \mathsf{str}_s : \mathrm{String}}\; s \in \mathit{String}$$

$$\textsc{Nf/T-Print}\quad \frac{\Gamma \vdash_{\mathrm{NF}} c : \mathrm{Cap} \qquad \Gamma \vdash_{\mathrm{NF}} s : \mathrm{String} \qquad \Gamma, \mathrm{Unit} \vdash_{\mathrm{NF}} m : \mathrm{T}\,A}{\Gamma \vdash_{\mathrm{NF}} \mathsf{let}\,(\mathsf{print}\, c\, s)\, m : \mathrm{T}\,A}$$

Fig. 15. Normal forms of λ_{IS4}+Moggi$^{\mathrm{Cap}}$ (omitting the unchanged normal forms of λ_{IS4})

To extend the NbE model of λ_{IS4} with an interpretation for the monad, we use the standard techniques used for normalizing computational effects [Ahman and Staton 2013; Filinski 2001]. The interpretation of the other primitive types also follows a standard pursuit [Valliappan et al. 2021]: we interpret Cap by neutrals of type Cap and String by the disjoint union of *String* and neutrals of type String. The difference in their interpretation is caused by the fact that there is no introduction form for the type Cap.

Proving Capability Safety. Programs that lack access to capabilities are necessarily capability safe. We say that a program $\Gamma \vdash p : A$ is *trivially capability safe* if there is a program $\cdot \vdash p' : A$ such that $\Gamma \vdash p \sim \mathit{leftConcat}_\Gamma\, p' : A$, where the operation $\mathit{leftConcat}_\Gamma : \Delta \vdash A \to \Gamma, \Delta \vdash A$ on terms is the one given by Lemma 14 for an arbitrary context Δ.

First, we prove an auxiliary lemma about normal forms with a capability in context.

Lemma 17. *For any context Γ, type A and normal form $c : \mathrm{Cap}, \text{🔒}, \Gamma \vdash_{NF} n : A$ there is a normal form $\text{🔒}, \Gamma \vdash_{NF} n' : A$ such that $n = \mathit{leftConcat}_{\cdot,\mathrm{Cap}}\, n'$.*

Proof. By mutual induction on the normal and neutral forms. The nonimmediate case is when the neutral is of the form $c : \mathrm{Cap}, \text{🔒}, \Gamma \vdash_{\mathrm{NE}} \mathsf{unbox}\, e\, n : A$ for some $e : \Delta \lhd_{\lambda_{\mathrm{IS4}}} c : \mathrm{Cap}, \text{🔒}, \Gamma$ and $n : \Delta \vdash_{\mathrm{NE}} \Box A$. There are two cases to consider depending on whether Δ contains the leftmost lock in $c : \mathrm{Cap}, \text{🔒}, \Gamma$. If this is the case, we apply induction; otherwise, we appeal to the subformula property [Clouston 2018, Theorem 2.7] and conclude that this case is impossible: there are no neutral terms of the form $c : \mathrm{Cap} \vdash_{\mathrm{NE}} \Box A$. □

Now, we observe that all terms $c : \mathrm{Cap} \vdash t : \Box A$ are trivially capability safe. By normalization, we have that $c : \mathrm{Cap} \vdash t \sim \mathit{norm}\, t : \Box A$. Given the definition of normal forms of λ_{IS4}+Moggi$^{\mathrm{Cap}}$, $\mathit{norm}\, t$ must be $\mathsf{box}\, n$ for some normal form $c : \mathrm{Cap}, 🔒 \vdash_{\mathrm{NF}} n : A$. By Lemma 17, there is a normal form $🔒 \vdash_{\mathrm{NF}} n' : A$ such that $n = \mathit{leftConcat}_{\cdot,\mathrm{Cap}}\, n'$. Since the operation $\mathit{leftConcat}$ commutes with box, i.e. $\mathit{leftConcat}_{\cdot,\mathrm{Cap}}\, (\mathsf{box}\, n') = \mathsf{box}\, (\mathit{leftConcat}_{\cdot,\mathrm{Cap}}\, n')$, we also have that $t \sim \mathsf{box}\, n = \mathit{leftConcat}_{\cdot,\mathrm{Cap}}\, (\mathsf{box}\, n')$. As a result, t must be trivially capability safe.

A consequence of this observation is that any term $c : \mathrm{Cap} \vdash t : \Box(\mathrm{T}\,\mathrm{Unit})$ is trivially capability safe. This means that t does not print since it could not possibly do so without a capability. Going further, we can also observe that $t \sim \mathsf{return}\, \mathsf{unit} : \mathrm{T}\,\mathrm{Unit}$, since the only normal form of type $\mathrm{T}\,\mathrm{Unit}$ in the empty context is $\cdot \vdash_{\mathrm{NF}} \mathsf{return}\, \mathsf{unit} : \mathrm{T}\,\mathrm{Unit}$. Note that this argument (and the one above) readily adapts to a vector of capabilities $\vec{c}$ in context as opposed to a single capability c.

5.2 Information-Flow Control

Information-flow control (IFC) [Sabelfeld and Myers 2003] is a technique used to protect the confidentiality of data in a program by tracking the flow of information within the program.

In type-based *static* IFC [e.g. Abadi et al. 1999; Russo et al. 2008; Shikuma and Igarashi 2008] types are used to associate values with confidentiality levels such as *secret* or *public*. The type system ensures that secret inputs do not interfere with public outputs, enforcing a security policy that is typically formalized as a kind of *noninterference* property [Goguen and Meseguer 1982].

Noninterference is proved by reasoning about the semantic behaviour of a program. Tomé Cortiñas and Valliappan [2019] present a proof technique that uses normalization for showing noninterference for a static IFC calculus based on Moggi's monadic metalanguage [Moggi 1991]. This technique exploits the insight that normal forms represent equivalence classes of terms identified by their semantics, and thus reasoning about normal forms of terms (as opposed to terms themselves) vastly reduces the set of programs that we must take into consideration. Having developed normalization for Fitch-style calculi, we can leverage this technique to prove noninterference. In this subsection, we illustrate the technique for an extension of λ_{IK} with Booleans, by interpreting the type $\Box A$ as a secret of type A.

Extension with Booleans. Noninterference can be better appreciated in the presence of a type whose values are distinguishable by an external observer. To this extent, we extend the base calculus λ_{IK} with a type Bool and corresponding introduction and elimination forms as described in Fig. 16.

$$\mathit{Ty} \quad A, B ::= \ldots \mid \mathrm{Bool} \qquad\qquad \mathit{Ctx} \quad \Gamma ::= \ldots$$

$$\text{Bool-Intro-true}\ \ \frac{}{\Gamma \vdash \mathsf{true} : \mathrm{Bool}} \qquad \text{Bool-Intro-false}\ \ \frac{}{\Gamma \vdash \mathsf{false} : \mathrm{Bool}} \qquad \text{Bool-Elim}\ \ \frac{\Gamma \vdash b : \mathrm{Bool} \qquad \Gamma, \Gamma' \vdash t_1 : A \qquad \Gamma, \Gamma' \vdash t_2 : A}{\Gamma, \Gamma' \vdash \mathsf{ifte}\, b\, t_1\, t_2 : A}$$

Fig. 16. Types, contexts and intrinsically-typed terms of λ_{IK}+Bool (omitting the unchanged rules of Fig. 5)

We modify the usual elimination rule for Bool by allowing the context of the conclusion $\mathsf{ifte}\, b\, t_1\, t_2$ and branches t_1 and t_2 in the rule Bool-Elim to extend the context of the scrutinee b. This enables the following *commuting conversion*, which is required to ensure that terms can be fully normalized and normal forms enjoy the subformula property [Clouston 2018, Fig. 2]:

$$\frac{\Delta \vdash b : \mathrm{Bool} \qquad \Delta, \Delta' \vdash t_1 : \Box A \qquad \Delta, \Delta' \vdash t_2 : \Box A \qquad e : \Delta, \Delta' \lhd \Gamma}{\Gamma \vdash \mathsf{unbox}\, (\mathsf{ifte}\, b\, t_1\, t_2)\, e \sim \mathsf{ifte}\, b\, (\mathsf{unbox}\, t_1\, e)\, (\mathsf{unbox}\, t_2\, e)}$$

A commuting conversion is required as usual for every other elimination rule, including the rule $\Rightarrow$-Elim. These are however standard and thus omitted here.

The normal forms of λ_{IK}+Bool include those of λ_{IK} in addition to the following.

$$\frac{}{\Gamma \vdash_{\mathrm{NF}} \mathsf{true} : \mathsf{Bool}}\ \textsc{Nf/Bool-Intro-true} \qquad \frac{}{\Gamma \vdash_{\mathrm{NF}} \mathsf{false} : \mathsf{Bool}}\ \textsc{Nf/Bool-Intro-false} \qquad \frac{\Gamma \vdash_{\mathrm{NE}} n : \mathsf{Bool} \quad \Gamma, \Gamma' \vdash_{\mathrm{NF}} m_1 : A \quad \Gamma, \Gamma' \vdash_{\mathrm{NF}} m_2 : A}{\Gamma, \Gamma' \vdash_{\mathrm{NF}} \mathsf{ifte}\, n\, m_1\, m_2 : A}\ \textsc{Nf/Bool-Elim}$$

To extend the NbE model of λ_{IK} to Booleans, we leverage the interpretation of sum types used by Abel and Sattler [2019], who attribute their idea to Altenkirch and Uustalu [2004]. This interpretation readily supports commuting conversions, and a minor refinement that reflects the change to the rule Bool-Elim yields a reifiable interpretation for Booleans in λ_{IK}+Bool.

Proving Noninterference. A program $\cdot \vdash f : \Box A \Rightarrow \mathsf{Bool}$ is *noninterferent* if it is the case that $\cdot \vdash \mathsf{app}\, f\, s_1 \sim \mathsf{app}\, f\, s_2 : \mathsf{Bool}$ for any two secrets $\cdot \vdash s_1, s_2 : \Box A$. By instantiating A to Bool, we can show that any program $\cdot \vdash f : \Box \mathsf{Bool} \Rightarrow \mathsf{Bool}$ is noninterferent and thus cannot leak a secret Boolean argument. In λ_{IK}+Bool, the type system ensures that data of type $\Box A$ type can only influence (or *flow* to) data of type $\Box B$, thus all programs of type $\Box \mathsf{Bool} \Rightarrow \mathsf{Bool}$ must be noninterferent. To show this, we analyze the possible normal forms of f and observe that they must be equivalent to a constant function, such as $\lambda x.\, \mathsf{true}$ or $\lambda x.\, \mathsf{false}$, which evidently does not use its input argument x and is thus noninterferent.

In detail, normal forms of type $\Box \mathsf{Bool} \Rightarrow \mathsf{Bool}$ must have the shape $\lambda x.\, m$, for some normal form $\cdot, \Box \mathsf{Bool} \vdash_{\mathrm{NF}} m : \mathsf{Bool}$. If m is either true or false, then $\lambda x.\, m$ must be a constant function and we are done. Otherwise, it must be some normal form $\cdot, \Box \mathsf{Bool} \vdash_{\mathrm{NF}} \mathsf{ifte}\, n\, m_1\, m_2 : \mathsf{Bool}$ with a neutral $n : \mathsf{Bool}$ either in context $\cdot$ or in context $\cdot, \Box \mathsf{Bool}$. Such a neutral could either be of shape $\mathsf{unbox}\, n'$ or $\mathsf{app}\, n''\, m'$ for some neutrals n' and n''. However, this is impossible, since the context of the neutral $\mathsf{unbox}\, n'$ must contain a lock, and neither the context $\cdot$ nor the context $\cdot, \Box \mathsf{Bool}$ do. The existence of n'' can be dismissed using a special subformula property exhibited by neutrals [Tomé Cortiñas and Valliappan 2019, Property 3.1].

Discussion. Observe that not all Fitch-style calculi are well-suited for interpreting the type $\Box A$ as a secret, because noninterference might not hold. In λ_{IS4}, the term $\lambda x.\, \mathsf{unbox}\, x : \Box A \Rightarrow A$ (axiom T) is well-typed but leaks the secret x, thus breaking noninterference. The validity of the interpretation and thus noninterference depends on the calculus under consideration and the axioms it exhibits.

5.3 Partial Evaluation

Davies and Pfenning [1996, 2001] present a modal type system for staged computation based on IS4. In their system, the type $\Box A$ represents *code* of type A that is to be executed at a later stage, and the axioms of IS4 correspond to operations that manipulate code. The axiom $\mathrm{K} : \Box(A \Rightarrow B) \Rightarrow (\Box A \Rightarrow \Box B)$ corresponds to substituting code in code, $\mathrm{T} : \Box A \Rightarrow A$ to evaluating code, and $4 : \Box A \Rightarrow \Box\Box A$ to further delaying the execution of code to a subsequent stage. A desired property of this type system is that code must only depend on code, and thus the term $\lambda x : A.\, \mathsf{box}\, x$ must be ill-typed.

Although λ_{IS4} exhibits the desired properties of a type system for staging, its equational theory in Fig. 12 (and the semantics of Fitch-style calculi in general) does not reflect the semantics of staged computation. For example, the result of normalizing the term $\mathsf{box}\,(2 * \mathsf{unbox}\,(\mathsf{box}\, 3))$ in a Fitch-style calculus (extended with natural numbers and multiplication) is $\mathsf{box}\, 6$, while the result expected from reducing it in accordance with Davies and Pfenning's reduction semantics is $\mathsf{box}\,(2 * 3)$.

If we restrict our attention to a special case of staged computation in partial evaluation [Jones et al. 1993], however, the semantics of Fitch-style calculi are better suited. In the context of partial evaluation, the type $\Box A$ represents a *dynamic* computation of type A that must be executed at runtime, and other types represent *static* computations. The goal of a partial evaluator is to optimize runtime execution of a program by eagerly evaluating as many static computations as possible. In this interpretation the normalized term $\mathsf{box}\, 6$ from before is arguably better suited than the term $\mathsf{box}\, (2 * 3)$ since it is evidently more optimal.

In partial evaluation, as with staging in general, we desire that the term $\lambda x : A.\, \mathsf{box}\, x$ be disallowed, since a runtime execution of a dynamic computation must not have a static dependency. This requirement is captured by a property that we refer to as *binding-time correctness*[3]. In this subsection, we extend λ_{IK} with natural numbers and multiplication, and illustrate how normalization can be used to prove binding-time correctness for the extended system.

Extension with Natural Numbers and Multiplication. We extend λ_{IK} with a type Nat, a construct lift for natural number literals, and an operation $*$ for multiplying terms of type Nat—as described in Fig. 17.

$$\mathit{Ty} \quad A, B ::= \ldots \mid \mathrm{Nat} \qquad\qquad \mathit{Ctx} \quad \Gamma ::= \ldots$$

$$\textsc{Nat-Lit}\ \frac{}{\Gamma \vdash \mathsf{lift}\, k : \mathrm{Nat}}\ k \in \mathbb{N} \qquad\qquad \textsc{Nat-Mul}\ \frac{\Gamma \vdash t_1 : \mathrm{Nat} \qquad \Gamma \vdash t_2 : \mathrm{Nat}}{\Gamma \vdash t_1 * t_2 : \mathrm{Nat}}$$

Fig. 17. Types, contexts, intrinsically-typed terms of λ_{IK}+Nat (omitting the unchanged rules of Fig. 5)

We extend the equational theory of λ_{IK} with some expected rules such as $\mathsf{lift}\, k_1 * \mathsf{lift}\, k_2 \sim \mathsf{lift}\, (k_1 * k_2)$ (for natural numbers k_1 and k_2), $\mathsf{lift}\, 0 * t \sim \mathsf{lift}\, 0$, $\mathsf{lift}\, 1 * t \sim t$, $\mathsf{lift}\, k * t \sim t * \mathsf{lift}\, k$, etc. The normal forms of λ_{IK}+Nat include those of λ_{IK} in addition to the following.

$$\textsc{Nf/Nat}_1\ \ \Gamma \vdash_{\mathrm{NF}} \mathsf{lift}\, 0 : \mathrm{Nat} \qquad\qquad \textsc{Nf/Nat}_2\ \frac{\Gamma \vdash_{\mathrm{NE}} n_1 : \mathrm{Nat} \qquad \ldots \qquad \Gamma \vdash_{\mathrm{NE}} n_j : \mathrm{Nat}}{\Gamma \vdash_{\mathrm{NF}} \mathsf{lift}\, k * n_1 * \cdots * n_j : \mathrm{Nat}}\ k \in \mathbb{N} \setminus \{0\}$$

The normal form $\mathsf{lift}\, k * n_1 * \cdots * n_j$ denotes a multiplication of a nonzero literal with a sequence of neutrals of type Nat, which can possibly be empty. The term $\mathsf{box}\, (2 * \mathsf{unbox}\, (\mathsf{box}\, 3))$ from earlier can be represented in λ_{IK}+Nat as $\mathsf{box}\, (\mathsf{lift}\, 2 * \mathsf{unbox}\, (\mathsf{box}\, (\mathsf{lift}\, 3)))$, and its normal form as $\mathsf{box}\, (\mathsf{lift}\, 6)$. To extend the NbE model for λ_{IK} to natural numbers and multiplication, we use the interpretation put forth by Valliappan et al. [2021] for normalizing arithmetic expressions. Omitting the rule $\mathsf{lift}\, 0 * t \sim \mathsf{lift}\, 0$, this interpretation also resembles the one constructed systematically in the framework of Yallop et al. [2018] for commutative monoids.

Proving Binding-Time Correctness. Binding-time correctness for a term $\cdot \vdash f : \mathrm{Nat} \Rightarrow \Box\mathrm{Nat}$ can be stated similar to noninterference: it must be the case that $\cdot \vdash \mathsf{app}\, f\, u_1 \sim \mathsf{app}\, f\, u_2 : \Box\mathrm{Nat}$ for any two arguments $\cdot \vdash u_1, u_2 : \mathrm{Nat}$. The satisfaction of this property implies that a well-typed term $\lambda x : \mathrm{Nat}.\, \mathsf{box}\, x$ cannot exist, since applying it to different arguments yields different results. As before with noninterference, we can prove this property by case analysis on the possible normal forms of f. A normal form of f must be either $\lambda x.\, \mathsf{box}\, (\mathsf{lift}\, 0)$ or $\lambda x.\, \mathsf{box}\, (\mathsf{lift}\, k * n_1 * \cdots * n_j)$ for some natural number k and neutrals $n_1, \ldots, n_j$, where $\cdot, \mathrm{Nat}, \text{🔒} \vdash_{\mathrm{NE}} n_i : \mathrm{Nat}$. In the former case, we are done immediately since $\lambda x.\, \mathsf{box}\, (\mathsf{lift}\, 0)$ is a constant function that evidently satisfies the desired criteria. In the latter case, by analysis on derivation of neutrals, we observe that no such neutrals n_i can exist, and it must also be a constant function $\lambda x.\, \mathsf{box}\, (\mathsf{lift}\, k)$.

[3] named after the fact that static and dynamic are also known as *binding-time annotations*

As a part of binding-time correctness, we may also desire that the term $\lambda\, x : \Box A.\, \mathrm{unbox}\, x$ be disallowed since a static computation must not have a dynamic dependency. This can also be shown by following an argument similar to the proof of noninterference in Subsection 5.2.

Discussion. The reduction semantics for staged computation is given by Davies and Pfenning via translation to a dual-context calculus for IS4, where evaluation under the introduction rule for $\Box$ is disallowed. While it is possible to implement a normalization function for λ_{IS4} that does not normalize under box, this still misses certain reductions that are enabled by the translation. The term $\mathrm{box}\,(1 + \mathrm{unbox}\,(\mathrm{box}\, 2))$ is already in normal form if we simply disallow normalization under box. The translation ensures the reduction of $\mathrm{unbox}\,(\mathrm{box}\, 2)$, and reduces the term to $\mathrm{box}\,(1 + 2)$. This mismatch, in addition to the lack of a model for their system, makes the applicability of Fitch-style calculi for staged computation unclear.

6 RELATED AND FURTHER WORK

Fitch-style Calculi. Fitch-style modal type systems [Borghuis 1994; Martini and Masini 1996] 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 [1994], and later investigated further by Clouston [2018]. 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 $\lambda_{\mathrm{IS4}}/\Box$-$\textsc{Elim}$. One possibility is to define unbox by explicitly recording the context extension as a part of the term former. Davies and Pfenning [1996] present such a system where they annotate the term former unbox as unbox_n to denote the number of 🔒s[4] 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 [2018], 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, et al. [2019] present yet another possibility that reformulates the system for IS4 in Clouston [2018]. They further extend it with dependent types, and also prove a normalization

[4] 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 🔒.

result using NbE with respect to an equational theory that includes both Rule $\Box$-β and Rule $\Box$-η. Although their approach is semantic in the sense of using NbE, their semantic domain has a very syntactic flavour [Gratzer, Sterling, et al. 2019, 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 for Fitch-style Calculi. 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 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 [1984].

Dual-Context Calculi. Dual-context calculi [Davies and Pfenning 2001; Kavvos 2020; Pfenning and Davies 2001] 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 [Abel and Sattler 2019; Altenkirch, Dybjer, et al. 2001; Lindley 2007], 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, Kavvos, et al. [2020] 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 [2021] 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 $\mathrm{R} : A \Rightarrow \Box A$ and $\mathrm{GL} : \Box(\Box A \Rightarrow A) \Rightarrow \Box A$, especially considering that the calculi may differ in more than just the elimination rule for the $\Box$ type.

REFERENCES

Martín Abadi, Anindya Banerjee, Nevin Heintze, and Jon G. Riecke. 1999. "A Core Calculus of Dependency". In: *POPL '99, Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, TX, USA, January 20-22, 1999.* Ed. by Andrew W. Appel and Alex Aiken. ACM, 147–160. DOI: 10.1145/292540.292555.

Andreas Abel, Guillaume Allais, et al., *Agda 2* version 2.6.2.1, 2005–2021. Chalmers University of Technology and Gothenburg University. URL: https://wiki.portal.chalmers.se/agda/pmwiki.php, VCS: https://github.com/agda/agda.

Andreas Abel and Christian Sattler. 2019. "Normalization by Evaluation for Call-By-Push-Value and Polarized Lambda Calculus". In: *Proceedings of the 21st International Symposium on Principles and Practice of Programming Languages, PPDP 2019, Porto, Portugal, October 7-9, 2019.* Ed. by Ekaterina Komendantskaya. ACM, 3:1–3:12. DOI: 10.1145/3354166.3354168.

Danel Ahman and Sam Staton. 2013. "Normalization by Evaluation and Algebraic Effects". In: *Proceedings of the Twenty-ninth Conference on the Mathematical Foundations of Programming Semantics, MFPS 2013, New Orleans, LA, USA, June 23-25, 2013* (Electronic Notes in Theoretical Computer Science). Ed. by Dexter Kozen and Michael W. Mislove. Vol. 298. Elsevier, 51–69. DOI: 10.1016/j.entcs.2013.09.007.

Thorsten Altenkirch, Peter Dybjer, Martin Hofmann, and Philip J. Scott. 2001. "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.* IEEE Computer Society, 303–310. DOI: 10.1109/LICS.2001.932506.

Thorsten Altenkirch and Tarmo Uustalu. 2004. "Normalization by Evaluation for lambda^{-2}". In: *Functional and Logic Programming, 7th International Symposium, FLOPS 2004, Nara, Japan, April 7-9, 2004, Proceedings* (Lecture Notes in Computer Science). Ed. by Yukiyoshi Kameyama and Peter J. Stuckey. Vol. 2998. Springer, 260–275. DOI: 10.1007/978-3-540-24754-8_19.

Lennart Augustsson et al., *Haskell* 1990. URL: https://www.haskell.org/.

Ulrich Berger and Helmut Schwichtenberg. 1991. "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.* IEEE Computer Society, 203–211. DOI: 10.1109/LICS.1991.151645.

Valentijn Anton Johan Borghuis. 1994. *Coming to terms with modal logic.* On the interpretation of modalities in typed λ-calculus, Dissertation, Technische Universiteit Eindhoven, Eindhoven, 1994. Technische Universiteit Eindhoven, Eindhoven, x+219.

Milan Božić and Kosta Došen. 1984. "Models for normal intuitionistic modal logics". *Studia Logica*, 43, 3, 217–245. DOI: 10.1007/BF02429840.

Vikraman Choudhury and Neel Krishnaswami. 2020. "Recovering purity with comonads and capabilities". *Proc. ACM Program. Lang.*, 4, ICFP, 111:1–111:28. DOI: 10.1145/3408993.

Ranald Clouston. 2018. "Fitch-Style Modal Lambda Calculi". In: *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* (Lecture Notes in Computer Science). Ed. by Christel Baier and Ugo Dal Lago. Vol. 10803. Springer, 258–275. DOI: 10.1007/978-3-319-89366-2_14.

Catarina Coquand. 2002. "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. DOI: 10.1023/A:1019964114625.

Rowan Davies and Frank Pfenning. 1996. "A Modal Analysis of Staged Computation". In: *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.* Ed. by Hans-Juergen Boehm and Guy L. Steele Jr. ACM Press, 258–270. DOI: 10.1145/237721.237788.

Rowan Davies and Frank Pfenning. 2001. "A modal analysis of staged computation". *J. ACM*, 48, 3, 555–604. DOI: 10.1145/382780.382785.

W. B. Ewald. 1986. "Intuitionistic Tense and Modal Logic". *J. Symb. Log.*, 51, 1, 166–179. DOI: 10.2307/2273953.

Andrzej Filinski. 2001. "Normalization by Evaluation for the Computational Lambda-Calculus". In: *Typed Lambda Calculi and Applications, 5th International Conference, TLCA 2001, Krakow, Poland, May 2-5, 2001, Proceedings* (Lecture Notes in Computer Science). Ed. by Samson Abramsky. Vol. 2044. Springer, 151–165. DOI: 10.1007/3-540-45413-6_15.

Gisèle Fischer-Servi. 1981. "Semantics for a class of intuitionistic modal calculi". In: *Italian studies in the philosophy of science.* Boston Stud. Philos. Sci. Vol. 47. Reidel, Dordrecht-Boston, Mass., 59–72.

Phil Freeman, *PureScript* 2013. URL: https://www.purescript.org/.

Joseph A. Goguen and José Meseguer. 1982. "Security Policies and Security Models". In: *1982 IEEE Symposium on Security and Privacy, Oakland, CA, USA, April 26-28, 1982.* IEEE Computer Society, 11–20. DOI: 10.1109/SP.1982.10014.

Daniel Gratzer. 2021. "Normalization for multimodal type theory". *CoRR*, abs/2106.01414. https://arxiv.org/abs/2106.01414 arXiv: 2106.01414.

Daniel Gratzer, G. A. Kavvos, Andreas Nuyts, and Lars Birkedal. 2020. "Multimodal Dependent Type Theory". In: *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020.* Ed. by Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller. ACM, 492–506. DOI: 10.1145/3373718.3394736.

Daniel Gratzer, Jonathan Sterling, and Lars Birkedal. 2019. “Implementing a modal dependent type theory”. *Proc. ACM Program. Lang.*, 3, ICFP, 107:1–107:29. doi: 10.1145/3341711.

C. Barry Jay and Neil Ghani. 1995. “The Virtues of Eta-Expansion”. *J. Funct. Program.*, 5, 2, 135–154. doi: 10.1017/S0956796800001301.

Neil D. Jones, Carsten K. Gomard, and Peter Sestoft. 1993. *Partial evaluation and automatic program generation.* Prentice Hall international series in computer science. Prentice Hall. isbn: 978-0-13-020249-9.

G. A. Kavvos. 2020. “Dual-Context Calculi for Modal Logic”. *Log. Methods Comput. Sci.*, 16, 3. https://lmcs.episciences.org/6722.

Sam Lindley. 2007. “Extensional Rewriting with Sums”. In: *Typed Lambda Calculi and Applications, 8th International Conference, TLCA 2007, Paris, France, June 26-28, 2007, Proceedings* (Lecture Notes in Computer Science). Ed. by Simona Ronchi Della Rocca. Vol. 4583. Springer, 255–271. doi: 10.1007/978-3-540-73228-0_19.

Simone Martini and Andrea Masini. 1996. “A computational interpretation of modal proofs”. In: *Proof theory of modal logic (Hamburg, 1993).* Appl. Log. Ser. Vol. 2. Kluwer Acad. Publ., Dordrecht, 213–241. doi: 10.1007/978-94-017-2798-3_12.

Robin Milner, Mads Tofte, and Robert Harper. 1990. *Definition of standard ML.* MIT Press. isbn: 978-0-262-63132-7.

Kenji Miyamoto and Atsushi Igarashi. 2004. “A modal foundation for secure information flow”. In: *In Proceedings of IEEE Foundations of Computer Security (FCS)*, 187–203.

Eugenio Moggi. 1989. “Computational Lambda-Calculus and Monads”. In: *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS ’89), Pacific Grove, California, USA, June 5-8, 1989.* IEEE Computer Society, 14–23. doi: 10.1109/LICS.1989.39155.

Eugenio Moggi. 1991. “Notions of Computation and Monads”. *Inf. Comput.*, 93, 1, 55–92. doi: 10.1016/0890-5401(91)90052-4.

Frank Pfenning and Rowan Davies. 2001. “A judgmental reconstruction of modal logic”. *Math. Struct. Comput. Sci.*, 11, 4, 511–540. doi: 10.1017/S0960129501003322.

Gordon D. Plotkin and Colin Stirling. 1986. “A Framework for Intuitionistic Modal Logics”. In: *Proceedings of the 1st Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, CA, USA, March 1986.* Ed. by Joseph Y. Halpern. Morgan Kaufmann, 399–406.

Alejandro Russo, Koen Claessen, and John Hughes. 2008. “A library for light-weight information-flow security in haskell”. In: *Proceedings of the 1st ACM SIGPLAN Symposium on Haskell, Haskell 2008, Victoria, BC, Canada, 25 September 2008.* Ed. by Andy Gill. ACM, 13–24. doi: 10.1145/1411286.1411289.

Andrei Sabelfeld and Andrew C. Myers. 2003. “Language-based information-flow security”. *IEEE J. Sel. Areas Commun.*, 21, 1, 5–19. doi: 10.1109/JSAC.2002.806121.

Naokata Shikuma and Atsushi Igarashi. 2008. “Proving Noninterference by a Fully Complete Translation to the Simply Typed Lambda-Calculus”. *Log. Methods Comput. Sci.*, 4, 3. doi: 10.2168/LMCS-4(3:10)2008.

Alex K. Simpson. 1994. “The proof theory and semantics of intuitionistic modal logic”. PhD thesis. University of Edinburgh, UK. http://hdl.handle.net/1842/407.

Carlos Tomé Cortiñas and Nachiappan Valliappan. 2019. “Simple Noninterference by Normalization”. In: *Proceedings of the 14th ACM SIGSAC Workshop on Programming Languages and Analysis for Security, CCS 2019, London, United Kingdom, November 11-15, 2019.* Ed. by Piotr Mardziel and Niki Vazou. ACM, 61–72. doi: 10.1145/3338504.3357342.

Nachiappan Valliappan, Alejandro Russo, and Sam Lindley. 2021. “Practical normalization by evaluation for EDSLs”. In: *Haskell 2021: Proceedings of the 14th ACM SIGPLAN International Symposium on Haskell, Virtual Event, Korea, August 26-27, 2021.* Ed. by Jurriaan Hage. ACM, 56–70. doi: 10.1145/3471874.3472983.

Jeremy Yallop, Tamara von Glehn, and Ohad Kammar. 2018. “Partially-static data as free extension of algebras”. *Proc. ACM Program. Lang.*, 2, ICFP, 100:1–100:30. doi: 10.1145/3236795.