A Modal Foundation for Meta-Variables

Aleksandar Nanevski, Brigitte Pientka, and Frank Pfenning

Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{aleks,bp,fp}@cs.cmu.edu

Abstract. We report on work in progress regarding a foundation for the notion of meta-variable in logical frameworks and type theories. Our proposal is to treat meta-variables as modal variables in a modal type theory, which is logically clean and justifies several low-level implementation techniques for meta-variables. We also speculate on other logical extensions of our modal type theory, at present without clear applications.

1 Introduction

In recent years, higher-order reasoning systems and logical frameworks have matured and been successful in large-scale applications such as proof-carrying code. Nonetheless, there are still some foundational and implementation issues which are poorly understood. In this report on work in progress we investigate a logical foundation for meta-variables in higher-order setting.

When a type theory or logic is defined it contains intrinsic notions of variable and variable binding, such as λ-abstraction or various forms of quantification, which are subject to α-conversion. This internal form of variable is generally insufficient for advanced proof search procedures, such as those requiring unification. In this case we need *meta-variables* that stand for as yet undetermined terms from the type theory or logic. Meta-variables also turn out to be useful to describe structure editing, where they serve as placeholders for information to be supplied by the programmer, and type reconstruction, where they serve as placeholders for omitted information to be determined from typing constraints.

Despite the practical importance of meta-variables, both their logical foundation and their efficient implementation have not been settled definitively. One of the principal difficulties in the higher-order setting stems from the interplay of ordinary variables, including those that are bound in a term, and meta-variables. As a result, meta-variables are generally not “first-class”. For example, we cannot quantify or abstract meta-variables in the same way we would with ordinary variables.

In this preliminary paper we advance the view that meta-variables may be best understood via a modal type theory. This has several interesting and potentially important consequences. First, since meta-variables have a completely logical explanation they become first-class and we can safely quantify over them.

Second, the modal properties of meta-variables in our type theory immediately suggest an efficient implementation strategy. This serves as a *post hoc* justification for some of the implementation decisions taken in the Twelf system and clarifies several somewhat mysterious invariants maintained in our algorithms. Third, the general paradigm of modal type theory yields new constructs as described in Section 5. At the time of this progress report we do not yet have a definitive application of this construct; we simply note that it exists and has some pleasing internal properties.

The remainder of the paper is organized as follows. In Section 2 we discuss some prior work on meta-variables. This is a highly selective overview and by no means exhaustive. In Section 3 we provide some background material on modal logic which is suitable for generalization in a type theory presented in Section 4. In Section 5 we introduce another extension to include further modal type operators. We conclude with a few remarks on future work in Section 6.

2 Meta-Variables

In the seminal work on higher-order unification by Huet [4], the tension between ordinary variables and meta-variables is resolved by ensuring that meta-variables may never contain free ordinary variables. Intuitively, this means all meta-variables are existentially quantified on the outside, followed by universal quantified ordinary variables on the inside. During higher-order unification existential variables are subject to substitution, while universal variables are treated as constants except that they may not occur free in a substitution for the existential variables.

Unfortunately, meta-variables must usually be created in a context of ordinary variables Γ. This means that any instantiation for the meta-variable is allowed to depend on the variables in Γ but no others. In Huet's approach this can be achieved via *raising*, which we sketch here in the slightly more general setting of dependent types. If a meta-variable $u{:}A$ can depend on ordinary variables $x_1{:}A_1, \ldots, x_n{:}A_n$ we instead introduce a variable $u'{:}\Pi x_1{:}A_1.\ \ldots \Pi x_n{:}A_n.\ A$ and replace u by $u'\, x_1 \ldots x_n$. Substitutions for u' must now be closed. Besides being potentially inefficient, this also suffers from the problem that the notion of (dependent) function type in underlying type theory may not match the intended semantics. Moreover, it does not help in providing a logical reading for meta-variables.

Miller [6] generalizes this idea by allowing a mixed prefix of universally and existentially quantified variables. Raising is then an operation related to Skolemization, replacing $\forall x{:}A.\ \exists y{:}B(x).C(x, y)$ by $\exists y{:}(\Pi x{:}A.\ B(x)).\ \forall x{:}A.\ C(x, y(x))$, but higher-order unification can also be described directly with respect to a mixed prefix.

Unfortunately, in many cases the natural pattern of permitted dependencies cannot be sorted into a linear order. While raising can overcome those difficulties, it reintroduces inefficiencies present in Huet's approach. A more expressive alternative investigated by Dowek *et al.* [1, 2] is to allow each meta-variable to carry a separate context of precisely those variables it is allowed to depend on. In combination with explicit substitutions and de Bruijn indices, it is then possible to allow *grafting*, that is, instantiation of meta-variables without regard to variable capture. This achieves in essence a first-order equational implementation of higher-order unification, but the context-carrying meta-variables are justified purely algorithmically, not logically. A similar remark applies to the operation of pre-cooking which translates from higher-order to first-order equational problems.

In this paper, we propose an abstract view of meta-variables based on modal logic. We use a modality to cleanly distinguish between ordinary variables and meta-variables. Our approach does not require de Bruijn indices, which is seen as a separate implementation issue, and therefore remains higher-order while retaining the advantages of the approach by Dowek *et al.* In particular, substitution for meta-variables can easily be seen to be implementable by grafting and operations such a raising and its inverse, *lowering*, have logical justifications inside the type theory. Moreover, we generalize the representation by Dowek *et al.* from simple types to fully dependent types.

As our view does not require de Bruijn indices, we believe it is easier to understand and reason about and eliminates the need for pre-cooking. Moreover, we do not require closures $M[\sigma]$ as first-class terms, as other general explicit substitution calculi would. It also suggests a clean and efficient way to abstract directly over meta-variables. In related work we have shown [10, 8] that the presented framework provides insight into several important optimizations and implementation techniques such as higher-order pattern unification and higher-order term indexing. In this paper we concentrate on the logical foundation and its properties.

3 Modal Logic

Our development of the dependently typed modal calculus follows the methodology of Pfenning and Davies [7] which in turn is based on Martin-Löf's distinction between judgments and propositions [5]. The key idea is to assign constructive meaning explanations to modal operators by distinguishing between propositions that are true, propositions that are valid, and propositions that are possible. A proposition is valid if its truth does not depend on the truth of any other proposition. A proposition is possible if it is true or if its truth depends on other possible propositions. We defer the discussion of possibility to Section 5, which leaves us with the basic hypothetical judgment

$$A_1 \; valid, \ldots A_n \; valid; B_1 \; true, \ldots, B_m \; true \vdash C \; true$$

which is abbreviated as $\Delta; \Gamma \vdash C \; true$.

Under the multiple-world interpretation of constructive S4, *C valid* corresponds to *C true* in *all* reachable worlds. This means *C true* without any assumptions, except those that are assumed to be true in all worlds. Conversely, if *A valid* then certainly *A true*, since reachability is assumed to be reflexive. This yields the following principles, the first of which has definitional character while the second is a new hypothesis rule.

1. If $\Delta; \cdot \vdash C\ \mathit{true}$ then $\Delta; \Gamma \vdash C\ \mathit{valid}$
2. $\Delta, A\,\mathit{valid}, \Delta'; \Gamma \vdash A\,\mathit{true}$

We can generalize this idea to also capture truth relative to a set of specified assumptions by writing $C\ \mathit{valid}\ \Psi$, where Ψ abbreviates $C_1\ \mathit{true}, \ldots, C_n\ \mathit{true}$. In terms of the multiple world semantics, this means that C is true in any world where C_1 through C_n are all true and we say C is valid relative to the assumptions in Ψ. Hypotheses about relative validity are more complex now, so our general judgment form is

$$A_1\ \mathit{valid}\ \Psi_1, \ldots, A_n\ \mathit{valid}\ \Psi_n; B_1\ \mathit{true}, \ldots, B_m\ \mathit{true} \vdash C\ \mathit{true}.$$

Now if we have an assumption $A\ \mathit{valid}\ \Psi$ we can conclude $A\,\mathit{true}$ only if we can verify all assumptions in Ψ.

$$\frac{\Delta, A\ \mathit{valid}\ \Psi, \Delta'; \Gamma \vdash \Psi}{\Delta, A\ \mathit{valid}\ \Psi, \Delta'; \Gamma \vdash A\ \mathit{true}}\ (*)$$

Here, $\cdots \vdash \Psi$ means that all judgments $C_i\ \mathit{true}$ in Ψ can be derived. In other words, if we know $A\,\mathit{true}$ in Ψ, and all elements in Ψ can be verified from the assumptions in Γ, then we can conclude $A\,\mathit{true}$ in Γ. As we will see in the next section, this transition from one context Ψ to another context Γ corresponds to a substitution from Ψ to Γ.

In the next section we generalize the ideas sketched above from propositional modal logic above to a fully dependent type theory, drawing the connection to the problem of meta-variables that will no doubt be unclear at this point.

4 Modal Type Theory

In our dependently typed modal λ-calculus, we distinguish between modal variables $u{::}(\Psi{\vdash}A)$ for relative validity assumptions $A\ \mathit{valid}\ \Psi$ declared in a modal context, and ordinary variables $x{:}A$ for truth assumptions $A\,\mathit{true}$ declared in an (ordinary) context. Besides the constructs present in LF we also introduce substitutions σ, but only as part of the syntax for occurrences of modal variables. c and a are constants, which are declared in a signature. This is a conservative extension of the LF type theory [3] so we suppress some routine details such as signatures.

$$\begin{array}{rrcl}
\text{Kinds} & K & ::= & \mathsf{type} \mid \Pi x{:}A.\ K \\
\text{Families} & A, B, C & ::= & a \mid A\,M \mid \Pi x{:}A_1.\ A_2 \\
\text{Objects} & M, N & ::= & c \mid x \mid u[\sigma] \mid \lambda x{:}A.\ M \mid M_1\,M_2 \\
\text{Substitutions} & \sigma, \tau & ::= & \cdot \mid \sigma, M/x \\
\text{Contexts} & \Gamma, \Psi & ::= & \cdot \mid \Gamma, x{:}A \\
\text{Modal Contexts} & \Delta & ::= & \cdot \mid \Delta, u{::}(\Psi{\vdash}A)
\end{array}$$

The principal judgments are listed below. We omit similar judgments on types and kinds and all judgments concerning definitional equality which are carried over from LF.

$\Delta; \Gamma \vdash M : A$	Object M has type A
$\Delta; \Gamma \vdash \sigma : \Psi$	Substitution σ matches context Ψ
$\vdash \Delta\ \mathsf{mctx}$	Δ is a valid modal context
$\Delta \vdash \Psi\ \mathsf{ctx}$	Ψ is a valid context

We will tacitly rename bound variables, and maintain that contexts and substitutions declare no variable more than once. Note that substitutions σ are defined only on ordinary variables x and not modal variables u. We write id_Γ for the identity substitution $(x_1/x_1, \ldots, x_n/x_n)$ for a context $\Gamma = (\cdot, x_1{:}A_1, \ldots, x_n{:}A_n)$. We also streamline the calculus slightly by always substituting simultaneously for all ordinary variables. This is not essential, but saves some tedium in relating simultaneous and iterated substitution. The typing rules are given below.

$$\frac{}{\Delta; \Gamma, x{:}A, \Gamma' \vdash x : A} \qquad \frac{\Delta, u{::}(\Psi{\vdash}A), \Delta'; \Gamma \vdash \sigma : \Psi}{\Delta, u{::}(\Psi{\vdash}A), \Delta'; \Gamma \vdash u[\sigma] : [\sigma]A}\ (*)$$

$$\frac{\Delta; \Gamma, x{:}A_1 \vdash M : A_2}{\Delta; \Gamma \vdash \lambda x{:}A_1.\ M : \Pi x{:}A_1.\ A_2} \qquad \frac{\Delta; \Gamma \vdash M_1 : \Pi x{:}A_2.\ A_1 \quad \Delta; \Gamma \vdash M_2 : A_2}{\Delta; \Gamma \vdash M_1\, M_2 : [\mathsf{id}_\Gamma, M_2/x]A_1}$$

$$\frac{}{\Delta; \Gamma \vdash (\cdot) : (\cdot)} \qquad \frac{\Delta; \Gamma \vdash \sigma : \Psi \quad \Delta; \Gamma \vdash M : [\sigma]A}{\Delta; \Gamma \vdash (\sigma, M/x) : (\Psi, x{:}A)}$$

$$\frac{}{\vdash (\cdot)\ \mathsf{mctx}} \qquad \frac{\vdash \Delta\ \mathsf{mctx} \quad \Delta \vdash \Psi\ \ \mathsf{ctx} \quad \Delta; \Psi \vdash A : \mathsf{type}}{\vdash (\Delta, u{::}(\Psi{\vdash}A))\ \mathsf{mctx}}$$

$$\frac{}{\Delta \vdash (\cdot)\ \mathsf{ctx}} \qquad \frac{\Delta \vdash \Psi\ \mathsf{ctx} \quad \Delta; \Psi \vdash A : \mathsf{type}}{\Delta \vdash (\Psi, x{:}A)\ \mathsf{ctx}}$$

Note that the rule for modal variables is the rule (*) presented in the previous section, annotated with proof terms and slightly generalized, because of the dependent type theory we are working in. Note also that modal contexts Δ can have internal dependencies: the types of later modal variables can mention earlier modal variables.

Our convention is that substitutions as defined operations on expressions are written in prefix notation $[\sigma]P$ for an object, family, kind, or substitution P. These operations are capture-avoiding as usual. Moreover, we always assume that all free variables in P are declared in σ. Substitutions that are part of the syntax are written in postfix notation, $u[\sigma]$. Note that such explicit substitutions occur only for variables u labeling relative validity assumptions.

Substitutions are defined in a standard manner. We omit the details at the level of types and kinds for the sake of brevity.

$$
\begin{aligned}
[\sigma]c &= c \\
[\sigma_1, M/x, \sigma_2]x &= M \\
[\sigma](u[\tau]) &= u[[\sigma]\tau] \\
[\sigma](N_1\, N_2) &= ([\sigma]N_1)\,([\sigma]N_2) \\
[\sigma](\lambda y{:}A.\ N) &= \lambda y{:}[\sigma]A.\ [\sigma, y/y]N \text{ provided } y \text{ not declared or free in } \sigma \\
\\
[\sigma](\cdot) &= (\cdot) \\
[\sigma](\tau, N/y) &= ([\sigma]\tau, [\sigma]N/y) \qquad \text{provided } y \text{ not declared or free in } \sigma
\end{aligned}
$$

The side conditions can always be verified by (tacitly) renaming bound variables. We do not need an operation of applying a substitution σ to a context. The last principle makes it clear that $[\sigma]\tau$ corresponds to composition of substitutions, which is sometimes written as $\tau \circ \sigma$.

The following substitution principles for substitutions σ hold. They are suggested by the modal interpretation and proved by simple structural inductions. We elide corresponding principles for families and kinds.

Theorem 1 (Explicit substitutions).

1. *If $\Delta; \Gamma \vdash \sigma : \Psi$ and $\Delta; \Psi \vdash N : C$ then $\Delta; \Gamma \vdash [\sigma]N : [\sigma]C$.*
2. *If $\Delta; \Gamma \vdash \sigma : \Psi$ and $\Delta; \Psi \vdash \tau : \Psi'$ then $\Delta; \Gamma \vdash [\sigma]\tau : \Psi'$.*
3. *$[\sigma]([\tau]M) = [[\sigma]\tau]M$ and $[\sigma]([\tau]\tau') = [[\sigma]\tau]\tau'$*

A new and interesting operation realizes the substitution principles for relative validity in Theorem 2 below. The new operation is denoted as $[\![M/u]\!]$. It substitutes an object M for a *modal* variable u, and we define it below. The substitution is compositional, but two interesting situations arise: when a variable u is encountered, and when we substitute into a λ-abstraction. For sake of brevity, we only give the substitution on objects.

$$
\begin{aligned}
[\![M/u]\!]c &= c \\
[\![M/u]\!]x &= x \\
[\![M/u]\!](u[\sigma]) &= [[\![M/u]\!]\sigma]M \\
[\![M/u]\!](v[\sigma]) &= v[[\![M/u]\!]\sigma] \quad \text{for } u \neq v \\
[\![M/u]\!](N_1\, N_2) &= ([\![M/u]\!]N_1)\,([\![M/u]\!]N_2) \\
[\![M/u]\!](\lambda y{:}A.\ N) &= \lambda y{:}[\![M/u]\!]A.\ [\![M/u]\!]N
\end{aligned}
$$

We remark that the rule for substitution into a λ-abstraction does not require a side condition. This is because the object M is defined in a different context, which is accounted for by the explicit substitutions stored at occurrences of u.

Finally, consider the case of substituting into a closure, which is the critical case of this definition.

$$[\![M/u]\!](u[\sigma]) = [[\![M/u]\!]\sigma]M$$

This is clearly well-founded, because σ is a subexpression (so $[\![M/u]\!]\sigma$ will terminate) and application of an ordinary substitution has been defined previously without reference to the new form of substitution.

Similar to the substitution properties for ordinary explicit substitutions, we now can show that the new substitution operation for relative validity satisfies the substitution principles. This highlights the interplay between ordinary and modal substitutions. Again, this is motivated by the logical interpretation and follows by simple inductions after straightforward generalization to encompass all syntactic categories.

Theorem 2 (Modal substitutions).

1. *If* $\Delta;\Psi \vdash M : A$ *and* $\Delta, u{::}(\Psi{\vdash}A), \Delta'; \Gamma \vdash N : C$
then $\Delta, [\![M/u]\!]\Delta'; [\![M/u]\!]\Gamma \vdash [\![M/u]\!]N : [\![M/u]\!]C$
2. *If* $\Delta;\Psi \vdash M : A$ *and* $\Delta, u{::}(\Psi{\vdash}A), \Delta'; \Gamma \vdash \tau : \Psi'$
then $\Delta, [\![M/u]\!]\Delta'; [\![M/u]\!]\Gamma \vdash [\![M/u]\!]\tau : [\![M/u]\!]\Psi'$
3. $[\![M/u]\!]([\sigma]P) = [[\![M/u]\!]\sigma]([\![M/u]\!]P)$
4. $[\![M/u]\!]([\![N/v]\!]P) = [\![[\![M/u]\!]N/v]\!]([\![M/u]\!]P)$ *if* $u \neq v$ *and* v *not free in* M

As mentioned several times above, in the implementation the modal variables in Δ are used to represent meta-variables (also known as existential variables), while the variables in Γ are ordinary variables (also known as universal variables or parameters). A modal variable $u{::}(\Psi{\vdash}A)$ corresponds to a meta-variable whose whose substitution term M may mention ordinary variables in Ψ and must have type A.

In the implementation, meta-variables are created in an ambient context Ψ and then lowered as described below. We do not explicitly maintain a context Δ of existential variables, but it is important that a proper order for them exists. Dowek *et al.* do not need to consider this issue due to the absence of dependent types.

As mentioned earlier, we often want to enforce that meta-variables are of atomic type. This can be achieved by lowering and raising. Lowering replaces a variable $u{::}(\Psi{\vdash}\Pi x{:}A_1.\ A_2)$ by a new variable $u'{:}(\Psi, x{:}A_1{\vdash}A_2)$. This process is repeated until all existential variables have a type of the form $\Psi \vdash b\, N_1 \ldots N_k$. In our framework, it is justified by the modal substitution principle.

Lemma 1.

1. *(Lowering) If* $\Delta, u{::}(\Psi{\vdash}\Pi x{:}A_1.\ A_2), \Delta'; \Gamma \vdash M : A$
then $\Delta, u'{::}(\Psi, x{:}A_1{\vdash}A_2), \Delta'^{-}; \Gamma^{-} \vdash M^{-} : A^{-}$
where $(P)^{-} = [\![(\lambda x{:}A_1.\ u'[\mathsf{id}_\Psi, x/x])/u]\!]P$.
2. *(Raising) If* $\Delta, u'{::}(\Psi, x{:}A_1{\vdash}A_2), \Delta'; \Gamma \vdash M : A$
then $\Delta, u{::}(\Psi{\vdash}\Pi x{:}A_1.\ A_2), \Delta'^{+}; \Gamma^{+} \vdash M^{+} : A^{+}$
where $(P)^{+} = [\![(u[\mathsf{id}_\Psi]\ x)/u']\!]P$.
3. $()^{+}$ *and* $()^{-}$ *are inverse substitutions (modulo* $\beta\eta$*-conversion).*

Proof. Direct, by weakening and the modal substitution principle. For part (1) we observe that $\Delta, u'{::}(\Psi, x{:}A_1{\vdash}A_2); \Psi \vdash \lambda x{:}A_1.\ u'[\mathsf{id}_\Psi, x/x] : \Pi x{:}A_1.\ A_2$. For part (2) we use instead that $\Delta, u{::}(\Psi{\vdash}\Pi x{:}A_1.\ A_2); \Psi, x{:}A_1 \vdash u[\mathsf{id}_{\Psi}]\, x : A_2$. Part (3) is direct by calculation.

In certain operations, and particularly after type reconstruction, we need to abstract over the existential variables in a term. Since the LF type theory provides no means to quantify over $u{::}(\Psi{\vdash}A)$ we raise such variables until they have the form $u'{::}(\cdot{\vdash}A')$. It turns out that in the context of type reconstruction we can now quantify over them as ordinary variables $x'{:}A'$. However, this is not satisfactory as it requires first raising the type of existential variables for abstraction, and later again lowering the type of existential variables during unification to undo the effect of raising. To efficiently treat existential variables, it seems important that we can directly quantify over modal variables u. We believe it also may simplify the abstraction process itself.

The judgmental reconstruction in terms of modal logic suggests two simple and clean ways to incorporate such variables. One is via a general modal operator $\Box_{\Psi}$, and the other is via a new quantifier $\Pi^{\Box} u{::}(\Psi{\vdash}A_1).\ A_2$. We will briefly discuss these choices.

Categorical validity of propositional constructive S4 is usually formulated by means of proof terms **box** and **let box** with the following typing rules

$$\dfrac{\Delta; \cdot \vdash M : A}{\Delta; \Gamma \vdash \mathbf{box}\ M : \Box A} \qquad \dfrac{\Delta; \Gamma \vdash M : \Box A \quad \Delta, u{::}A; \Gamma \vdash N : B}{\Delta; \Gamma \vdash \mathbf{let\ box}\ u = M\ \mathbf{in}\ N : B}$$

Naive extension to relative validity may be attempted by the following reformulation

$$\dfrac{\Delta; \Psi \vdash M : A}{\Delta; \Gamma \vdash \mathbf{box}\ M : \Box_{\Psi} A} \qquad \dfrac{\Delta; \Gamma \vdash M : \Box_{\Psi} A \quad \Delta, u{::}(\Psi{\vdash}A); \Gamma \vdash N : B}{\Delta; \Gamma \vdash \mathbf{let\ box}\ u = M\ \mathbf{in}\ N : B}$$

However, at the very least such an extension would violate the existence of canonical forms for LF due to the necessary commuting conversion. Moreover, one would have to consider if B in the elimination rule should be allowed to depend on u in which case it would have to be replaced in the conclusion.

Therefore, proof-theoretically it seems simpler to introduce a new quantifier $\Pi^{\Box} u{::}(\Psi{\vdash}A_1).\ A_2$, with formation, introduction, and elimination rules as shown below.

$$\begin{array}{ll} \text{Families} & A, B, C ::= \ldots \mid \Pi^{\Box} u{::}(\Psi{\vdash}A_1).\ A_2 \\ \text{Objects} & M, N ::= \ldots \mid \lambda^{\Box} u{::}(\Psi{\vdash}A).\ M \mid M_1 \mathbin{\scriptstyle\Box} M_2 \end{array}$$

$$\dfrac{\Delta \vdash \Psi\ \mathsf{ctx} \quad \Delta; \Psi \vdash A : \mathsf{type} \quad \Delta, u{::}(\Psi{\vdash}A); \Gamma \vdash B : \mathsf{type}}{\Delta; \Gamma \vdash \Pi^{\Box} u{::}(\Psi{\vdash}A).\ B : \mathsf{type}}$$

$$\dfrac{\Delta, u{::}(\Psi{\vdash}A); \Gamma \vdash M : B}{\Delta; \Gamma \vdash \lambda^{\Box} u{::}(\Psi{\vdash}A).\ M : \Pi^{\Box} u{::}(\Psi{\vdash}A).\ B}$$

$$\dfrac{\Delta; \Gamma \vdash N : \Pi^{\Box} u{::}(\Psi{\vdash}A).\ B \quad \Delta; \Psi \vdash M : A}{\Delta; \Gamma \vdash N \mathbin{\scriptstyle\Box} M : [\![M/u]\!]B}$$

The main complication of this extension is that variables u can now be bound and modal substitution must be capture avoiding. Moreover, in an implementation with explicitly named variables we would have to account for the unexpectedly large scope of variables in a declaration $u{::}(\Psi{\vdash}A)$ according to the elimination rule for $\Pi^{\Box}$. Since in the present implementation $\Pi^{\Box}$ is not accessible to the user and we employ de Bruijn indices, this issue has not arisen in Twelf and we leave it to be considered in future work.

To extend the notion of substitution, we note that the context Ψ and the type A in the type family $\Pi^{\Box}u{::}(\Psi{\vdash}A)$ are separate from the context Γ. In other words, elements in the context Ψ may depend on each other, but they may not depend on Γ. Similarly, the type family A, only depends on the context Ψ and not on the context Γ. In accordance with this comment, explicit substitutions need only descend into the body of the $\lambda^{\Box}$. Similarly, in a box-application $N \mathbin{\Box} M$ we do not need to apply the substitution σ to M, since M is well-typed in a different context. The operation of explicit substitution is therefore extended with the new cases as follows.

$$\begin{aligned}
[\sigma](N_1 \mathbin{\Box} N_2) &= ([\sigma]N_1) \mathbin{\Box} N_2 \\
[\sigma](\lambda^{\Box}u{::}(\Psi{\vdash}A).N) &= \lambda^{\Box}u{::}(\Psi{\vdash}A)[\sigma]N
\end{aligned}$$

Next, we extend the modal substitutions. This is straightforward, but now requires a side condition.

$$\begin{aligned}
\llbracket M/u \rrbracket(\lambda^{\Box}v{::}(\Psi{\vdash}A).N) &= \lambda^{\Box}v{::}(\llbracket M/u \rrbracket\Psi{\vdash}\llbracket M/u \rrbracket A).\llbracket M/u \rrbracket N \\
&\qquad\qquad \text{for } u \neq v,\ v \text{ not free in } M \\
\llbracket M/u \rrbracket(N_1 \mathbin{\Box} N_2) &= (\llbracket M/u \rrbracket N_1) \mathbin{\Box} (\llbracket M/u \rrbracket N_2)
\end{aligned}$$

The proofs of the substitution principles stated in Theorem 1 and Theorem 2 readily extend. The extension does not require any changes to the statement of these theorems. However, at present we have not investigated the full theory of LF extended with first-class abstractions over meta-variables.

5 Simultaneous Possibility

In the previous sections we have presented a judgmental formulation of validity relative to a context Ψ and its use to model existential variables. It is also possible to extend the modal possibility to capture the notion of *simultaneous possibility* with respect to a context Ψ. We do not have an an appealing application for simultaneous possibility yet, but we present it here as a dual development to relative validity.

Under the multiple-world interpretation of constructive S4, C *poss* corresponds to C *true* in *some* reachable world. This is given by the following two principles (see [7]), the first of which must be a rule, the second should be an admissible substitution principle.

1. If $\Delta; \Gamma \vdash C$ *true* then $\Delta; \Gamma \vdash C$ *poss*

2. If $\Delta; \Gamma \vdash A\ \mathit{poss}$ and $\Delta; A\ \mathit{true} \vdash C\ \mathit{poss}$ then $\Delta; \Gamma \vdash C\ \mathit{poss}$

In this section, we generalize the judgment for possibility into a judgment for simultaneous possibility, and develop a dependently typed λ-calculus for it. To that end, we introduce the following syntactic categories to the calculus from previous sections.

$$\begin{array}{rl} \text{Families } A, B, C & ::= \ \ldots \mid \Diamond_{\Psi} A \\ \text{Frames } \quad E, F & ::= \langle \sigma, M \rangle \mid \textbf{let dia } x = M \textbf{ in } E \\ \text{Objects } \ M, N & ::= \ldots \mid \textbf{dia } E \end{array}$$

If $\Psi = x_1{:}A_1, \ldots, x_n{:}A_n$, we will write $\Delta; \Gamma \vdash E \div \langle \Psi \rangle A$ to establish that in some reachable world, the types $A_1, \ldots, A_n, A$ can be realized simultaneously, and that the frame E is a witness to that. Of course, because the calculus is dependent, A is allowed to depend on the context Ψ. However, we prohibit that Ψ and A depend on Γ; a crucial invariant in the formulation of the calculus. The new typing rules are given below.

$$\frac{\Delta; \Psi \vdash A : \mathsf{type}}{\Delta; \Gamma \vdash \Diamond_{\Psi} A : \mathsf{type}} \qquad \frac{\Delta; \Gamma \vdash \sigma : \Psi \quad \Delta; \Gamma \vdash M : [\sigma]A}{\Delta; \Gamma \vdash \langle \sigma, M \rangle \div \langle \Psi \rangle A}\ (1)$$

$$\frac{\Delta; \Gamma \vdash M : \Diamond_{\Psi} A \quad \Delta; (\Psi, x{:}A) \vdash E \div \langle \Psi' \rangle B}{\Delta; \Gamma \vdash \textbf{let dia } x = M \textbf{ in } E \div \langle \Psi' \rangle B}\ (2) \qquad \frac{\Delta; \Gamma \vdash E \div \langle \Psi \rangle A}{\Delta; \Gamma \vdash \textbf{dia } E : \Diamond_{\Psi} A}\ (3)$$

Observe that rule (1) generalizes the definitional properties of possibility that we listed at the beginning of the section, and that (2) and (3) are simply introduction and elimination rules for $\Diamond_{\Psi} A$.

The rule (1) establishes that if Ψ and A are simultaneously true, then they certainly are simultaneously possible. A witness for a simultaneous possibility of Ψ and A is a pair consisting of a substitution σ witnessing the truth of Ψ and an object M witnessing the truth of $[\sigma]A$. This justifies a comparison of simultaneous possibility with dependent sums; where dependent sums quantify over a single variable, the simultaneous possibility quantifies over an arbitrary context Ψ. Moreover, dependent sums are not subject to a modal restriction. Rules (2) and (3) form a matched pair of introduction and elimination rules, which is ultimately justified by the frame substitution principle (Theorem 5).

The definitions of explicit and modal substitutions readily extend to frames, as do the corresponding substitution principles. The substitution principles on objects are also easily updated to account for the new object constructor **dia**.

$$\begin{array}{c} [\sigma](\Diamond_{\Psi} A) = \Diamond_{\Psi} A \\ [\sigma](\textbf{dia } E) = \textbf{dia } [\sigma]E \\ [\sigma]\langle \tau, M \rangle = \langle [\sigma]\tau, [\sigma]M \rangle \\ [\sigma](\textbf{let dia } x = M \textbf{ in } E) = \textbf{let dia } x = [\sigma]M \textbf{ in } E \end{array}$$

Notice that in the last clause of the above definition, the substitution σ is not applied to the body E of **let dia**. Indeed, E is defined in a context provided by

M, and hence does not share any variables with σ. This property is also reflected in the following theorem which formulates the explicit substitution principle for frames. Observe that the types in the concluding judgment of the principle are not modified by the substitution σ, which is in contrast to the corresponding principles for objects.

Theorem 3 (Explicit substitutions for frames).

1. *If* $\Delta; \Gamma \vdash \sigma : \Psi$*, and* $\Delta; \Psi \vdash E \div \langle \Psi' \rangle A$*, then* $\Delta; \Gamma \vdash [\sigma]E \div \langle \Psi' \rangle A$*.*
2. $[\sigma]([\tau]E) = [[\sigma]\tau]E$

The operation of modal substitution commutes with all the new constructors.

$$\begin{aligned} [\![M/u]\!](\Diamond_{\Psi} A) &= \Diamond_{([\![M/u]\!]\Psi)} [\![M/u]\!]A \\ [\![M/u]\!](\mathbf{dia}\ E) &= \mathbf{dia}\ [\![M/u]\!]E \\ [\![M/u]\!]\langle \tau, N \rangle &= \langle [\![M/u]\!]\tau, [\![M/u]\!]N \rangle \\ [\![M/u]\!](\mathbf{let\ dia}\ x = N\ \mathbf{in}\ E) &= \mathbf{let\ dia}\ x = [\![M/u]\!]N\ \mathbf{in}\ [\![M/u]\!]E \end{aligned}$$

Theorem 4 (Modal substitutions for frames). *If* $\Delta; \Psi \vdash M : A$ *and* $\Delta, u{::}(\Psi \vdash A), \Delta'; \Gamma \vdash E \div \langle \Psi' \rangle B$*, then* $\Delta, [\![M/u]\!]\Delta'; [\![M/u]\!]\Gamma \vdash [\![M/u]\!]E \div \langle [\![M/u]\!]\Psi' \rangle [\![M/u]\!]B$*.*

The most interesting development, however, is the formulation of the new operation of frame substitution. It is the dependent version of substitution principle (2) from the beginning of this section. Frame substitution is therefore an operation transforming the frame $F \div \langle \Psi \rangle A$ and the frame E such that $\Psi, x{:}A \vdash E \div \langle \Psi' \rangle B$ into a frame $\langle\!\langle F/x \rangle\!\rangle E$ which witnesses the simultaneous possibility $\langle \Psi' \rangle B$. Curiously, the operation is defined by induction on the structure of F rather than E:

$$\begin{aligned} \langle\!\langle \langle \sigma, M \rangle / x \rangle\!\rangle E &= [\sigma, M/x]E \\ \langle\!\langle \mathbf{let\ dia}\ y = M\ \mathbf{in}\ F/x \rangle\!\rangle E &= \mathbf{let\ dia}\ y = M\ \mathbf{in}\ \langle\!\langle F/x \rangle\!\rangle E \end{aligned}$$

Lastly, we need a theorem to establish the soundness of the operation, that is, the fact that the frame $\langle\!\langle F/x \rangle\!\rangle E$ indeed is a witness for the required simultaneous possibility.

Theorem 5 (Frame substitution principle). *If* $\Delta; \Gamma \vdash F \div \langle \Psi \rangle A$*, and* $\Delta; (\Psi, x{:}A) \vdash E \div \langle \Psi' \rangle B$*, then* $\Delta; \Gamma \vdash \langle\!\langle F/x \rangle\!\rangle E \div \langle \Psi' \rangle B$*.*

6 Conclusion

We have presented an abstract view of meta-variables based on a fragment of the constructive modal logic of necessity and possibility. The judgmental formulation of this logic distinguishes between modal and ordinary variable contexts, and we used the modal variables to model the meta-variables in logical frameworks. We extended the conventional validity concept to relative validity which allowed us to explain several efficient implementation techniques for unification and related operations in the higher-order setting. We also developed a calculus involving simultaneous possibility, which is logically dual to relative validity, although currently without clear application.

The core dependent calculus with modal variables admits well-behaved notions of equality and canonical forms and type-checking remains decidable [9]. We conjecture that these results can be extended to encompass $\Pi^{\Box}$ and $\Diamond$.

References

1. Gilles Dowek, Thérèse Hardin, and Claude Kirchner. Higher-order unification via explicit substitutions. In D. Kozen, editor, *Proceedings of the Tenth Annual Symposium on Logic in Computer Science*, pages 366–374, San Diego, California, June 1995. IEEE Computer Society Press.
2. Gilles Dowek, Thérèse Hardin, Claude Kirchner, and Frank Pfenning. Unification via explicit substitutions: The case of higher-order patterns. In M. Maher, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 259–273, Bonn, Germany, September 1996. MIT Press.
3. Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.
4. Gérard Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.
5. Per Martin-Löf. On the meanings of the logical constants and the justifications of the logical laws. *Nordic Journal of Philosophical Logic*, 1(1):11–60, 1996.
6. Dale Miller. Unification under a mixed prefix. *Journal of Symbolic Computation*, 14:321–358, 1992.
7. Frank Pfenning and Rowan Davies. A judgmental reconstruction of modal logic. *Mathematical Structures in Computer Science*, 11(4):511–540, 2001.
8. Brigitte Pientka. Higher-order substitution tree indexing. In C. Palamidessi, editor, *19th International Conference on Logic Programming, Mumbai, India*, Lecture Notes in Computer Science (LNCS), to appear. Springer-Verlag, 2003.
9. Brigitte Pientka. *Tabled higher-order logic programming.* PhD thesis, Department of Computer Sciences, Carnegie Mellon University, 2003. forthcoming.
10. Brigitte Pientka and Frank Pfennning. Optimizing higher-order pattern unification. In F. Baader, editor, *19th International Conference on Automated Deduction, Miami, USA*, Lecture Notes in Computer Science (LNAI 2741), pages 473–487. Springer, 2003.