Submitted to MFPS 22

A Categorical Normalization Proof for the Modal Lambda-Calculus

Jason Z. S. Hu[1] and Brigitte Pientka[2]

School of Computer Science
McGill University
Montréal, QC, Canada

Abstract

We investigate a simply typed modal λ-calculus, $\lambda^{\to\Box}$, due to Pfenning, Wong and Davies, where we define a well-typed term with respect to a context stack that captures the possible world semantics in a syntactic way. It provides logical foundation for multi-staged meta-programming. Our main contribution in this paper is a normalization by evaluation (NbE) algorithm for $\lambda^{\to\Box}$ which we prove sound and complete. The NbE algorithm is a moderate extension to the standard presheaf model of simply typed λ-calculus. However, central to the model construction and the NbE algorithm is the observation of *unified (simultaneous) substitutions* on context stacks which brings together two previously separate concepts, structural modal transformations on context stacks and substitutions for individual assumptions. Moreover, unified substitutions allow us to give a formulation for contextual types, which can represent open code in a meta-programming setting. Our work lays the foundation for extending the logical foundation by Pfenning, Wong, and Davies towards building a practical, dependently typed foundation for meta-programming.

Keywords: modal λ-calculus, normalization by evaluation, presheaf model, contextual types

1 Introduction

The Curry-Howard correspondence fundamentally connects formulas and proofs to types and programs. This view not only provides logical explanations for computational phenomena, but also serves as a guiding principle in designing type theories and programming languages.

Extending the Curry-Howard correspondence to modal logic has been fraught with challenges. One of the first such calculi for the modal logic S4 were proposed by Bierman and de Paiva [7,8] and subsequently by Pfenning and Davies [33]. A key characteristic of this work is to separate the assumptions that are valid in every world from the assumptions that presently hold in the current world. This leads to a dual-context style formulation that satisfies substitution properties (see for example [17]).

In recent years, modal type systems based on this dual-context style have received renewed attention and provided insights into a wide range of seemingly unconnected areas: from reasoning about universes in homotopy type theory [29,37] to mechanizing meta-theory [35,36], to reasoning about effects [38], and meta-programming [26]. This line of work builds on the dual-context formulation of Pfenning and Davies. However, due to the permutation conversions it is also challenging to extend to dependent type theories and directly prove normalization via logical relations. An alternative to dual-context-style modal calculi

[1] Email: zhong.s.hu@mail.mcgill.ca
[2] Email: bpientka@cs.mcgill.ca

is pursued by Clouston, Birkedal and collaborators (see [13,23]). This line of work is inspired by the Fitch-style proof representation given by Borghuis [12]. Following Borghuis they have been calling their representation Fitch style. Fitch-style systems model Kripke semantics [28] and use locks to manage assumptions in a context. To date, existing formulations of $S4$ in Fitch style [13,23] mainly considers idempotency where $\Box T$ is isomorphic to $\Box\Box T$. However, this distinction is important from a computational view. For example, in multi-staged programming (see [33,15]) $\Box T$ describes code generated in one stage, while $\Box\Box T$ denotes code generated in two stages. It is also fruitful to keep the distinction from a theoretical point of view, as it allows for a fine grained study of different, related modal logics.

In this paper, we take $\lambda^{\to\Box}$, an intuitionistic version of modal logic $S4$, from Pfenning, Wong and Davies [34,15] as a starting point. Historically, $\lambda^{\to\Box}$ is also motivated by Kripke semantics [28] (see [14, Section 3] and [15, Section 4]) and is hence referred to as Kripke style. Unlike the Fitch-style systems where worlds are represented by segments between two adjacent "lock" symbols, in $\lambda^{\to\Box}$, each world is represented by a context in a context stack. Nevertheless, the conversion between Kripke and Fitch styles is largely straightforward [3]. Here, we will often use "context" and "world" interchangeably. In $\lambda^{\to\Box}$, a term t is typed in a context stack $\overrightarrow{\Gamma}$ where initially, the context stack consists of a single local context which is itself empty (i.e. $\epsilon;\cdot$).

$$\epsilon;\Gamma_1;\ldots;\Gamma_n \vdash t : T \qquad \text{or} \qquad \overrightarrow{\Gamma} \vdash t : T$$

The rightmost (or topmost) context represents the current world. In the $\Box$ introduction rule, we extend the context stack with a new world (i.e. new context). In the elimination rule, if $\Box T$ is true in a context stack $\overrightarrow{\Gamma}$, then T is true in any worlds $\overrightarrow{\Gamma};\Delta_1;\ldots;\Delta_n$ reachable from $\overrightarrow{\Gamma}$. The choice of the level n corresponds to reflexivity and transitivity of the accessibility relation between worlds in the Kripke semantics.

$$\frac{\overrightarrow{\Gamma};\cdot \vdash t : T}{\overrightarrow{\Gamma} \vdash \texttt{box}\ t : \Box T} \qquad \frac{\overrightarrow{\Gamma} \vdash t : \Box T}{\overrightarrow{\Gamma};\Delta_1;\ldots;\Delta_n \vdash \texttt{unbox}_n\ t : T}$$

There are two key advantages of this `unbox` formulation in $\lambda^{\to\Box}$. First, it introduces a syntactic convenience to use natural numbers to describe levels and therefore allows us to elegantly capture various modal logics differing only in one parameter of the `unbox` rule. By introducing `unbox` levels, $\Box$ is naturally non-idempotent. Having `unbox` allows us to study the relation of various sublogics of $S4$ and treat them uniformly and compactly.

Axiom \ System	K	T	$K4$	$S4$
K: $\Box(S \longrightarrow T) \to \Box S \to \Box T$	✓	✓	✓	✓
T: $\Box T \to T$		✓		✓
4: $\Box T \to \Box\Box T$			✓	✓
`unbox` level (UL) n	$\{\ 1\ \}$	$\{\ 0, 1\ \}$	$\mathbb{N}^+$	$\mathbb{N}$

Second, compared to dual-context formulation, it directly corresponds to computational idioms quote (`box`) and unquote (`unbox`) in practice, thereby giving a logical foundation to multi-staged meta-programming [15]. In particular, allowing $n = 0$ gives us the power to not only generate code, but also to run and evaluate code.

A major stumbling block in reasoning about $\lambda^{\to\Box}$ (see also [19]) is the fact that it is not obvious how to define substitution properties for context stacks. This prevents us from formulating an explicit substitution calculus for $\lambda^{\to\Box}$ which may serve as an efficient implementation. More importantly, it also seems to be the bottleneck in developing normalization proofs for $\lambda^{\to\Box}$ that can be easily adapted to the various subsystems of S4.

In this paper, we make the following contributions:

[3] However, we note that $\lambda^{\to\Box}$ has never been identified as or called a Fitch-style system.

$\boxed{\overrightarrow{\Gamma} \vdash t : T}$ Term t has type T in context stack $\overrightarrow{\Gamma}$

$$\frac{x : T \in \Gamma}{\overrightarrow{\Gamma}; \Gamma \vdash x : T} \qquad \frac{\overrightarrow{\Gamma}; \cdot \vdash t : T}{\overrightarrow{\Gamma} \vdash \mathtt{box}\ t : \Box T} \qquad \frac{\overrightarrow{\Gamma} \vdash t : \Box T \qquad |\overrightarrow{\Delta}| = n}{\overrightarrow{\Gamma}; \overrightarrow{\Delta} \vdash \mathtt{unbox}_n\ t : T} \qquad \frac{\overrightarrow{\Gamma}; \Gamma, x : S \vdash t : T}{\overrightarrow{\Gamma}; \Gamma \vdash \lambda x.t : S \longrightarrow T}$$

$$\frac{\overrightarrow{\Gamma} \vdash t : S \longrightarrow T \qquad \overrightarrow{\Gamma} \vdash s : S}{\overrightarrow{\Gamma} \vdash t\ s : T}$$

$\boxed{\overrightarrow{\Gamma} \vdash t \approx t' : T}$ Terms t and t' have type T and are equivalent in context stack $\overrightarrow{\Gamma}$

β equivalence:

$$\frac{\overrightarrow{\Gamma}; \cdot \vdash t : T \qquad |\overrightarrow{\Delta}| = n}{\overrightarrow{\Gamma}; \overrightarrow{\Delta} \vdash \mathtt{unbox}_n\ (\mathtt{box}\ t) \approx t\{n/0\} : T} \qquad \frac{\overrightarrow{\Gamma}; (\Gamma, x : S) \vdash t : T \qquad \overrightarrow{\Gamma}; \Gamma \vdash s : S}{\overrightarrow{\Gamma}; \Gamma \vdash (\lambda x.t)s \approx t[s/x] : T}$$

η equivalence:

$$\frac{\overrightarrow{\Gamma} \vdash t : \Box T}{\overrightarrow{\Gamma} \vdash t \approx \mathtt{box}\ (\mathtt{unbox}_1\ t) : \Box T} \qquad \frac{\overrightarrow{\Gamma} \vdash t : S \longrightarrow T}{\overrightarrow{\Gamma} \vdash t \approx \lambda x.(t\ x) : S \longrightarrow T}$$

Fig. 1. Typing judgments and some chosen equivalence judgments

(i) We introduce the concept of of *unified (simultaneous) substitutions* on context stacks (Sec. 3) which combines two previously separate concepts: modal transformations on context stacks (such as modal weakening and fusion) and substitution properties for individual assumptions within a given context.

(ii) We extend the standard presheaf model [5] for simply typed λ-calculus and obtain a normalization by evaluation (NbE) algorithm for $\lambda^{\to\Box}$ in Sec. 4. One critical feature of our development is that the algorithm and the proof accommodate all four subsystems of S4 *without change*.

(iii) As opposed to Nanevski et al. [31], we provide a contextual type formulation in $\lambda^{\to\Box}$ inspired by our notion of unified substitutions in Sec. 5 which can serve as a construct for describing open code in a meta-programming setting.

This work opens the door to a substitution calculus and normalization of a dependently typed modal type theory. There are A partial formalization [24] of this work in Agda [2,32] and an accompanying technical report [25].

2 Definition of $\lambda^{\to\Box}$

In this section, we introduce the simply typed modal λ-calculus, $\lambda^{\to\Box}$, by Pfenning, Wong and Davies [34,15] more formally. We concentrate here on the fragment containing function types $S \longrightarrow T$, the necessity modality $\Box T$, and a base type B.

S, T	$:= B \mid \Box T \mid S \longrightarrow T$	Types, Typ
l, m, n		$\mathtt{unbox}$ levels or offsets, $\mathbb{N}$
x, y		Variables, Var
s, t, u	$:= x \mid \mathtt{box}\ t \mid \mathtt{unbox}_n\ t \mid \lambda x.t \mid s\ t$	Terms, Exp
Γ, Δ, Φ	$:= \cdot \mid \Gamma, x : T$	Contexts, Ctx
$\overrightarrow{\Gamma}, \overrightarrow{\Delta}$	$:= \epsilon \mid \overrightarrow{\Gamma}; \Gamma$	Context stack, $\overrightarrow{\mathsf{Ctx}}$
w	$:= v \mid \mathtt{box}\ w \mid \lambda x.w$	Normal form, Nf
v	$:= x \mid v\ w \mid \mathtt{unbox}_n\ v$	Neutral form, Ne

Following standard practice, we consider variables, applications, and `unbox` neutral. Functions, boxed terms and neutral terms are normal. Note that we allow reductions under binders and inside boxed terms. As a consequence, a function or a boxed term is normal, if their body is normal.

We define typing rules and type-directed equivalence between terms in Fig. 1. We only show the rules for β and η equivalence of terms, but the full set of rules can be found in Appendix A. We use Barendregt's abstract naming and α renaming to ensure that variables are unique with respect to context stacks. The variable rule asserts that one can only refer to a variable in the current world (the topmost context). In a typing judgment, we require all context stacks to be non-empty, so the topmost context must exist.

From Kripke semantics' point of view, the introduction rule for $\Box$ says a term of $\Box T$ is just a term of T in the next world. The elimination rule brings $\Box T$ from some previous world to the current world. This previous world is determined by the level n. As mentioned earlier, the choice of n determines which logic the system corresponds to. $|\overrightarrow{\Delta}|$ counts the number of contexts in $\overrightarrow{\Delta}$.

To illustrate, we recap how the axioms in Sec. 1 can be described in $\lambda^{\rightarrow\Box}$. K is defined by choosing $n = 1$. Axiom T requires that $n = 0$ and Axiom 4 is only possible when `unbox` levels (ULs) can be > 1.

$$K \quad : \Box(S \longrightarrow T) \to \Box S \to \Box T \qquad T \quad : \Box T \to T \qquad A4 \quad : \Box T \to \Box\Box T$$
$$K\ f\ x := \texttt{box}\ ((\texttt{unbox}_1\ f)(\texttt{unbox}_1\ x)) \qquad T\ x := \texttt{unbox}_0\ x \qquad A4\ x := \texttt{box}\ (\texttt{box}\ (\texttt{unbox}_2\ x))$$

The term equivalence rules are largely standard. In the η rule for $\Box$, we restrict `unbox` to level 1. In the β rule for $\Box$, we rely on the modal transformation [15], written as $\{n/0\}$, which allows us to transform the term t which is well-typed in the context stack $\overrightarrow{\Gamma}; \cdot$ to the context stack $\overrightarrow{\Gamma}; \overrightarrow{\Delta}$. We abuse slightly notation and use ; for both extending a context stack with a context and appending two context stacks. We will discuss modal transformations more later in this section.

2.1 Term Substitutions

A term substitution simply replaces a variable x with a term s in a term t. It simply pushes the substitution inside the subterms of t and avoiding capture using renaming. Below, we simply restate the ordinary term substitution lemma:

Lemma 2.1 (Term Substitution)
If $\overrightarrow{\Gamma}; (\Gamma, x{:}S, \Gamma'); \overrightarrow{\Delta} \vdash t : T$ *and* $\overrightarrow{\Gamma}; (\Gamma, \Gamma') \vdash s : S$, *then* $\overrightarrow{\Gamma}; (\Gamma, \Gamma'); \overrightarrow{\Delta} \vdash t[s/x] : T$.

2.2 Modal Transformations (MoTs)

In addition to the usual structural properties (weakening and contraction) of individual contexts, $\lambda^{\rightarrow\Box}$ also relies on structural properties of context stacks, e.g. in the β rule for $\Box$. In particular, we need to be able to weaken a context stack $\overrightarrow{\Gamma}; \overrightarrow{\Gamma}'$ to $\overrightarrow{\Gamma}; \overrightarrow{\Delta}; \overrightarrow{\Gamma}'$ by splicing in additional contexts $\overrightarrow{\Delta}$ (*modal weakening*). *Modal fusion* allows us to combine two adjacent contexts in a context stack transforming a context stack $\overrightarrow{\Gamma}; \Gamma_0; \Gamma_1; \overrightarrow{\Delta}$ to a context stack $\overrightarrow{\Gamma}; (\Gamma_0, \Gamma_1); \overrightarrow{\Delta}$.

These modal transformations (*MoTs*) require us to relabel the level n associated with the `unbox` eliminator. This is accomplished by the operation $t\{n/l\}$. Assume that t is well-typed in a context stack $\overrightarrow{\Gamma}$. If $n > 0$, then at position l in the stack (i.e. $\overrightarrow{\Gamma} = \overrightarrow{\Gamma}'; \overrightarrow{\Delta}$ and $|\overrightarrow{\Delta}| = l$), we splice in $n - 1$ additional contexts. If $n = 0$, then this can be interpreted as fusing the two adjacent contexts at position l in the stack $\overrightarrow{\Gamma}$.

$$\begin{aligned}
x\{n/l\} &:= x \\
\texttt{box}\ t\{n/l\} &:= \texttt{box}\ (t\{n/l+1\}) \\
\texttt{unbox}_m\ t\{n/l\} &:= \begin{cases} \texttt{unbox}_m\ (t\{n/l-m\}) & \text{if } m \le l \\ \texttt{unbox}_{n+m-1}\ t & \text{if } m > l \end{cases} \\
\lambda x.t\{n/l\} &:= \lambda x.(t\{n/l\}) \\
s\ t\{n/l\} &:= (s\{n/l\})\ (t\{n/l\})
\end{aligned}$$

In the `box` case, l increases by one, as we extend the context stack by a new world. In the `unbox` case, we distinguish cases based on the `unbox` level m. If $m \leq l$, then we simply rearrange the ULs recursively in t. If $m > l$, we only need to adjust the UL and do not recurse on t. MoTs satisfy the following lemma:

Lemma 2.2 (Structural Property of Context Stacks)
If $\overrightarrow{\Gamma}; \Gamma_0; \Delta_0; \cdots; \Delta_l \vdash t : T$, *then* $\overrightarrow{\Gamma}; \Gamma_0; \cdots; (\Gamma_n, \Delta_0); \cdots; \Delta_l \vdash t\{n/l\} : T$.

We call the case where $n = 0$ *modal fusion* or just *fusion*. Other cases are *modal weakening*. These names can be made sense of from the following examples:

- When $n = l = 0$, the lemma states that if $\overrightarrow{\Gamma}; \Gamma_0; \Delta_0 \vdash t : T$, then $\overrightarrow{\Gamma}; (\Gamma_0, \Delta_0) \vdash t\{0/0\} : T$. Notice that Γ_0 and Δ_0 in the premise are fused into one in the conclusion, hence "modal fusion".
- When $n = 2$ and $l = 0$, the lemma states that if $\overrightarrow{\Gamma}; \Gamma_0; \Delta_0 \vdash t : T$, then $\overrightarrow{\Gamma}; \Gamma_0; \Gamma_1; (\Gamma_2, \Delta_0) \vdash t\{2/0\} : T$. A new context Γ_1 is inserted into the stack and the topmost context is (locally) weakened by Γ_2, hence "modal weakening".
- In the β rule for $\Box$, a MoT $\{n/0\}$ is used to transform t in context stack $\overrightarrow{\Gamma}; \cdot$ to $\overrightarrow{\Gamma}; \overrightarrow{\Delta}$.
- When $l > 0$, the leading l contexts are skipped. If $n = 2$, $l = 1$ and $\overrightarrow{\Gamma}; \Gamma_0; \Delta_0; \Delta_1 \vdash t : T$, then $\overrightarrow{\Gamma}; \Gamma_0; \Gamma_1; (\Gamma_2, \Delta_0); \Delta_1 \vdash t\{2/1\} : T$. Here Δ_1 is kept as is.

3 Unified Substitutions

Traditionally, we have viewed term substitutions and modal transformations as two separate operations [15]. This makes reasoning about $\lambda^{\to\Box}$ complex. For example, a composition of n MoTs leads to up to 2^n cases in the `unbox` case. This becomes quickly unwieldy. How can we avoid such case analyses by *unifying* MoTs and term substitutions as one operation that transforms context stacks? – We will view context stacks as a category and these *unified* simultaneous substitutions as morphisms (denoted by $\Rightarrow$). MoTs are then simply a special case of these morphisms. Lemma 2.2 suggests to view a MoT as a morphism:

$$\{n/l\} : \overrightarrow{\Gamma}; \Gamma_0; \cdots; (\Gamma_n, \Delta_0); \cdots; \Delta_l \Rightarrow \overrightarrow{\Gamma}; \Gamma_0; \Delta_0; \cdots; \Delta_l$$

because $\{n/l\}$ moves t from the codomain context stack to the domain context stack. If *unified* substitutions are closed under composition, then a category of context stacks can be organized.

3.1 Composing MoTs

If unified substitutions are morphisms in a category, then they must be able to represent any composition of MoTs. The following diagram is a composition of multiple MoTs, forming a morphism $\overrightarrow{\Gamma} \Rightarrow \overrightarrow{\Delta}$:

$$\begin{array}{lllllllll} \overrightarrow{\Gamma} := \epsilon; & \Gamma_0; & \Delta_0; & \Delta_1; & (\Gamma_1, \Gamma_2, \Gamma_3); & \Delta_2; & \Gamma_4 \\ \Downarrow \quad\quad & \downarrow \quad\quad & & & \downarrow\ \downarrow\ \downarrow & & \downarrow \quad\quad \\ \overrightarrow{\Delta} := \epsilon; & \Gamma_0; & & & \Gamma_1; \Gamma_2; \Gamma_3; & & \Gamma_4 \end{array}$$

This composition contains both fusion (Γ_1, Γ_2, Γ_3) and modal weakening (Δ_0, Δ_1, Δ_2). The thin arrows correspond to contexts in both stacks. Some of these arrows are local identities (Γ_0 and Γ_4). They happen between contexts that are affected by modal weakenings. The rest are local weakenings (Γ_1, Γ_2 and Γ_3); in this case, they are affected by modal fusions. The first observation is that the size of gaps between adjacent thin arrows varies, because it is determined by different MoTs. Another observation is that thin arrows do not have to be just local weakenings; if they contain general terms, then we obtain a general simultaneous substitution. Moreover, thanks to the thin arrows, we know exactly which context stack each term should be well-typed in. Combining all information, we arrive at the definition of *unified substitutions*:

Definition 3.1 A unified substitution $\overrightarrow{\sigma}$, or just **substitution**, between context stacks is defined as

$$\begin{array}{lll} \sigma, \delta & := () \mid \sigma, t/x & \text{Local substitutions, } \mathsf{Subst} \\ \overrightarrow{\sigma}, \overrightarrow{\delta} & := \varepsilon; \sigma \mid \overrightarrow{\sigma}; \Uparrow^n \sigma & \text{Unified substitutions, } \mathsf{Substs} \end{array}$$

$$\frac{\sigma : \overrightarrow{\Gamma} \Rightarrow \Gamma}{\varepsilon; \sigma : \overrightarrow{\Gamma} \Rightarrow \epsilon; \Gamma} \qquad \frac{\overrightarrow{\sigma} : \overrightarrow{\Gamma} \Rightarrow \overrightarrow{\Delta} \qquad |\overrightarrow{\Gamma}'| = n \qquad \sigma : \overrightarrow{\Gamma}; \overrightarrow{\Gamma}' \Rightarrow \Delta}{\overrightarrow{\sigma}; \Uparrow^n \sigma : \overrightarrow{\Gamma}; \overrightarrow{\Gamma}' \Rightarrow \overrightarrow{\Delta}; \Delta}$$

where a local substitution $\sigma : \overrightarrow{\Gamma} \Rightarrow \Gamma$ is defined as a list of well-typed terms in $\overrightarrow{\Gamma}$ for all bindings in Γ.

Just as context stacks must be non-empty and consist of at least one context, a unified substitution must have a topmost local substitution written as $\varepsilon; \sigma$ in the base case. It provides a mapping for the context stack $\epsilon; \Gamma$. We extend a unified substitution $\overrightarrow{\sigma}$ with $\Uparrow^n \sigma$ where n captures the offsets due to MoTs and σ is the local substitution. To illustrate, the morphism in the previous diagram can be represented as $\varepsilon; \mathsf{id}; \Uparrow^3 \mathsf{wk}_1; \Uparrow^0 \mathsf{wk}_2; \Uparrow^0 \mathsf{wk}_3; \Uparrow^2 \mathsf{id}$ where id is the local identity substitution and $\mathsf{wk}_i : \Gamma_0; \Delta_0; \Delta_1; (\Gamma_1, \Gamma_2, \Gamma_3) \Rightarrow \Gamma_i$ are appropriate local weakenings. We break down this representation:

(i) We start with $\varepsilon; \mathsf{id} : \epsilon; \Gamma_0 \Rightarrow \epsilon; \Gamma_0$.

(ii) We add an offset 3 and a local weakening wk_1, forming $\varepsilon; \mathsf{id}; \Uparrow^3 \mathsf{wk}_1 : \epsilon; \Gamma_0; \Delta_0; \Delta_1; (\Gamma_1, \Gamma_2, \Gamma_3) \Rightarrow \epsilon; \Gamma_0; \Gamma_1$. The offset 3 adds three contexts to the domain stack (Δ_0, Δ_1 and $\Gamma_1, \Gamma_2, \Gamma_3$). Local weakening wk_1 extracts Γ_1 from $\Gamma_1, \Gamma_2, \Gamma_3$.

(iii) We extend the substitution to $\varepsilon; \mathsf{id}; \Uparrow^3 \mathsf{wk}_1; \Uparrow^0 \mathsf{wk}_2 : \epsilon; \Gamma_0; \Delta_0; \Delta_1; (\Gamma_1, \Gamma_2, \Gamma_3) \Rightarrow \epsilon; \Gamma_0; \Gamma_1; \Gamma_2$. Since the offset associated with wk_2 is 0, no context is added to the domain stack. This effectively represents fusion. wk_2 is similar to wk_1.

(iv) The rest of the substitution proceeds similarly.

Subsequently, we may simply write $\overrightarrow{\sigma}; \sigma$ instead of $\overrightarrow{\sigma}; \Uparrow^1 \sigma$. In particular, we will write $\overrightarrow{\sigma}; \mathsf{wk}$ instead of $\overrightarrow{\sigma}; \Uparrow^1 \mathsf{wk}$ and $\overrightarrow{\sigma}; \mathsf{id}$ instead of $\overrightarrow{\sigma}; \Uparrow^1 \mathsf{id}$. We often omit offsets that are 1 for readability.

3.2 Representing MoTs

Now we show that MoTs are a special case of substitutions. Let $l := |\overrightarrow{\Delta}|$. We define modal weakenings as

$$\begin{aligned} \{n+1/l\} &: \overrightarrow{\Gamma}; \Gamma_1; \cdots; (\Gamma_{n+1}, \Delta_0); \overrightarrow{\Delta} \Rightarrow \overrightarrow{\Gamma}; \Delta_0; \overrightarrow{\Delta} \\ \{n+1/l\} &= \varepsilon; \underbrace{\mathsf{id}; \cdots; \mathsf{id}}_{|\overrightarrow{\Gamma}|}; \Uparrow^{n+1} \mathsf{wk}; \underbrace{\mathsf{id}; \cdots; \mathsf{id}}_{|\overrightarrow{\Delta}|} \end{aligned}$$

where the offset $n+1$ on the right adds $\Gamma_1; \cdots; (\Gamma_{n+1}, \Delta_0)$ to $\overrightarrow{\Gamma}$ in the domain stack and $\mathsf{wk} : \overrightarrow{\Gamma}; \Gamma_1; \cdots; (\Gamma_{n+1}, \Delta_0) \Rightarrow \Delta_0$.

Fusion is also easily defined:

$$\begin{aligned} \{0/l\} &: \overrightarrow{\Gamma}; (\Gamma_1, \Gamma_2); \overrightarrow{\Delta} \Rightarrow \overrightarrow{\Gamma}; \Gamma_1; \Gamma_2; \overrightarrow{\Delta} \\ \{0/l\} &= \varepsilon; \underbrace{\mathsf{id}; \cdots; \mathsf{id}}_{|\overrightarrow{\Gamma}|}; \mathsf{wk}_1; \Uparrow^0 \mathsf{wk}_2; \underbrace{\mathsf{id}; \cdots; \mathsf{id}}_{|\overrightarrow{\Delta}|} \end{aligned}$$

where the offset 0 associated with wk_2 allows us to fuse Γ_1 and Γ_2, and $\mathsf{wk}_i : \overrightarrow{\Gamma}; (\Gamma_1, \Gamma_2) \Rightarrow \Gamma_i$ for $i \in \{1, 2\}$.

3.3 Operations of Unified Substitutions

We now show that unified substitutions are morphisms in a category of context stacks. In order to define composition, we describe two essential operations: 1) *truncation* ($\overrightarrow{\sigma} \mid n$) drops n topmost substitutions

from a unified substitution $\overrightarrow{\sigma}$ and 2) *truncation offset* ($\mathcal{O}(\overrightarrow{\sigma}, n)$) computes the *total* number of contexts that need to be dropped from the domain context stack, given that we truncate $\overrightarrow{\sigma}$ by n. It computes the sum of n leading offsets. Let $\overrightarrow{\sigma} := \overrightarrow{\sigma}'; \Uparrow^{m_n}\sigma_n; \ldots; \Uparrow^{m_1}\sigma_1$, then $\mathcal{O}(\overrightarrow{\sigma}, n) = m_n + \ldots + m_1$ and $\overrightarrow{\sigma} \mid n = \overrightarrow{\sigma}'$. For the operation to be meaningful, n must be less than $|\Delta|$.

$$
\begin{array}{ll}
\text{Truncation Offset} & \mathcal{O}(\_,\_) : (\overrightarrow{\Gamma} \Rightarrow \overrightarrow{\Delta}) \to \mathbb{N} \to \mathbb{N} \\
\mathcal{O}(\overrightarrow{\sigma}, 0) & := 0 \\
\mathcal{O}(\overrightarrow{\sigma}; \Uparrow^n \sigma, 1 + m) & := n + \mathcal{O}(\overrightarrow{\sigma}, m)
\end{array}
$$

Truncation simply drops n local substitutions regardless of the offset that is associated with each local substitution.

$$
\begin{array}{ll}
\text{Truncation} & \_ \mid \_ : (\overrightarrow{\sigma} : \overrightarrow{\Gamma} \Rightarrow \overrightarrow{\Delta}) \to (n : \mathbb{N}) \to \overrightarrow{\Gamma} \mid \mathcal{O}(\overrightarrow{\sigma}, n) \Rightarrow \overrightarrow{\Delta} \mid n \\
\overrightarrow{\sigma} \mid 0 & := \overrightarrow{\sigma} \\
(\overrightarrow{\sigma}; \Uparrow^m \sigma) \mid 1 + n & := \overrightarrow{\sigma} \mid n
\end{array}
$$

Similar to truncation of substitutions, we rely on truncation of contexts, written as $\overrightarrow{\Gamma} \mid n$ which simply drops n contexts from the context stack $\overrightarrow{\Gamma}$, i.e. if $\overrightarrow{\Gamma} = \overrightarrow{\Gamma}'; \Gamma_1; \ldots; \Gamma_n$, then $\overrightarrow{\Gamma} \mid n = \overrightarrow{\Gamma}'$. Note that n must satisfy $n < |\overrightarrow{\Gamma}|$, otherwise the operation would not be meaningful. We emphasize that no further restrictions are placed on n and hence our definitions apply to any of the combinations of Axioms K, T and 4 described in the introduction.

3.4 Unified Substitution Operation

We now can define the substitution operation as follows:

$$
\begin{array}{lll}
x[\varepsilon; \sigma] & := \sigma(x) & \text{lookup } x \text{ in } \sigma \\
x[\overrightarrow{\sigma}; \Uparrow^k \sigma] & := \sigma(x) & \\
(\texttt{box}\ t)[\overrightarrow{\sigma}] & := \texttt{box}\ t[\overrightarrow{\sigma}; \Uparrow^1()] & \\
(\texttt{unbox}_n\ t)[\overrightarrow{\sigma}] & := \texttt{unbox}_{\mathcal{O}(\overrightarrow{\sigma}, n)}\ (t[\overrightarrow{\sigma} \mid n]) & \\
(\lambda x.t)[\varepsilon; \sigma] & := \lambda x.t[\varepsilon; (\sigma, x/x)] & \\
(\lambda x.t)[\overrightarrow{\sigma}; \Uparrow^k \sigma] & := \lambda x.t[\overrightarrow{\sigma}; \Uparrow^k (\sigma, x/x)] & \\
(s\ t)[\overrightarrow{\sigma}] & := s[\overrightarrow{\sigma}]\ u[\overrightarrow{\sigma}] &
\end{array}
$$

In the `box` case, the recursive call adds to $\overrightarrow{\sigma}$ an empty local substitution. Note that the offset must be 1, since we extend in the box-introduction rule our context stack with a new empty context.

The `unbox` case for unified substitutions incorporates MoTs. Instead of distinguishing cases based on the unbox level n, we use the truncation offset operation to re-compute the UL and the recursive call $t[\overrightarrow{\sigma} \mid n]$ continues with $\overrightarrow{\sigma}$ truncated, because t is typed in a shorter stack.

Due to the typing invariants, we know that $\mathcal{O}(\overrightarrow{\sigma}, n)$ is indeed defined for all valid UL n in all our target systems. This fact can be checked easily. In System K, since $n = 1$ and all MoT offsets in $\overrightarrow{\sigma}$ are 1, we have that $\mathcal{O}(\overrightarrow{\sigma}, n) = 1$. In System T where $n \in \{0, 1\}$ and $\overrightarrow{\sigma}$ only contains MoT offsets 0 and 1, we have $\mathcal{O}(\overrightarrow{\sigma}, n) \in \{0, 1\}$. In System $K4$ where $n \geq 1$, $\mathcal{O}(\overrightarrow{\sigma}, n)$ cannot be 0 and thus $\mathcal{O}(\overrightarrow{\sigma}, n) \geq 1$. In System $S4$, since $n \in \mathbb{N}$, $\mathcal{O}(\overrightarrow{\sigma}, n) \in \mathbb{N}$ naturally holds.

The following lemma shows that substitutions are indeed the proper notion we seek:

Lemma 3.2 *If* $\overrightarrow{\Gamma} \vdash t : T$ *and* $\overrightarrow{\sigma} : \overrightarrow{\Delta} \Rightarrow \overrightarrow{\Gamma}$, *then* $\overrightarrow{\Delta} \vdash t[\overrightarrow{\sigma}] : T$.

$$\begin{array}{lll}
[\![_]\!] : \mathsf{Typ} \to \mathcal{W}^{op} \Rightarrow \mathcal{S}et & [\![_]\!] : \mathsf{Ctx} \to \mathcal{W}^{op} \Rightarrow \mathcal{S}et & [\![_]\!] : \overrightarrow{\mathsf{Ctx}} \to \mathcal{W}^{op} \Rightarrow \mathcal{S}et \\
[\![B]\!] := \mathsf{Ne}\ B & [\![\cdot]\!] := \hat{\top} & [\![\epsilon; \Gamma]\!] := \hat{\top} \hat{\times} [\![\Gamma]\!] \\
[\![\Box T]\!] := \hat{\Box}[\![T]\!] & [\![\Gamma, x : T]\!] := [\![\Gamma]\!] \hat{\times} [\![T]\!] & [\![\overrightarrow{\Gamma}; \Gamma]\!]_{\overrightarrow{\Delta}} := (\Sigma n < |\overrightarrow{\Delta}|.[\![\overrightarrow{\Gamma}]\!]_{\overrightarrow{\Delta}|_n}) \times [\![\Gamma]\!]_{\overrightarrow{\Delta}} \\
[\![S \longrightarrow T]\!] := [\![S]\!] \hat{\longrightarrow} [\![T]\!] & & \text{(where the offset } n \text{ depends on UL)}
\end{array}$$

Fig. 2. Interpretations of types, contexts and context stacks to presheaves

3.5 Categorical Structure

We are now ready to organize unified simultaneous substitutions into a category. First we define the identity substitution:

$$\begin{array}{l}
\overrightarrow{\mathsf{id}}_{\overrightarrow{\Gamma}} : \overrightarrow{\Gamma} \Rightarrow \overrightarrow{\Gamma} \\
\overrightarrow{\mathsf{id}}_{\overrightarrow{\Gamma}} := \varepsilon; \mathsf{id}; \mathsf{id}; \cdots; \mathsf{id}
\end{array}$$

where id's are appropriate local identities. We again omit the offsets when we extend unified substitutions with id, since they are 1. We also omit the subscript $\overrightarrow{\Gamma}$ on $\overrightarrow{\mathsf{id}}$ for readability.

Composition is defined in terms of application:

$$\begin{array}{rl}
_ \circ _ & : \overrightarrow{\Gamma}' \Rightarrow \overrightarrow{\Gamma}'' \to \overrightarrow{\Gamma} \Rightarrow \overrightarrow{\Gamma}' \to \overrightarrow{\Gamma} \Rightarrow \overrightarrow{\Gamma}'' \\
(\varepsilon; \sigma) \circ \overrightarrow{\delta} & := \varepsilon; (\sigma[\overrightarrow{\delta}]) \\
(\overrightarrow{\sigma}; \Uparrow^n \sigma) \circ \overrightarrow{\delta} & := (\overrightarrow{\sigma} \circ (\overrightarrow{\delta} \mid n)); \Uparrow^{\mathcal{O}(\overrightarrow{\delta}, n)} (\sigma[\overrightarrow{\delta}])
\end{array}$$

where $\sigma[\overrightarrow{\delta}]$ iteratively applies $\overrightarrow{\delta}$ to all terms in σ. In the recursive case, we continue with a truncated substitution $\overrightarrow{\delta} \mid n$ and recompute the offset.

Verification of the categorical laws is then routine:

Theorem 3.3 *Context stacks and substitutions form a category with identities and composition defined as above.*

3.6 Properties of Truncation and Truncation Offset

Finally, we summarize some critical properties of truncation and truncation offset. Let $\overrightarrow{\sigma} : \overrightarrow{\Gamma}' \Rightarrow \overrightarrow{\Gamma}$ and $\overrightarrow{\delta} : \overrightarrow{\Gamma}'' \Rightarrow \overrightarrow{\Gamma}'$:

Lemma 3.4 *If $n < |\overrightarrow{\Gamma}|$, then $\mathcal{O}(\overrightarrow{\sigma}, n) < |\overrightarrow{\Gamma}'|$.*

Lemma 3.5 (Distributivity of Addition) *If $n + m < |\overrightarrow{\Gamma}|$, then $\overrightarrow{\sigma} \mid (n+m) = (\overrightarrow{\sigma} \mid n) \mid m$ and $\mathcal{O}(\overrightarrow{\sigma}, n+m) = \mathcal{O}(\overrightarrow{\sigma}, n) + \mathcal{O}(\overrightarrow{\sigma} \mid n, m)$.*

Lemma 3.6 (Distributivity of Composition) *If $n < |\overrightarrow{\Gamma}|$, then $\mathcal{O}(\overrightarrow{\sigma} \circ \overrightarrow{\delta}, n) = \mathcal{O}(\overrightarrow{\delta}, \mathcal{O}(\overrightarrow{\sigma}, n))$ and $(\overrightarrow{\sigma} \circ \overrightarrow{\delta}) \mid n = (\overrightarrow{\sigma} \mid n) \circ (\overrightarrow{\delta} \mid \mathcal{O}(\overrightarrow{\sigma}, n))$.*

These properties will be used in Sec. 4. Later we will define other instances of truncation and truncation offsets but all these instances satisfy properties listed here. Therefore, these properties sufficiently characterize an algebra of truncation and truncation offset.

4 Normalization: A Presheaf Model

In this section, we present our NbE algorithm based on a presheaf model. Once we determine the base category, the rest of the construction is largely standard following Altenkirch et al. [5] with minor differences, which we will highlight.

To construct the presheaf model, we first determine the base category. Then we interpret types, contexts and context stacks to presheaves and terms to natural transformations. After that, we define two operations, reification and reflection, and use them to define the NbE algorithm. Last, we briefly discuss the completeness and soundness proof. The algorithm is implemented in Agda [24].

4.1 Unified Weakenings

In the simply typed λ-calculus (STLC), the base category is the category of weakenings. In $\lambda^{\to\Box}$, we must consider the effects of MoTs and we will use the more general notion of *unified weakenings* which characterizes how a well-typed term in $\lambda^{\to\Box}$ moves from one context stack to another.

Definition 4.1 A unified weakening $\overrightarrow{\gamma} : \overrightarrow{\Gamma} \Rightarrow_w \overrightarrow{\Delta}$ is:

$$\overrightarrow{\gamma} := \varepsilon \mid q(\overrightarrow{\gamma}) \mid p(\overrightarrow{\gamma}) \mid \overrightarrow{\gamma}; \Uparrow^n \qquad \text{(Unified weakenings)}$$

$$\frac{}{\varepsilon : \epsilon; \cdot \Rightarrow_w \epsilon; \cdot} \qquad \frac{\overrightarrow{\gamma} : \overrightarrow{\Gamma}; \Gamma \Rightarrow_w \overrightarrow{\Delta}; \Delta}{q(\overrightarrow{\gamma}) : \overrightarrow{\Gamma}; (\Gamma, x : T) \Rightarrow_w \overrightarrow{\Delta}; (\Delta, x : T)} \qquad \frac{\overrightarrow{\gamma} : \overrightarrow{\Gamma}; \Gamma \Rightarrow_w \overrightarrow{\Delta}; \Delta}{p(\overrightarrow{\gamma}) : \overrightarrow{\Gamma}; (\Gamma, x : T) \Rightarrow_w \overrightarrow{\Delta}; \Delta}$$

$$\frac{\overrightarrow{\gamma} : \overrightarrow{\Gamma} \Rightarrow_w \overrightarrow{\Delta} \qquad |\overrightarrow{\Gamma}'| = n}{\overrightarrow{\gamma}; \Uparrow^n \ : \overrightarrow{\Gamma}; \overrightarrow{\Gamma}' \Rightarrow_w \overrightarrow{\Delta}; \cdot} \text{(the offset } n \text{ depends on UL)}$$

The q constructor is the identity extension of the weakening $\overrightarrow{\gamma}$, while p accommodates weakening of an individual context. These constructors are typical in the category of weakenings [5, Definition 2]. To accommodate MoTs, we add to the category of weakenings the last rule which transforms a context stack. In the last rule, the offset n is again parametricin, subject to the *same* UL as the syntactic system, and its choice determines which modal logic the system corresponds to. Note that we also write $\overrightarrow{\mathsf{id}}$ for the identity unified weakening. Following our truncation and truncation offset operations for unified substitutions in Sec. 3, we can easily define these operations together with composition also for unified weakenings. We omit these definitions for brevity and we simply note that a truncated unified weakening remains a unified weakening. Now we obtain the base category:

Lemma 4.2 *Unified Weakenings form a category $\mathcal{W}$.*

4.2 Presheaves

The NbE proof is built on the presheaf category $\widehat{\mathcal{W}}$ over $\mathcal{W}$. $\widehat{\mathcal{W}}$ has presheaves $\mathcal{W}^{op} \Rightarrow \mathcal{S}et$ as objects and natural transformations as morphisms. We know from the Yoneda lemma that two presheaves F and G can form a presheaf exponential $F \hat{\longrightarrow} G$:

$$F \hat{\longrightarrow} G \qquad : \mathcal{W}^{op} \Rightarrow \mathcal{S}et$$
$$(F \hat{\longrightarrow} G)_{\overrightarrow{\Gamma}} := \forall \overrightarrow{\Delta} \Rightarrow_w \overrightarrow{\Gamma}. F_{\overrightarrow{\Delta}} \to G_{\overrightarrow{\Delta}}$$

It is natural in $\overrightarrow{\Delta}$. As a convention, we use subscripts for both functorial applications and natural transformation components. As in [5], presheaf exponentials model functions. To model $\Box$, we define

$$\hat{\Box} F : \mathcal{W}^{op} \Rightarrow \mathcal{S}et$$
$$(\hat{\Box} F)_{\overrightarrow{\Gamma}} := F_{\overrightarrow{\Gamma},\cdot}$$

where F is a presheaf. In Kripke semantics, $\hat{\Box}$ takes F to the next world. Unlike presheaf exponentials which always exist regardless of the base category, $\hat{\Box}$ requires the base category to have the notion of "the next world". This dependency in turn allows us to embed the Kripke structure of context stacks into the base category, so that our presheaf model can stay a moderate extension of the standard construction [5].

Evaluation $[\![\_]\!] : \overrightarrow{\Gamma} \vdash t : T \to [\![\overrightarrow{\Gamma}]\!] \Rightarrow [\![T]\!]$

Expanded form $[\![t]\!]_{\overrightarrow{\Delta}} : [\![\overrightarrow{\Gamma}]\!]_{\overrightarrow{\Delta}} \to [\![T]\!]_{\overrightarrow{\Delta}}$

$[\![x]\!]_{\overrightarrow{\Delta}}((\_,\rho)) := \rho(x)$ lookup x in ρ

$[\![\texttt{box}\ t]\!]_{\overrightarrow{\Delta}}(\overrightarrow{\rho}) := [\![t]\!]_{\overrightarrow{\Delta};\cdot}(((1,\overrightarrow{\rho}),*))$

$[\![\texttt{unbox}_n\ t]\!]_{\overrightarrow{\Delta}}(\overrightarrow{\rho}) := [\![t]\!]_{\overrightarrow{\Delta}|_m}(\overrightarrow{\rho} \mid n)[\overrightarrow{\mathsf{id}}; \Uparrow^m]$ where $m := \mathcal{O}(\overrightarrow{\rho}, n)$ and $\overrightarrow{\mathsf{id}}; \Uparrow^m : \overrightarrow{\Delta} \Rightarrow_w \overrightarrow{\Delta} \mid m; \cdot$

$[\![\lambda x.t]\!]_{\overrightarrow{\Delta}}(\overrightarrow{\rho}) := (\overrightarrow{\gamma} : \overrightarrow{\Delta}' \Rightarrow_w \overrightarrow{\Delta})(a) \mapsto [\![t]\!]_{\overrightarrow{\Delta}'}((\pi,(\rho,a)))$ where $(\pi,\rho) := \overrightarrow{\rho}[\overrightarrow{\gamma}]$

$[\![t\ s]\!]_{\overrightarrow{\Delta}}(\overrightarrow{\rho}) := [\![t]\!]_{\overrightarrow{\Delta}}(\overrightarrow{\rho}, \overrightarrow{\mathsf{id}}_{\overrightarrow{\Delta}}, [\![s]\!]_{\overrightarrow{\Delta}}(\overrightarrow{\rho}))$

Reification $\downarrow^T : [\![T]\!] \Rightarrow \mathsf{Nf}\ T$

$\downarrow^B_{\overrightarrow{\Gamma}}(a) := a$

$\downarrow^{\Box T}_{\overrightarrow{\Gamma}}(a) := \texttt{box}\ \downarrow^T_{\overrightarrow{\Gamma};\cdot}(a)$ notice $a \in (\hat{\Box}[\![T]\!])_{\overrightarrow{\Gamma}} = [\![T]\!]_{\overrightarrow{\Gamma};\cdot}$

$\downarrow^{S\longrightarrow T}_{\overrightarrow{\Gamma};\Gamma}(a) := \lambda x.\downarrow^T_{\overrightarrow{\Gamma};(\Gamma,x:S)}(a\ (p(\overrightarrow{\mathsf{id}})\ ,\ \uparrow^S_{\overrightarrow{\Gamma};(\Gamma,x:S)}(x)))$ where $p(\overrightarrow{\mathsf{id}}) : \overrightarrow{\Gamma};\Gamma, x{:}S \Rightarrow_w \overrightarrow{\Gamma};\Gamma$

Reflection $\uparrow^T : \mathsf{Ne}\ T \Rightarrow [\![T]\!]$

$\uparrow^B_{\overrightarrow{\Gamma}}(v) := v$

$\uparrow^{\Box T}_{\overrightarrow{\Gamma}}(v) := \uparrow^T_{\overrightarrow{\Gamma};\cdot}(\texttt{unbox}_1\ v)$

$\uparrow^{S\longrightarrow T}_{\overrightarrow{\Gamma};\Gamma}(v) := (\overrightarrow{\gamma} : \overrightarrow{\Delta} \Rightarrow_w \overrightarrow{\Gamma};\Gamma)(a) \mapsto \uparrow^T_{\overrightarrow{\Delta}}(v[\overrightarrow{\gamma}]\ \downarrow^S_{\overrightarrow{\Delta}}(a))$

Fig. 3. Evaluation, reification and reflection functions

With this setup, we give the interpretations of types, contexts, and context stacks in Fig. 2. The interpretation of the base type B is the presheaf from context stacks to neutral terms of type B. We write $\mathsf{Ne}\ T\ \overrightarrow{\Gamma}$ for the set of neutral terms of type T in stack $\overrightarrow{\Gamma}$. $\mathsf{Ne}\ T$ then is the presheaf $\overrightarrow{\Gamma} \mapsto \mathsf{Ne}\ T\ \overrightarrow{\Gamma}$. $\mathsf{Nf}\ T\ \overrightarrow{\Gamma}$ and $\mathsf{Nf}\ T$ are defined similarly. The case $[\![\Box T]\!] := \hat{\Box}[\![T]\!]$ states that semantically, a value of $[\![\Box T]\!]$ is just a value of $[\![T]\!]$ in the next world, which implicitly relies on unified weakening's capability of expressing MoTs. $\hat{\top}$ are $\hat{\times}$ are a chosen terminal object and products in $\widehat{\mathcal{W}}$ and $*$ is *the only element* in the chosen singleton set.

The interpretation of context stacks is more interesting. In the step case, $\overrightarrow{\Gamma};\Gamma$ is interpreted as a product. To extract both part of $\overrightarrow{\rho} \in [\![\overrightarrow{\Gamma}]\!]_{\overrightarrow{\Delta}}$, we write $(\pi,\rho) := \overrightarrow{\rho}$ where $(n,\overrightarrow{\rho}') := \pi$. The first component, namely π, again consists of two parts: 1) the level n satisfying $n < |\overrightarrow{\Delta}|$ which corresponds to the MoTs that we support. We note that our definitions again apply to any of the combinations of Axioms K, T and 4 depending on the choice of n. 2) the recursive interpretation of $\overrightarrow{\Gamma}$ in the truncated stack $\overrightarrow{\Delta} \mid n$ described by $\overrightarrow{\rho}'$. This stack truncation is necessary to interpret `unbox`.

The second component, namely ρ, describes the interpretation of the top-most context Γ. The fact that our interpretation of context stacks stores the level n ultimately justifies the offsets stored in unified substitutions.

Lemma 4.3 (Functoriality) *$[\![T]\!]$, $[\![\Gamma]\!]$ and $[\![\overrightarrow{\Gamma}]\!]$ are presheaves.*

Functoriality means the interpretations also act on morphisms in $\mathcal{W}$. Given $\overrightarrow{\gamma} : \overrightarrow{\Gamma} \Rightarrow_w \overrightarrow{\Delta}$ and $a \in [\![T]\!]_{\overrightarrow{\Delta}}$, we write $a[\overrightarrow{\gamma}] \in [\![T]\!]_{\overrightarrow{\Gamma}}$. We intentionally overload the notation for applying substitutions to draw a connection. This notation also applies for morphism actions of $[\![\Gamma]\!]$ and $[\![\overrightarrow{\Gamma}]\!]$.

4.3 Evaluation

The interpretation of well-typed terms to natural transformations, or *evaluation* (see Fig. 3), relies on truncation and the truncation offset. These operations are defined below and follow the same principles that lie behind the corresponding operations for syntactic substitutions.

$$\begin{array}{ll} \text{Truncation Offset } \mathcal{O}(\_,\_) : [\![\overrightarrow{\Gamma}]\!]_{\overrightarrow{\Delta}} \to \mathbb{N} \to \mathbb{N} & \text{Truncation } \_ \mid \_ : (\overrightarrow{\rho} : [\![\overrightarrow{\Gamma}]\!]_{\overrightarrow{\Delta}})\ (n : \mathbb{N}) \to [\![\overrightarrow{\Gamma} \mid n]\!]_{\overrightarrow{\Delta}|\mathcal{O}(\overrightarrow{\rho},n)} \\ \mathcal{O}(\overrightarrow{\rho}, 0) := 0 & \overrightarrow{\rho} \mid 0 := \overrightarrow{\rho} \\ \mathcal{O}(((n, \overrightarrow{\rho}), \rho), 1 + m) := n + \mathcal{O}(\overrightarrow{\rho}, m) & ((n, \overrightarrow{\rho}), \rho) \mid 1 + m := \overrightarrow{\rho} \mid m \end{array}$$

Most cases in evaluation are straightforward. In the `box` case the recursion continues with an extended environment and t in the next world. In the `unbox` case, we first recursively interpret t with a truncated environment and then the result is unified-weakened. This is because from the well-typedness of t, we know $\overrightarrow{\Gamma} \mid n \vdash t : \Box T$, so $[\![t]\!]_{\overrightarrow{\Delta}|m}(\overrightarrow{\rho} \mid n)$ gives an element in set $[\![\Box T]\!]_{\overrightarrow{\Delta}|m} = [\![T]\!]_{\overrightarrow{\Delta}|m;\cdot}$. To obtain our goal $[\![T]\!]_{\overrightarrow{\Delta}}$, we can apply monotonicity of $[\![T]\!]$ using a morphism $\overrightarrow{\Delta} \Rightarrow_w \overrightarrow{\Delta} \mid m; \cdot$, which is given by $\overrightarrow{\mathsf{id}}; \Uparrow^m$. The cases related to functions are identical to [5]. In the λ case, since we need to return a set function due to presheaf exponentials, we use $\mapsto$ to construct this function. We first weaken the environment $\overrightarrow{\rho}$ and then extend it with the input value a. In the application case, since $[\![t]\!]$ gives us a presheaf exponential, we just need to apply it to $[\![s]\!]$. We simply supply $\overrightarrow{\mathsf{id}}_{\overrightarrow{\Delta}}$ for the unified weakening argument because no extra weakening is needed.

The following lemma proves that $[\![t]\!]$ is a natural transformation in $\overrightarrow{\Delta}$:

Lemma 4.4 (Naturality) *If $\overrightarrow{\Gamma} \vdash t : T$ and $\overrightarrow{\rho} \in [\![\overrightarrow{\Gamma}]\!]_{\overrightarrow{\Delta}}$, then for all unified weakenings $\overrightarrow{\gamma} : \overrightarrow{\Delta}' \Rightarrow_w \overrightarrow{\Delta}$, we have $[\![t]\!]_{\overrightarrow{\Delta}'}(\overrightarrow{\rho}[\overrightarrow{\gamma}]) = [\![t]\!]_{\overrightarrow{\Delta}}(\overrightarrow{\rho})[\overrightarrow{\gamma}]$.*

The lemma states that the result of evaluation in a unified-weakened environment is the same as unified-weakening the result evaluated in the original environment. In STLC, despite being a fact, naturality is not used anywhere in the proof. In $\lambda^{\to\Box}$, since unified weakenings encode MoTs, naturality is necessary in the completeness proof.

After evaluation, we obtain a semantic value of the semantic type $[\![T]\!]$. In the last step, we use a *reification* function to convert the semantic value back to a normal form. Reification is defined mutually with *reflection* in Fig. 3. As suggested by their signatures, they are both natural transformations, but our proof does not rely on this fact. Both reification and reflection are type-directed, so after reification we obtain $\beta\eta$ normal forms. We reify a semantic value a of box type $\Box T$ in a context stack $\overrightarrow{\Gamma}$ recursively extending the context stack to $\overrightarrow{\Gamma}; \cdot$. Note that a has the semantic type $(\hat{\Box}[\![T]\!])_{\overrightarrow{\Gamma}}$ which is defined as $[\![T]\!]_{\overrightarrow{\Gamma};\cdot}$. In the case of function type $S \longrightarrow T$, since a is a presheaf exponential, we supply a unified weakening and a value, the result of which is then recursively reified.

Reflection turns neutral terms into semantic values. We reflect neutral terms of type $\Box T$ recursively and incrementally extending the context stack with one context at a time. In the function case, to construct a presheaf exponential, we first take two arguments $\overrightarrow{\gamma}$ and a. Since v is a neutral term, $v[\overrightarrow{\gamma}]$ is also neutral but now well-typed in $\overrightarrow{\Delta}$. Both recursive calls to reification and reflection then go down to $\overrightarrow{\Delta}$ instead.

Normalization by evaluation (NbE) takes a well-typed term t in a context stack $\overrightarrow{\Gamma}$ as input, interprets t to its semantic counterpart in the initial environment, and reifies it back. Before defining NbE more formally, we define the identity environment $[\![\overrightarrow{\Gamma}]\!]_{\overrightarrow{\Gamma}}$ that is used as the initial environment:

$$\begin{array}{ll} \uparrow & : (\overrightarrow{\Gamma} : \overrightarrow{\mathsf{Ctx}}) \to [\![\overrightarrow{\Gamma}]\!]_{\overrightarrow{\Gamma}} \\ \uparrow^{\epsilon;\cdot} & := (*, *) \\ \uparrow^{\overrightarrow{\Gamma};\cdot} & := ((1, \uparrow^{\overrightarrow{\Gamma}}), *) \\ \uparrow^{\overrightarrow{\Gamma};\Gamma,x:T} & := (\pi, (\rho, \uparrow^{T}_{\overrightarrow{\Gamma};\Gamma,x:T}(x))) \ \text{ where } (\pi, \rho) :=\uparrow^{\overrightarrow{\Gamma};\Gamma}\ [p(\overrightarrow{\mathsf{id}})] \end{array}$$

Finally we define the NbE algorithm:

Definition 4.5 (Normalization by Evaluation) If $\overrightarrow{\Gamma} \vdash t : T$, then

$$\mathsf{nbe}^{T}_{\overrightarrow{\Gamma}}(t) := \downarrow^{T}_{\overrightarrow{\Gamma}} (\llbracket t \rrbracket_{\overrightarrow{\Gamma}}(\uparrow^{\overrightarrow{\Gamma}}))$$

4.4 Completeness and Soundness

The algorithm given above is sound and complete:

Theorem 4.6 (Completeness) *If* $\overrightarrow{\Gamma} \vdash t \approx t' : T$, *then* $\mathit{nbe}^{T}_{\overrightarrow{\Gamma}}(t) = \mathit{nbe}^{T}_{\overrightarrow{\Gamma}}(t')$.

Theorem 4.7 (Soundness) *If* $\overrightarrow{\Gamma} \vdash t : T$, *then* $\overrightarrow{\Gamma} \vdash t \approx \mathit{nbe}^{T}_{\overrightarrow{\Gamma}}(t) : T$.

Due to space limitation, we are not able to present the whole proof. Fortunately, the proof is very standard [5]. To prove completeness, we simply need to prove that equivalent terms always evaluate to the same natural transformation:

Lemma 4.8 *If* $\overrightarrow{\Gamma} \vdash t \approx t' : T$, *then for any* $\overrightarrow{\rho} \in \llbracket \overrightarrow{\Gamma} \rrbracket_{\overrightarrow{\Delta}}$, $\llbracket t \rrbracket_{\overrightarrow{\Delta}}(\overrightarrow{\rho}) = \llbracket t' \rrbracket_{\overrightarrow{\Delta}}(\overrightarrow{\rho})$.

Proof. Induct on $\overrightarrow{\Gamma} \vdash t \approx t' : T$ and apply naturality in most cases about $\Box$. □

The soundness proof is established by a *Kripke gluing model*. The gluing model $t \sim a \in (\!|T|\!)_{\overrightarrow{\Gamma}}$ relates a syntactic term t and a natural transformation a, so that after reifying a, the resulting normal form is equivalent to t:

$$\begin{aligned}
(\!|T|\!)_{\overrightarrow{\Gamma}} &\subseteq \mathsf{Exp} \times \llbracket T \rrbracket_{\overrightarrow{\Gamma}} \\
t \sim a \in (\!|B|\!)_{\overrightarrow{\Gamma}} &:= \overrightarrow{\Gamma} \vdash t \approx a : B \\
t \sim a \in (\!|\Box T|\!)_{\overrightarrow{\Gamma}} &:= \overrightarrow{\Gamma} \vdash t : \Box T \text{ and } \forall \overrightarrow{\Delta}.\mathtt{unbox}_{|\overrightarrow{\Delta}|}\ t \sim a[\overrightarrow{\mathsf{id}}; \Uparrow^{|\overrightarrow{\Delta}|}] \in (\!|T|\!)_{\overrightarrow{\Gamma};\overrightarrow{\Delta}} \\
t \sim a \in (\!|S \longrightarrow T|\!)_{\overrightarrow{\Gamma}} &:= \overrightarrow{\Gamma} \vdash t : S \longrightarrow T \text{ and } \forall \overrightarrow{\gamma} : \overrightarrow{\Delta} \Rightarrow_w \overrightarrow{\Gamma}, s \sim b \in (\!|S|\!)_{\overrightarrow{\Delta}}.t[\overrightarrow{\gamma}]\ s \sim a(\overrightarrow{\gamma}, b) \in (\!|T|\!)_{\overrightarrow{\Delta}}
\end{aligned}$$

The gluing model should be monotonic in $\overrightarrow{\Gamma}$, hence Kripke. Again the gluing model is very standard [5]. It is worth mentioning that in the $\Box T$ case, the Kripke predicate effectively requires that t and a are related only when their results of *any* `unbox`ing remains related. We then can move on to prove some properties of the gluing model and define its generalization to substitutions, which eventually allow us to conclude the soundness theorem. Please find more details in our technical report.

4.5 Adaptiveness

We emphasize that our construction is stable no matter our choice of UL. Hence, our construction applies to all four modal systems, K, T, $K4$ and $S4$ that we introduced in Sec. 1 *without change*. The key insight that allows us to keep our construction and model generic is the fact that unified substitutions, unified weakenings, and $\llbracket \overrightarrow{\Gamma} \rrbracket$ are instances of the algebra formed by truncation and truncation offsets and satisfy all the properties, in particular identity and distributivity, listed at the end of Sec. 3. More importantly, all the truncation and truncation offset functions are defined for all choices of UL thereby accommodating all four modal systems with their varying level of `unbox`ing.

5 Contextual Types

In $S4$ and a meta-programming setting, $\Box$ is interpreted as stages, where a term of type $\Box T$ is considered as a term of type T but available only in the next stage. However, as pointed out in [15,31], $\Box$ only characterizes closed code. Nanevski et al. [31] propose contextual types which relativize the surrounding context of a term so representing open code becomes possible. However, this notion of contextual types is

in dual-context style and how contextual types can be formulated with `unbox` and context stacks remains open. In this section, we answer this question by utilizing our notion of unified substitutions.

5.1 Typing Judgments and Semi-substitutions

With contextual types, we augment the syntax as follows:

$$S, T := \cdots \mid \lceil \overrightarrow{\Delta} \vdash T \rceil \qquad\qquad s, t, u := \cdots \mid \lceil \overrightarrow{\Delta} \vdash t \rceil \mid \lfloor t \rfloor_{\overrightarrow{\sigma}}$$

$\lceil \overrightarrow{\Delta} \vdash T \rceil$ is a contextual type. It captures a list of contexts $\overrightarrow{\Delta}$ which a term of type T can be open in. Note that $\overrightarrow{\Delta}$ here can be empty. This notion of contextual types is very general and captures a term open in multiple stages. $\lceil \overrightarrow{\Delta} \vdash t \rceil$ is the constructor of a contextual type, where the contexts that it captures are specified. $\lfloor t \rfloor_{\overrightarrow{\sigma}}$ is the eliminator. Instead of an `unbox` level, we now require a different argument $\overrightarrow{\sigma}$, which is a *semi-substitution* storing `unbox` offsets and terms. We will discuss more very shortly.

The introduction rule for contextual types is straightforward:

$$\frac{\overrightarrow{\Gamma}; \overrightarrow{\Delta} \vdash t : T}{\overrightarrow{\Gamma} \vdash \lceil \overrightarrow{\Delta} \vdash t \rceil : \lceil \overrightarrow{\Delta} \vdash T \rceil}$$

If we let $\overrightarrow{\Delta} = \epsilon; \cdot$, then we recover $\Box$. If we let $\overrightarrow{\Delta} = \epsilon; \Delta$ for some Δ, then we have an open term t which uses only assumptions in the same stage. If $\overrightarrow{\Delta}$ has more contexts, then t is an open term which uses assumptions from previous stages. We can also let $\overrightarrow{\Delta} = \epsilon$. In this case, $\lceil \epsilon \vdash T \rceil$ is isomorphic to T and is not too meaningful but allowing so makes our formulation mathematically cleaner.

The elimination rule, on the other hand, becomes significantly more complex:

$$\frac{\overrightarrow{\Gamma} \mid \mathcal{O}(\overrightarrow{\sigma}) \vdash t : \lceil \overrightarrow{\Delta} \vdash T \rceil \qquad \overrightarrow{\sigma} : \overrightarrow{\Gamma} \Rightarrow_s \overrightarrow{\Delta}}{\overrightarrow{\Gamma} \vdash \lfloor t \rfloor_{\overrightarrow{\sigma}} : T}$$

It is no longer enough to eliminate with just an `unbox` level because the eliminator must specify how to replace all variables in $\overrightarrow{\Delta}$ and how contexts in $\overrightarrow{\Gamma}$ and $\overrightarrow{\Delta}$ relate. This information is collectively stored in a *semi-substitution* $\overrightarrow{\sigma}$ (notice the semi-arrow), which intuitively is not yet a valid substitution, but close:

Definition 5.1 A semi-substitution $\overrightarrow{\sigma}$ is defined as follows:

$$\overrightarrow{\sigma}, \overrightarrow{\delta} := \varepsilon \mid \overrightarrow{\sigma}; \Uparrow^n \sigma \qquad \text{Semi-substitutions, } \mathsf{SSubsts}$$

$$\frac{}{\varepsilon : \overrightarrow{\Gamma} \Rightarrow_s \epsilon} \qquad\qquad \frac{\overrightarrow{\sigma} : \overrightarrow{\Gamma} \Rightarrow_s \overrightarrow{\Delta} \qquad |\overrightarrow{\Gamma}'| = n \qquad \sigma : \overrightarrow{\Gamma}; \overrightarrow{\Gamma}' \Rightarrow \Delta}{\overrightarrow{\sigma}; \Uparrow^n \sigma : \overrightarrow{\Gamma}; \overrightarrow{\Gamma}' \Rightarrow_s \overrightarrow{\Delta}; \Delta}$$

Compared to unified substitutions, semi-substitutions differ in the base case, where empty ε is permitted, so they are not valid substitutions. However, if a semi-substitution is prepended by an identity substitution, then the result is a valid substitution. Also, $\mathcal{O}(\overrightarrow{\sigma})$ computes the sum of all offsets in $\overrightarrow{\sigma}$:

$$\begin{array}{lll} \overrightarrow{\mathsf{id}}; & : \mathsf{SSubsts} \to \mathsf{Substs} & \mathcal{O}(\_) \quad : \mathsf{SSubsts} \to \mathbb{N} \\ \overrightarrow{\mathsf{id}}; \varepsilon & := \overrightarrow{\mathsf{id}} & \mathcal{O}(\varepsilon) \quad := 0 \\ \overrightarrow{\mathsf{id}}; (\overrightarrow{\sigma}; \Uparrow^n \sigma) & := (\overrightarrow{\mathsf{id}}; \overrightarrow{\sigma}); \Uparrow^n \sigma & \mathcal{O}(\overrightarrow{\sigma}; \Uparrow^n \sigma) := \mathcal{O}(\overrightarrow{\sigma}) + n \end{array}$$

We can prove the following lemma:

Lemma 5.2 *If* $\overrightarrow{\sigma} : \overrightarrow{\Gamma} \Rightarrow_s \overrightarrow{\Delta}$*, then* $\overrightarrow{id}; \overrightarrow{\sigma} : \overrightarrow{\Gamma} \Rightarrow_s \overrightarrow{\Gamma} \mid \mathcal{O}(\overrightarrow{\sigma}); \overrightarrow{\Delta}$.

This lemma is needed to justify the β equivalence rule which we are about to discuss.

5.2 Equivalence of Contextual Types

Having defined the introduction and elimination rules, we are ready to describe how they interact. Note that the congruence rules are standard so we omit them here and only describe the β and η rules:

$$\frac{\overrightarrow{\Gamma} \mid \mathcal{O}(\overrightarrow{\sigma}); \overrightarrow{\Delta} \vdash t : T \qquad \overrightarrow{\sigma} : \overrightarrow{\Gamma} \Rightarrow_s \overrightarrow{\Delta}}{\overrightarrow{\Gamma} \vdash \lfloor \lceil \overrightarrow{\Delta} \vdash t \rceil \rfloor_{\overrightarrow{\sigma}} \approx t[\overrightarrow{\mathsf{id}}; \overrightarrow{\sigma}] : T} \qquad \frac{\overrightarrow{\Gamma} \vdash t : \lceil \overrightarrow{\Delta} \vdash T \rceil}{\overrightarrow{\Gamma} \vdash t \approx \lceil \overrightarrow{\Delta} \vdash \lfloor t \rfloor_{\overrightarrow{\mathsf{id}}} \rceil : \lceil \overrightarrow{\Delta} \vdash T \rceil}$$

In the η rule, $\overrightarrow{\mathsf{id}}$ denotes the identity semi-substitution, which is defined as

$$\overrightarrow{\mathsf{id}}_{\overrightarrow{\Delta}} : \overrightarrow{\Gamma}; \overrightarrow{\Delta} \Rightarrow_s \overrightarrow{\Delta}$$
$$\overrightarrow{\mathsf{id}}_{\overrightarrow{\Delta}} := \varepsilon; \underbrace{\mathsf{id}; \cdots ; \mathsf{id}}_{|\overrightarrow{\Delta}|}$$

We omit the subscript whenever possible. Both rules are easily justified. In the β rule, since t is typed in the context stack $\overrightarrow{\Gamma} \mid \mathcal{O}(\overrightarrow{\sigma}); \overrightarrow{\Delta}$, we obtain a term in $\overrightarrow{\Gamma}$ by applying $\overrightarrow{\mathsf{id}}; \overrightarrow{\sigma}$ due to Lemma 5.2. In the η rule, by definition, we know $(\overrightarrow{\Gamma}; \overrightarrow{\Delta}) \mid \mathcal{O}(\overrightarrow{\mathsf{id}}) = \overrightarrow{\Gamma}$ and therefore $\overrightarrow{\Gamma}; \overrightarrow{\Delta} \vdash \lfloor t \rfloor_{\overrightarrow{\mathsf{id}}} : T$.

In an extensional setting, where the constructor and the eliminator of modalities are congruent as done in this paper, we can show that the contextual type $\lceil \epsilon; \Delta_1; \cdots ; \Delta_n \vdash T \rceil$ is isomorphic to $\Box(\Delta_1 \to \cdots \Box(\Delta_n \to T))$ if we view contexts Δ_i as iterative products. This implies introducing contextual types does not increase the logical strength of the system and the system remains normalizing. Nevertheless, contextual types given here seem to have a natural adaptation to dependent types and set a stepping stone towards representing open code with dependent types and therefore a homogeneous, dependently typed meta-programming system.

6 Related Work and Conclusion

6.1 Modal Type Theories

There are many early attempts to give a constructive formulation of modal logic, especially the modal logic S4 starting back in the 1990's [10,9,6,3,12,30]. Pfenning and Davies [14,33] give the first formulation of $S4$ in dual-context style where we separate the assumptions that are valid in every world from assumptions that are true in the current world. This leads to a dual-context style formulation that satisfies substitution properties and has found many applications from staged computation to homotopy type theory (HoTT). For example, Shulman [37] extends idempotent $S4$ with dependent types, called spatial type theory and Licata et al. [29] define crisp type theory, which removes the idempotency from spatial type theory. However, both papers do not give a rigorous justification of their type theories. Most recently Kavvos [27] investigates modal systems based on this dual-context formulation for Systems K, T, $K4$ and $S4$ as well as the Löb induction principle. Kavvos also gives categorical semantics for these systems.

However, it has been difficult to develop direct normalization proofs for these dual-context formulations, since we must handle extensional properties like commuting conversions (c.f. [27,18]). Further, our four target systems have very different formulations in dual-context style as shown by Kavvos [27]. As a consequence, it is challenging to have one single normalization algorithm for all our four target systems.

An alternative to dual-context style is the Fitch-style approach pursued by Clouston, Birkedal and collaborators (see [13,23,11]). At the high-level, Fitch-style systems also model the Kripke semantics, but instead of using one context for each world, Fitch style uses a special symbol (usually 🔒) to segment one context into multiple sections, each of them representing one world. Variables to the left of the rightmost 🔒 are not accessible. Our normalization proof and the generalization of $\lambda^{\to\Box}$ to contextual types also can likely be adapted to those systems.

Clouston [13] gives Systems K and idempotent $S4$ in Fitch style and discusses their categorical semantics. Gratzer et al. [23] describe idempotent $S4$ with dependent types. Birkedal et al. [11] give K with dependent types and formulate dependent right adjoints, an important categorical concept of modalities.

Gratzer et al. [21,22,20] proposes MTT, a multimode type theory, which describes interactions between multiple modalities. Though MTT uses 🔒 to segment contexts, we think that MTT is actually a generalization of dual-context style.

Currently, existing Fitch-style systems usually consider idempotent $S4$ where $\Box T$ is isomorphic to $\Box\Box T$. However, we consider this distinction to be important from a computational view. For example, in multi-staged programming (see [33,15]) $\Box T$ and $\Box\Box T$ describe code generated in one stage and two stages, respectively. Moreover, $\texttt{unbox}_0\ t$ is interpreted as evaluating and running the code generated by t. It is nevertheless possible to develop a non-idempotent $S4$ system using `unbox` levels n in Fitch style by defining a function which truncates a context until its n'th 🔒. This is however more elegantly handled in $\lambda^{\to\Box}$, because worlds are separated syntactically. For this reason, we consider $\lambda^{\to\Box}$ as a more versatile and more suitable foundation for developing a dependently typed meta-programming system. In particular, our extension to contextual types shows how we can elegantly accomodate reasoning about open code which is important in practice.

Though context stacks in $\lambda^{\to\Box}$ are taken from Pfenning, Wong and Davies' development [33,15], Borghuis [12] also uses context stacks in his development of modal pure type systems. The elimination rules use explicit weakening and several "transfer" rules while $\lambda^{\to\Box}$ incorporates both using `unbox` levels, which we consider more convenient and more practical from a programmer's point of view. Martini and Masini [30] also use context stacks. Their system annotates all terms with a level which we consider too verbose to be practical.

6.2 Normalization

For the dual-context style, Nanevski et al. [31] give contextual types and prove normalization by reduction to another logical system with permutation conversions [16]. This means that the proof is indirect and does not directly yield an algorithm for normalizing terms. Kavvos [27] gives a rewriting-based normalization proof for dual-context style systems with Löb induction. Most recently, Gratzer [20] proves the normalization for MTT. It is not clear to us whether techniques in [20] scale to dependently typed Kripke-style systems.

Most closely related to our approach is the NbE algorithm of idempotent $S4$ in Fitch style given by Gratzer et al. [23]. As we do, the authors follow Abel [1]. Since their correctness proof is parameterized by an extra layer of PER to model the Kripke world structure introduced by $\Box$, as pointed out in [21], this proof cannot be easily adapted to dependently typed K by Birkedal et al. [11]. In contrast, our model is a moderate extension to the standard construction, requiring no such extra layer and adapting to multiple logics automatically, and we are confident that it will generalize more easily to the dependently typed setting. The ultimate reason is that we *internalize* the Kripke structure of context stacks in the presheaf model. The internalization happens in the base category, where MoTs are encoded as part of unified weakenings. The internalization captures peculiar behaviours of different systems and conflates the extra Kripke structure from context stacks and the standard model construction, so that the proofs become much simpler and closer to the typical construction.

6.3 Conclusion and Future Work

In this paper, we present a normalization-by-evaluation (NbE) algorithm for the simply-typed modal λ-calculus ($\lambda^{\to\Box}$) which covers all four sub-systems of S4. The key to achieving this result is our notion of unified substitutions which provides a unifying account for modal transformations and term substitutions and allows us to formulate a substitution calculus for modal logic S4 and its various sub-systems. Such calculus is not only important from a practical point of view, but play also a central role in our theoretical analysis. Using insights gained from unified substitutions we organize a presheaf model, from which we extract a normalization algorithm. The algorithm can be implemented in conventional programming languages and directly account for the normalization of $\lambda^{\to\Box}$. Deriving from unified substitutions, we are also able to give a formulation for contextual types with `unbox` and context stacks, which had been challenging prior to our observation of unified substitutions and is important for representing open code in a meta-programming setting.

This work serves as a basis for further investigations into coproducts [4] and categorical structure of context stacks. We also see this work as a step towards a Martin-Löf-style modal type theory in which

open code has an internal shallow representation. With a dependently typed extension and contextual types, it would allow us to develop a *homogeneous* meta-programming system with dependent types which has been challenging to achieve.

References

[1] Abel, A., "Normalization by evaluation: dependent types and impredicativity," Habilitation thesis, Ludwig-Maximilians-Universität München (2013).

[2] Agda Team, *Agda 2.6.2* (2021).

[3] Alechina, N., M. Mendler, V. de Paiva and E. Ritter, *Categorical and Kripke Semantics for Constructive S4 Modal Logic*, in: L. Fribourg, editor, *Computer Science Logic*, Lecture Notes in Computer Science (2001), pp. 292–307.

[4] Altenkirch, T., P. Dybjer, M. Hofmann and P. Scott, *Normalization by evaluation for typed lambda calculus with coproducts*, in: *Proceedings 16th Annual IEEE Symposium on Logic in Computer Science*, 2001, pp. 303–310, iSSN: 1043-6871.

[5] Altenkirch, T., M. Hofmann and T. Streicher, *Categorical reconstruction of a reduction free normalization proof*, in: D. Pitt, D. E. Rydeheard and P. Johnstone, editors, *Category Theory and Computer Science* (1995), pp. 182–199.

[6] Bellin, G., V. C. V. de Paiva and E. Ritter, *Extended Curry-Howard Correspondence for a Basic Constructive Modal Logic*, in: *In Proceedings of Methods for Modalities*, 2001.

[7] Bierman, G. and V. de Paiva, *Intuitionistic necessity revisited*, Technical Report CSR9610, University of Birmingham (1996).

[8] Bierman, G. M. and V. de Paiva, *On an intuitionistic modal logic*, Studia Logica **65** (2000), pp. 383–416.
URL https://doi.org/10.1023/A:1005291931660

[9] Bierman, G. M. and V. de Paiva, *On an Intuitionistic Modal Logic*, Studia Logica **65** (2000), pp. 383–416.
URL https://doi.org/10.1023/A:1005291931660

[10] Bierman, G. M. and V. C. V. de Paiva, *Intuitionistic Necessity Revisited*, Technical report, University of Birmingham (1996).

[11] Birkedal, L., R. Clouston, B. Mannaa, R. E. Mgelberg, A. M. Pitts and B. Spitters, *Modal dependent type theory and dependent right adjoints*, Mathematical Structures in Computer Science **30** (2020), pp. 118–138, publisher: Cambridge University Press.
URL https://www.cambridge.org/core/journals/mathematical-structures-in-computer-science/article/abs/modal-dependent-type-theory-and-dependent-right-adjoints/215682F46705DE70A67946FE73C95A3E

[12] Borghuis, V. A. J., "Coming to terms with modal logic : on the interpretation of modalities in typed lambda-calculus," PhD Thesis, Mathematics and Computer Science (1994).

[13] Clouston, R., *Fitch-Style Modal Lambda Calculi*, in: C. Baier and U. Dal Lago, editors, *Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science (2018), pp. 258–275.

[14] Davies, R. and F. Pfenning, *A modal analysis of staged computation*, in: H. Boehm and G. L. S. Jr., editors, *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* (1996), pp. 258–270.
URL https://doi.org/10.1145/237721.237788

[15] Davies, R. and F. Pfenning, *A modal analysis of staged computation*, Journal of the ACM **48** (2001), pp. 555–604.
URL https://dl.acm.org/doi/10.1145/382780.382785

[16] de Groote, P., *On the Strong Normalization of Natural Deduction with Permutation-Conversions*, in: P. Narendran and M. Rusinowitch, editors, *Rewriting Techniques and Applications*, Lecture Notes in Computer Science (1999), pp. 45–59.

[17] Ghani, N., V. de Paiva and E. Ritter, *Explicit substitutions for constructive necessity*, in: K. G. Larsen, S. Skyum and G. Winskel, editors, *25th International Colloquium on Automata, Languages and Programming (ICALP'98)*, Lecture Notes in Computer Science **1443** (1998), pp. 743–754.
URL https://doi.org/10.1007/BFb0055098

[18] Girard, J.-Y., "Proofs and types," Number 7 in Cambridge tracts in theoretical computer science, Cambridge University Press, Cambridge [England] ; New York, 1989.

[19] Goubault-Larrecq, J., *On computational interpretations of the modal logic s4: Ii. the λevq-calculus*, Technical report, Univeristy of Karlsruhe (1996).

[20] Gratzer, D., *Normalization for multimodal type theory*, in: *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22 (2022).

[21] Gratzer, D., G. A. Kavvos, A. Nuyts and L. Birkedal, *Multimodal Dependent Type Theory*, in: *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '20 (2020), pp. 492–506.
URL https://doi.org/10.1145/3373718.3394736

[22] Gratzer, D., G. A. Kavvos, A. Nuyts and L. Birkedal, *Multimodal Dependent Type Theory*, Log. Methods Comput. Sci. **17** (2021).

[23] Gratzer, D., J. Sterling and L. Birkedal, *Implementing a modal dependent type theory*, Proceedings of the ACM on Programming Languages **3** (2019), pp. 107:1–107:29.
URL https://doi.org/10.1145/3341711

[24] Hu, J. Z. S. and B. Pientka, *Agda mechanization.*
URL https://hustmphrrr.github.io/Kripke-style/Unbox.README.html

[25] Hu, J. Z. S. and B. Pientka, *An Investigation of Kripke-style Modal Type Theories* (2022), number: arXiv:2206.07823 arXiv:2206.07823 [cs].
URL http://arxiv.org/abs/2206.07823

[26] Jang, J., S. Gélineau, S. Monnier and B. Pientka, *Moebius: Metaprogramming using contextual types – the stage where system f can pattern match on itself*, Proc. ACM Program. Lang. (PACMPL) (2022).

[27] Kavvos, G. A., *Dual-context calculi for modal logic*, in: *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, 2017, pp. 1–12.

[28] Kripke, S. A., *Semantical Analysis of Modal Logic I Normal Modal Propositional Calculi*, Mathematical Logic Quarterly **9** (1963), pp. 67–96, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/malq.19630090502.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/malq.19630090502

[29] Licata, D. R., I. Orton, A. M. Pitts and B. Spitters, *Internal Universes in Models of Homotopy Type Theory*, in: H. Kirchner, editor, *3rd International Conference on Formal Structures for Computation and Deduction (FSCD 2018)*, Leibniz International Proceedings in Informatics (LIPIcs) **108** (2018), pp. 22:1–22:17, iSSN: 1868-8969.
URL http://drops.dagstuhl.de/opus/volltexte/2018/9192

[30] Martini, S. and A. Masini, *A Computational Interpretation of Modal Proofs*, in: H. Wansing, editor, *Proof Theory of Modal Logic*, Applied Logic Series, Springer Netherlands, Dordrecht, 1996 pp. 213–241.
URL https://doi.org/10.1007/978-94-017-2798-3_12

[31] Nanevski, A., F. Pfenning and B. Pientka, *Contextual modal type theory*, ACM Transactions on Computational Logic **9** (2008), pp. 23:1–23:49.
URL https://doi.org/10.1145/1352582.1352591

[32] Norell, U., "Towards a practical programming language based on dependent type theory," PhD Thesis, Chalmers University of Technology (2007).

[33] Pfenning, F. and R. Davies, *A judgmental reconstruction of modal logic*, Mathematical Structures in Computer Science **11** (2001).
URL http://www.journals.cambridge.org/abstract_S0960129501003322

[34] Pfenning, F. and H.-C. Wong, *On a modal λ-calculus for S4*, in: S. Brookes and M. Main, editors, *Proceedings of the Eleventh Conference on Mathematical Foundations of Programming Semantics*, New Orleans, Louisiana, 1995, *Electronic Notes in Theoretical Computer Science*, Volume 1, Elsevier.

[35] Pientka, B., *A type-theoretic foundation for programming with higher-order abstract syntax and first-class substitutions*, in: *35th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'08)*, 2008, pp. 371–382.

[36] Pientka, B., A. Abel, F. Ferreira, D. Thibodeau and R. Zucchini, *A type theory for defining logics and proofs*, in: *34th IEEE/ ACM Symposium on Logic in Computer Science (LICS'19)* (2019), pp. 1–13.

[37] Shulman, M., *Brouwer's fixed-point theorem in real-cohesive homotopy type theory*, Mathematical Structures in Computer Science **28** (2018), pp. 856–941, publisher: Cambridge University Press.

[38] Zyuzin, N. and A. Nanevski, *Contextual modal types for algebraic effects and handlers*, Proc. ACM Program. Lang. **5** (2021), pp. 1–29.
URL https://doi.org/10.1145/3473580

A Equivalence Rules

$$\frac{\overrightarrow{\Gamma} \vdash s \approx t : T}{\overrightarrow{\Gamma} \vdash t \approx s : T} \qquad \frac{\overrightarrow{\Gamma} \vdash s \approx t : T \qquad \overrightarrow{\Gamma} \vdash t \approx u : T}{\overrightarrow{\Gamma} \vdash s \approx u : T} \qquad \frac{x : T \in \Gamma}{\overrightarrow{\Gamma}; \Gamma \vdash x \approx x : T} \qquad \frac{\overrightarrow{\Gamma}; \cdot \vdash t \approx t' : T}{\overrightarrow{\Gamma} \vdash \texttt{box}\ t \approx \texttt{box}\ t' : \Box T}$$

$$\frac{\overrightarrow{\Gamma} \vdash t \approx t' : \Box T \qquad |\overrightarrow{\Delta}| = n}{\overrightarrow{\Gamma}; \overrightarrow{\Delta} \vdash \texttt{unbox}_n\ t \approx \texttt{unbox}_n\ t' : T'} \qquad \frac{\overrightarrow{\Gamma}; (\Gamma, x : S) \vdash t \approx t' : T}{\overrightarrow{\Gamma}; \Gamma \vdash \lambda x.t \approx \lambda x.t' : S \longrightarrow T}$$

$$\frac{\overrightarrow{\Gamma} \vdash t \approx t' : S \longrightarrow T \qquad \overrightarrow{\Gamma} \vdash s \approx s' : S}{\overrightarrow{\Gamma} \vdash t\ s \approx t'\ s' : T} \qquad \frac{\overrightarrow{\Gamma}; \cdot \vdash t : T \qquad |\overrightarrow{\Delta}| = n}{\overrightarrow{\Gamma}; \overrightarrow{\Delta} \vdash \texttt{unbox}_n\ (\texttt{box}\ t) \approx t\{n/0\} : T}$$

$$\frac{\overrightarrow{\Gamma}; (\Gamma, x : S) \vdash t : T \qquad \overrightarrow{\Gamma}; \Gamma \vdash s : S}{\overrightarrow{\Gamma}; \Gamma \vdash (\lambda x.t)s \approx t[s/x] : T} \qquad \frac{\overrightarrow{\Gamma} \vdash t : \Box T}{\overrightarrow{\Gamma} \vdash t \approx \texttt{box}\ \texttt{unbox}_1\ t : \Box T} \qquad \frac{\overrightarrow{\Gamma} \vdash t : S \longrightarrow T}{\overrightarrow{\Gamma} \vdash t \approx \lambda x.(t\ x) : S \longrightarrow T}$$