Calculi of Meta-variables

Masahiko Sato[1], Takafumi Sakurai[2], Yukiyoshi Kameyama[3], and Atsushi Igarashi[1]

[1] Graduate School of Informatics, Kyoto University
{masahiko,igarashi}@kuis.kyoto-u.ac.jp
[2] Department of Mathematics and Informatics, Chiba University
sakurai@math.s.chiba-u.ac.jp
[3] Institute of Information Sciences and Electronics, University of Tsukuba, and JST
kam@is.tsukuba.ac.jp

Abstract. The notion of meta-variable plays a fundamental role when we define formal systems such as logical and computational calculi. Yet it has been usually understood only informally as is seen in most textbooks of logic. Based on our observations of the usages of meta-variables in textbooks, we propose two formal systems that have the notion of meta-variable.

In both calculi, each variable is given a level (non-negative integer), which classifies variables into object variables (level 0), meta-variables (level 1), metameta-variables (level 2) and so on. Then, simple arity systems are used to exclude meaningless terms like a meta-level function operating on the metameta-level. A main difference of the two calculi lies in the definitions of substitution. The first calculus uses *textual* substitution, which can often be found in definitions of quantified formulae: when a term is substituted for a meta-variable, free object-level variables in the term may be captured. The second calculus is based on the observation that predicates can be regarded as meta-level functions on object-level terms, hence uses *capture-avoiding* substitution.

We show both calculi enjoy a number of properties including Church-Rosser and Strong Normalization, which are indispensable when we use them as frameworks to define logical systems.

Keywords: meta-variable, logical framework, context, λ-calculus

1 Introduction

The notion of *meta-variable* is a fundamental notion both in logic and computer science. It is because both logic and computer science mainly deal with linguistic objects such as formulas, proofs, programs etc., and whenever we make a *general* statement about these objects we use meta-variables to refer to these objects. For example, if we look at any book of logic, most variables we see are meta-variables. Meta-variables are also known as *metamathematical variables* [7] and *syntactical variables* [12]. However, it seems that, so far, only a very few attempts have been made to formalize the notion of meta-variable. One reason for this may be that we have to go to metameta-level to do so.

In this paper, we present two new formalizations of the concept of the meta-variable. These formalizations are based on our observations of the usages of meta-variables in text books and technical papers on logic.

The first observation is from Shoenfield [12] and Kleene [7]. In these books, we find the following sentence as one of the inductive clauses which define formulas. (We have slightly modified notations from the originals.)

If x is a variable and A is a formula, then $\exists x\ A$ is a formula.

In the above sentence, both 'x' and 'A' are meta-variables, and when we use the sentence as a rule which is used to construct a concrete formula, we must *instantiate* these meta-variables by concrete linguistic objects of the object language. Thus, for example, we may instantiate x by a concrete variable $\mathtt{x}$ and A by a concrete formula $\mathtt{x} = \mathtt{x}$. Then, by applying the instantiated rule, we have that $\exists\mathtt{x}\ \mathtt{x} = \mathtt{x}$ is a formula. Here, it is important to remark that the process of instantiation we just described is a form of *substitution*, but, unlike ordinary substitutions, the variables being substituted are meta-variables. There is another subtle point in this substitution process. To see this, we analyze the above instantiation process in two steps. Namely, we assume that the meta-variable 'x' is first instantiated and then, in the second step, the meta-variable 'A' is instantiated. Then, after the first step, we get the following sentence.

If $\mathtt{x}$ is a variable and A is a formula, then $\exists\mathtt{x}\ A$ is a formula.

In the second step, we substitute $\mathtt{x} = \mathtt{x}$ for A in the above sentence, and we get the fully instantiated sentence:

If $\mathtt{x}$ is a variable and $\mathtt{x} = \mathtt{x}$ is a formula, then $\exists\mathtt{x}\ \mathtt{x} = \mathtt{x}$ is a formula.

We note here that, unlike ordinary substitution, we have substituted $\mathtt{x} = \mathtt{x}$ for A in $\exists\mathtt{x}\ A$ without following the usual convention of renaming the name of the binding variable ($\mathtt{x}$ in this case) to avoid the capture of free variables.

The second observation we now make is also a very common one. Often in the literature, notation like $A(x)$ is used to indicate a formula where free occurrences of x in $A(x)$ is implicitly understood. Thus, if t is a term, then $A(t)$ stands for a formula which is obtained from $A(x)$ by substituting t for x in $A(x)$. In this usage, 'x', 't' and 'A' are all meta-variables and the first two meta-variables range over variables and terms in the object language. As for the third meta-variable 'A' it is possible to interpret its range in two ways.

The first interpretation is to regard 'A' as ranging over functions which, when applied to terms, will yield formulas. In this interpretation $A(t)$ denotes the result of applying the meta-level function A to a term t. So, in this interpretation, A is a *metameta*-variable, since its denotation is not a linguistic object of the object language but it is a function in the meta language.

The second interpretation is to regard 'A' as ranging over *abstracts* of the object language which can be instantiated to formulas by supplying terms of the object language. This interpretation is possible only if the object language contains such abstracts as its formal entities. Higher-order abstract syntax employed by, e.g., Edinburgh LF [5] is based on this interpretation.

In this paper, we will introduce two typed calculi $\lambda\mathcal{M}$ and λm, which are respectively designed based on the above two observations. In $\lambda\mathcal{M}$ and λm, with each variable a non-negative integer, which we call the level of the variable, is associated. We consider level 0 as the object-level, level 1 as the meta-level, level 2 as the metameta-level and so on.

In these formalizations, we believe that we can take virtually any formal system as the object-level of our systems. However, for the sake of concrete presentation, we take as the object-level a system of symbolic expressions we introduced in Sato [10] which is simple but powerful enough to represent syntax of many of the commonly used formal systems such as the λ-calculus and predicate calculus. Both $\lambda\mathcal{M}$ and λm will be constructed on top of this object-level system by adding higher-level structures, and we will show that these calculi enjoy nice properties such as confluence and strong normalizability. We will also show that the calculus $\lambda\mathcal{M}$ can represent the notion of context naturally since a context, which is an expression containing some holes in it, is inherently a linguistic object in the meta-level and not in the object-level.

Due to lack of space, we have omitted some lemmas and details of proofs. A full version of this paper with proofs is accessible at

`http://www.sato.kuis.kyoto-u.ac.jp/~masahiko/index-e.html`.

2 Informal Introduction to the Calculi

In this section we informally explain the two calculi $\lambda\mathcal{M}$ and λm which we propose in this paper. We assume our object language contains constants, abstraction ($\texttt{(}x\texttt{)}\,\texttt{[}M\texttt{]}$), and pair ($\langle M, N\rangle$). Abstraction and application in the meta-level are denoted by $\lambda X.M$ and MN. (We often use capital letters for meta-variables in examples, although both object- and meta-level variables belong to the same syntactic category in the formal definition.)

2.1 The Calculus $\lambda\mathcal{M}$

The first calculus $\lambda\mathcal{M}$ is based on the first observation in Section 1. Let us consider the first observation again and assume that we have just completed the first step. Then, we have the expression $\exists \mathtt{x}\ A$. We can represent this expression by $\langle '\exists', \texttt{(}x^0\texttt{)}\,\texttt{[}A^1\texttt{]}\rangle$ using a constant $'\exists'$. On the shoulder of a variable, we write a natural number to indicate its level, although we often omit the level if it is clear from the context. So, we simply write x for x^0 and it corresponds to the concrete object-level variable $\mathtt{x}$. In $\lambda\mathcal{M}$, the instantiation process of the meta-variable A by the object-level formula $x = x$ can be represented as the reduction process of the following β-redex:

$$(\lambda A.\langle '\exists', \texttt{(}x\texttt{)}\,\texttt{[}A\texttt{]}\rangle)\langle '=', \langle x, x\rangle\rangle.$$

In the reduction, as pointed out in Section 1, *non-standard* substitution is performed and we get:

$$\langle '\exists', \text{(}x\text{)}\,\text{[}\langle '=', \langle x, x\rangle\rangle\text{]}\rangle$$

which represents the formula $\exists\mathtt{x}\ \mathtt{x} = \mathtt{x}$ as expected. Note that the object-level variable x is captured through the substitution.

The non-standard (textual) substitution we have just introduced gives rise to the following two technical problems.

The first one is the non-confluence of the calculus. As argued in the literature on context calculi [8, 6, 11], calculi that have textual substitution cannot be confluent unless we restrict the evaluation order. For instance, let M be the term $(\lambda X^2.\,(\lambda x^1.\,X^2)y^0)x^1$. Depending on the evaluation-order, we will get different results y^0 and x^1. Our solution to this problem is (roughly) that, a redex may not be reduced if it contains variables of higher levels than the level of the redex[1]. In this example, the inner redex has level-1, so its reduction is postponed until the variable X^2 disappears.

The second problem in formulating $\lambda\mathcal{M}$ is that, since we restrict the evaluation-order, some reductions may get stuck. Consider the terms $(\lambda X^2.\,\lambda x^1.\,X^2)y^0z^0$, and $(\lambda x^1.\,\lambda X^2.\,X^2)y^0z^0$. The first term reduces to y^0, while the second term cannot be reduced. Since we do not consider terms like the second one meaningful, we introduce *arities* to rule out such terms. For instance, $\lambda x^1.\,X^2 : 0 \to_1 0$ signifies that this term denotes a level-1 function from objects to objects. Similarly we have $\lambda X^2.\,\lambda x^1.\,X^2 : 0 \to_2 (0 \to_1 0)$. On the other hand, $\lambda x^1.\,\lambda X^2.\,X^2$ would have arity $0 \to_1 (0 \to_2 0)$, and it would denote a level-1 function which returns a level-2 function. We will exclude such a term by defining arity properly, and show that the evaluation in $\lambda\mathcal{M}$ does not get stuck (Theorem 4).

Although $\lambda\mathcal{M}$ has non-standard substitution, we need the standard capture-avoiding substitution as well, when the variable being substituted for and one being captured are of the same level. Let us see the following reduction:

$$(\lambda X^2.\,\lambda Y^2.\,\lambda z^1.\,X^2)(Y^2z^1) \to \lambda W^2.\,\lambda z^1.\,Y^2z^1$$

in which the variable z^1 is captured, while the variable Y^2 is not captured since its level is the same as that of the variable X^2.

2.2 The Calculus λm

The second calculus λm formalizes the first interpretation[2] of the second observation in Section 1.

The formula $A(t)$ can be represented as $A^2(t^1)$ using the level-2 variable A^2 of arity $0 \to_1 0$, and the level-1 variable t of arity 0 in λm. The existential formula $\exists x\ A(x)$ is represented as:

$$\langle '\exists', \text{(}x^0\text{)}\,\text{[}A^2(x^0)\text{]}\rangle,$$

[1] The level of the redex $(\lambda X^i.\,M)N$ is i.

[2] In this paper, we do not formalize the second interpretation in which A in $A(t)$ ranges over abstracts of the object language. It should be a straightforward extension of this work, but details are left for future work.

and the substitution of $\lambda x^1 : 0.\, \langle '=', \langle x^1, x^1 \rangle \rangle$ for A^2 is realized by β-reduction, but this time we do *not* use the non-standard substitution. Hence, the term:

$$(\lambda A^2 : 0 \to_1 0.\, \langle '\exists', (x^0)\, [A^2(x^0)] \rangle)(\lambda x^1 : 0.\, \langle '=', \langle x^1, x^1 \rangle \rangle)$$

reduces (using *standard* substitution) to

$$\langle '\exists', (x^0)\, [\langle '=', \langle x^0, x^0 \rangle \rangle] \rangle$$

as expected.

3 The Calculus $\lambda\mathcal{M}$

In this section, we give a formal definition of the first calculus $\lambda\mathcal{M}$. The second calculus λm will be introduced in the next section.

3.1 Arities and Terms

We define *arity* (α) and its *level* ($|\alpha|$) as follows:

1. 0 is an arity and $|0| = 0$.
2. If α and β are arities, $0 < i$, $|\alpha| < i$, and $|\beta| \leq i$, then $\alpha \to_i \beta$ is an arity and $|\alpha \to_i \beta| = i$.

Note that the side condition of the second clause reflects the intended notion of level introduced in Section 1. Intuitively, the arity $\alpha \to_i \beta$ is for level-i functions from terms of arity α to terms of arity β, thus $|\alpha| < i$ and $|\beta| < i$ must be satisfied. The restriction on β is relaxed to $|\beta| \leq i$ to allow currying.

We assume that, for each natural number i and arity α, there are infinitely many variables, and the sets of variables of each level and arity are mutually disjoint. For a variable x of level i and arity α, we sometimes write it as x^i. The set of all variables is denoted by V. A *(variable) declaration* is an expression of the form $x^i : \alpha$, where α is an arity and either $|\alpha| < i$ or $|\alpha| = i = 0$. We say the *level* of this declaration is i. A *hypothesis sequence* is a finite sequence of declarations. A *judgment* is an expression of the form $\Gamma \vdash M : \alpha$ where Γ is a hypothesis sequence and α is an arity.

We have the following rules that are used to derive judgments.

The first rule introduces variables for each i and α, where we assume x is a variable of level i and arity α.

$$\frac{x^i : \alpha \in \Gamma}{\Gamma \vdash x^i : \alpha}\ (\mathsf{var})$$

The next two rules introduce abstraction and application for level-i ($i > 0$).

$$\frac{\Gamma, x^i : \alpha \vdash M : \beta \quad |\beta| \leq i}{\Gamma \vdash \lambda x^i{:}\alpha.\, M : \alpha \to_i \beta}\ (\mathsf{abs}) \qquad \frac{\Gamma \vdash M : \alpha \to_i \beta \quad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta}\ (\mathsf{app})$$

Note that, in the rule (abs), the level of the variable x^i and that of the arity $\alpha \to_i \beta$ should agree, and the side-condition $|\beta| \leq i$ is needed to form a (well-formed) arity $\alpha \to_i \beta$. Note also that we may not construct a term like $\lambda x^1 : 0.\, \lambda X^2 : 0.\, X^2$.

The last group of rules are those for the level-0, the object language.

$$\frac{c \text{ is a constant}}{\Gamma \vdash c : 0}\ (\mathsf{const0}) \qquad \frac{\Gamma, x^0 : 0 \vdash M : 0}{\Gamma \vdash (x^0)\,[M] : 0}\ (\mathsf{abs0})$$

$$\frac{\Gamma \vdash M : 0 \quad \Gamma \vdash N : 0}{\Gamma \vdash \langle M, N \rangle : 0}\ (\mathsf{pair0})$$

An expression M is said to be a *term* if a judgment of the form $\Gamma \vdash M : \alpha$ is derivable for some Γ and α. We sometimes just write $\lambda x^i.\, M$ for $\lambda x^i{:}\alpha.\, M$ when the arity α of x^i is irrelevant.

The scope of $\lambda x.$ and free occurrences of variables in a term are defined as usual. For a term M, the set of free variables in M is denoted by $\mathrm{FV}(M)$. The *level of a term* M, denoted by $|M|$, is the maximum level of variables in M, or 0 if there is no variable in M. Note that we take all variables (even variable occurrences in the scope of λ) into account—for instance, $|(\lambda x^2{:}\alpha.\, x^2) y^1| = 2$. Note also that $|M|$ is not necessarily equal to the level of its arity. The *level of a hypothesis sequence* Γ, denoted by $|\Gamma|$, is the maximum level of variables in Γ, or 0 if Γ is the empty sequence.

3.2 α-equivalence

In the calculus $\lambda\mathcal{M}$, we need special care to define α-equivalence: occurrences of a variable x in the scope of $\lambda x.$ – usually called bound occurrences – may or may not be subject to renaming since textual substitution does not commute with naive variable renaming. For instance, we may identify $\lambda x^1.\, y^2(\lambda z^1.\, x^1 z^1)$ with $\lambda x^1.\, y^2(\lambda u^1.\, x^1 u^1)$, but not with $\lambda w^1.\, y^2(\lambda z^1.\, w^1 z^1)$. To see its reason, let us substitute x^1 for y^2 in these terms. Since the level of y^2 is higher than x^1 and w^1, the textual substitution is used, and the first and the third terms become $\lambda x^1.\, x^1(\lambda z^1.\, x^1 z^1)$, and $\lambda w^1.\, x^1(\lambda z^1.\, w^1 z^1)$, resp., which do not have the same denotational meaning. Hence, we let an abstraction $\lambda x^i.\, e$ be α-convertible only when no variable at a level higher than i occurs in its scope.

We define the α-equivalence after a few auxiliary definitions. A *renaming* of variables is a partial function from V to V which is injective and satisfies $|f(x^i)| = i$ for all x^i in the domain of the partial function. For a partial function f, its domain is denoted by $\mathrm{dom}(f)$. For a renaming f and a variable x^i (which may not be in $\mathrm{dom}(f)$), $f \downarrow x^i$ is a renaming of variables such that $\mathrm{dom}(f \downarrow x^i) = \mathrm{dom}(f) - \{x^i\}$, and $f \downarrow x^i$ agrees with f on its domain.

For a renaming f and terms M and N in $\lambda\mathcal{M}$, we derive a judgment of the form $f \vdash M \simeq N$ by the following inference rules:

$$\frac{f(x^i) = y^i}{f \vdash x^i \simeq y^i} \qquad \frac{f \vdash M \simeq M' \quad f \vdash N \simeq N'}{f \vdash MN \simeq M'N'}$$

$$\frac{f \vdash M \simeq M'}{f \vdash \lambda x^i : \alpha.\, M \simeq \lambda x^i : \alpha.\, M'} \; (f(x^i) = x^i)$$

$$\frac{f \vdash M \simeq M'}{g \vdash \lambda x^i : \alpha.\, M \simeq \lambda y^i.\, \alpha.M'} \left(\begin{array}{c} |M| \leq i,\, |M'| \leq i, \\ f(x^i) = y^i,\, g \downarrow x^i = f \downarrow x^i \end{array} \right)$$

The last rule can be applied only when both M and M' are of level less than or equal to i. Otherwise, the term $\lambda x^i.\, M$ contains (not necessarily free) occurrences of meta-variables which have higher levels than i, and we cannot rename the bound variable x^i. In this case we can still apply the second last rule, since it does not rename the bound variable x^i. For brevity, we omit the inference rules for terms constructed by the rules (const0), (abs0), and (pair0), which are similarly defined. For instance, a term $\texttt{(}x^0\texttt{)[}M\texttt{]}$ has the same rules as the term $\lambda x^i : \alpha.\, M$.

Let `id` be the identity function on V. If $\texttt{id} \vdash M \simeq N$ is derived by the rules above, we say M *is α-equivalent to* N (written by $M \equiv_\alpha N$). It is easy to show the relation $\equiv_\alpha$ is a congruence on terms.

3.3 Substitution and Reduction

The notion of reduction in the calculus $\lambda\mathcal{M}$ is the union of those in the object language (which we do not specify) and the following β-reduction:

$(\boldsymbol{\beta})$ $(\lambda x^i.\, M)N \to [x^i := N]M \qquad$ if $|M| \leq i$ and $|N| \leq i$

in which $[x^i := N]M$ denotes the (non-standard) substitution defined below. We write $\stackrel{*}{\to}$ for the reflexive and transitive closure of $\to$.

For a level $i > 0$, a level-i variable x^i, and terms M and N such that $|M| \leq i$ and $|N| \leq i$, we define $[x^i := N]M$ as follows:

1. $[x^i := N]x^i \stackrel{\triangle}{=} N$
2. $[x^i := N]y^j \stackrel{\triangle}{=} y^j \qquad$ if $y^j \not\equiv x^i$
3. $[x^i := N](M_1 M_2) \stackrel{\triangle}{=} ([x^i := N]M_1)([x^i := N]M_2)$
4. $[x^i := N](\lambda y^j.\, M) \stackrel{\triangle}{=} \lambda y^j.\, [x^i := N]M \quad$ if $j < i$
5. $[x^i := N](\lambda y^i.\, M) \stackrel{\triangle}{=} \lambda y^i.\, [x^i := N]M \quad$ if $y^i \notin \mathrm{FV}(N)$ and $x^i \not\equiv y^i$
6. $[x^i := N]c \stackrel{\triangle}{=} c$
7. $[x^i := N]\texttt{(}y^0\texttt{)[}M\texttt{]} \stackrel{\triangle}{=} \texttt{(}y^0\texttt{)[}[x^i := N]M\texttt{]}$
8. $[x^i := N]\langle M_1, M_2\rangle \stackrel{\triangle}{=} \langle [x^i := N]M_1, [x^i := N]M_2\rangle$

The first three clauses are standard. For the fourth line, if the level of y^j is strictly less than that of x^i, the substitution behaves like textual replacement, that is, free variables may get captured through this substitution. For the fifth line, the bound variable y^i has the same level as x^i that is being substituted, in which case the substitution is the standard capture-avoiding one, hence the side-conditions $y^i \notin \mathrm{FV}(N)$ and $x^i \not\equiv y^i$ must be satisfied. (The second side-condition $x \not\equiv y$ is not needed for the fourth clause because variables at different levels are assumed to be different.) This side-condition can be always satisfied by taking an α-equivalent term. The last three clauses deal with the level 0 in a straightforward manner, except textual substitution.

For brevity, we identify α-equivalent terms in the following. Under this convention, we simply have that $[x^i := N]M$ is defined if and only if $\max(|M|, |N|) \leq i$.

The object-level may contain other redices than the β-redices in the above form, and to distinguish these two, we say that a redex $(\lambda x^i : \alpha.\, M)N$ is a *meta-redex*. A term is *meta-normal* if it does not have meta-redices and a reduction is called a *meta-reduction* if its redex is a meta-redex.

3.4 Replacing Level-0 Languages

Our level-0 language here is a simplest possible language, which has no notion of types or even computation. We have adopted such a language because it is simple but expressible enough to represent expressions that appear in typical logical systems. It would be possible, however, to adopt other languages such as untyped and simply typed λ-calculi as the level-0 language.

For example, for untyped λ-calculus, we could introduce another term constructor $M_1 \cdot M_2$ by the rule

$$\frac{\Gamma \vdash M : 0 \quad \Gamma \vdash N : 0}{\Gamma \vdash M \cdot N : 0}\ (\mathsf{app0})$$

and a reduction rule $(x^0)\texttt{[}M\texttt{]} \cdot N \to [x^0 := N]M$ (if $|M| = |N| = 0$) where the level-0 substitution would be defined as expected. It would be also straightforward to substitute the simply typed λ-calculus for the level-0 language, by extending the base arities from 0 to the set of simple types.

4 The Calculus λm

As discussed in the previous sections, the second calculus λm is based on the second observation discussed in the introduction. It is obtained from $\lambda\mathcal{M}$ by replacing the definition of substitution with the standard capture-avoiding one. Also, we need to use the standard definition of α-equivalence to identify $\lambda x^1.y^2(\lambda z^1.x^1 z^1)$ with $\lambda w^1.y^2(\lambda z^1.w^1 z^1)$, which are not α-equivalent in $\lambda\mathcal{M}$. Since the other definitions of arities, rules to derive judgments, β-reduction remain the same, we avoid repeating definitions and just show changes to be made.

For α-equivalence, we replace the third and fourth rules to derive $f \vdash M \simeq N$ with the following one:

$$\frac{f \vdash M \simeq M'}{g \vdash \lambda x^i : \alpha.M \simeq \lambda y^i : \alpha.M'}\ (f(x^i) = y^i,\ g \downarrow x^i = f \downarrow x^i)$$

Notice that there is no restriction on the levels of the bodies M and M' of λ-abstraction. Similarly, the definition of substitution will be as follows.

For a level $i > 0$, a level-i variable x^i, and terms M and N such that $|M| \leq i$ and $|N| \leq i$, $[x^i := N]M$ is defined by:

1. $[x^i := N]x^i \triangleq N$
2. $[x^i := N]y^j \triangleq y^j \quad$ if $y^j \not\equiv x^i$
3. $[x^i := N](M_1 M_2) \triangleq ([x^i := N]M_1)([x^i := N]M_2)$
4. $[x^i := N](\lambda y^j.\, M) \triangleq \lambda y^j.\, [x^i := N]M \quad$ if $y^j \notin \mathrm{FV}(N)$ and $x^i \not\equiv y^j$
5. $[x^i := N]c \triangleq c$
6. $[x^i := N]\texttt{(}y^0\texttt{)}\,\texttt{[}M\texttt{]} \triangleq \texttt{(}y^0\texttt{)}\,\texttt{[}[x^i := N]M\texttt{]} \quad$ if $y^0 \notin \mathrm{FV}(N)$
7. $[x^i := N]\langle M_1, M_2\rangle \triangleq \langle [x^i := N]M_1, [x^i := N]M_2\rangle$

Notice that the first three clauses are the same as before and the fourth clause is now a familiar one that avoids variable capturing.

5 Examples

In this section, we show a few examples of $\lambda\mathcal{M}$ and λm.

5.1 Representing Formulas

In earlier sections, we informally explained how the formula $\mathtt{x} = \mathtt{x}$ is substituted for the meta-variable A in $\exists\mathtt{x}\ A$. Here we consider this process formally in $\lambda\mathcal{M}$.

The arity of the corresponding term is inferred in $\lambda\mathcal{M}$ as follows:

$$\dfrac{\dfrac{\dfrac{\overline{x^0 : 0, A^1 : 0 \vdash {}'\exists' : 0} \quad \dfrac{\overline{x^0 : 0, A^1 : 0, x^0 : 0 \vdash A^1 : 0}}{x^0 : 0, A^1 : 0 \vdash \texttt{(}x^0\texttt{)}\,\texttt{[}A^1\texttt{]} : 0}}{x^0 : 0, A^1 : 0 \vdash \langle {}'\exists', \texttt{(}x^0\texttt{)}\,\texttt{[}A^1\texttt{]}\rangle : 0}}{x^0 : 0 \vdash \lambda A^1 : 0.\, \langle {}'\exists', \texttt{(}x^0\texttt{)}\,\texttt{[}A^1\texttt{]}\rangle : 0 \to_1 0} \qquad \begin{array}{c}\vdots\\ x^0 : 0 \vdash \langle {}'=', \langle x^0, x^0\rangle\rangle : 0\end{array}}{x^0 : 0 \vdash (\lambda A^1 : 0.\, \langle {}'\exists', \texttt{(}x^0\texttt{)}\,\texttt{[}A^1\texttt{]}\rangle)(\langle {}'=', \langle x^0, x^0\rangle\rangle) : 0}$$

Note that, the variable x^0 occurs free in the term of the conclusion, as indicated by the hypothesis sequence.

We can compute this term as:

$$\begin{aligned}
&(\lambda A^1 : 0.\, \langle {}'\exists', \texttt{(}x^0\texttt{)}\,\texttt{[}A^1\texttt{]}\rangle)(\langle {}'=', \langle x^0, x^0\rangle\rangle)\\
\to\ &[A^1 := \langle {}'=', \langle x^0, x^0\rangle\rangle]\langle {}'\exists', \texttt{(}x^0\texttt{)}\,\texttt{[}A^1\texttt{]}\rangle\\
\equiv\ &\langle {}'\exists', [A^1 := \langle {}'=', \langle x^0, x^0\rangle\rangle]\texttt{(}x^0\texttt{)}\,\texttt{[}A^1\texttt{]}\rangle\\
\equiv\ &\langle {}'\exists', \texttt{(}x^0\texttt{)}\,\texttt{[}\langle {}'=', \langle x^0, x^0\rangle\rangle\texttt{]}\rangle
\end{aligned}$$

Note that non-standard (textual) substitution is applied in the last step, since the level of A^1 is higher than that of x^0. As a result, the free occurrences of x^0 get bound, and x^0 does not occur free in the resulting term, which represents the formula $\exists\mathtt{x}\ \mathtt{x} = \mathtt{x}$.

We also informally explained how the same example can be written in λm. Its formal counterpart can be written as follows[3]:

$$
\dfrac{
  \dfrac{
    \dfrac{}{A^2 : 00 \vdash {}'\exists' : 0}
    \qquad
    \dfrac{
      \dfrac{
        \dfrac{}{\Gamma \vdash A^2 : 00}
        \quad
        \dfrac{}{\Gamma \vdash x^0 : 0}
      }{\Gamma \vdash A^2(x^0) : 0}
    }{A^2 : 00 \vdash \mathtt{(}x^0\mathtt{)}\,\mathtt{[}A^2(x^0)\mathtt{]} : 0}
  }{
    \dfrac{A^2 : 00 \vdash \langle '\exists', \mathtt{(}x^0\mathtt{)}\,\mathtt{[}A^2(x^0)\mathtt{]}\rangle : 0}{\vdash \lambda A^2 : 00.\, \langle '\exists', \mathtt{(}x^0\mathtt{)}\,\mathtt{[}A^2(x^0)\mathtt{]}\rangle : 00 \rightarrow_2 0}
  }
  \qquad
  \dfrac{
    \begin{array}{c}\vdots\\ y^1 : 0 \vdash \langle '=', \langle y^1, y^1\rangle\rangle : 0\end{array}
  }{\vdash \lambda y^1 : 0.\, \langle '=', \langle y^1, y^1\rangle\rangle : 00}
}{\vdash (\lambda A^2 : 00.\, \langle '\exists', \mathtt{(}x^0\mathtt{)}\,\mathtt{[}A^2(x^0)\mathtt{]}\rangle)(\lambda y^1 : 0.\, \langle '=', \langle y^1, y^1\rangle\rangle) : 0}
$$

where we put an arity $00 = 0 \rightarrow_1 0$ and $\Gamma = A^2 : 0 \rightarrow_1 0, x^0 : 0$ Note that we replaced the meta-variable A^1 in $\lambda\mathcal{M}$ by an application term $A(x)$, and to ensure this application is resolved in a meta-level, the arity of A should be $0 \rightarrow_1 0$, hence the level of the variable A must be 2 (or higher).

We omit the computation of the term here, since it is essentially the same as the standard β-reduction.

5.2 Representing Contexts

Contexts can be represented in $\lambda\mathcal{M}$ using meta-variables naturally. Let M be a term, C be a context in the object language, i.e., a term with a hole $[\,]$ in it, and $C[M]$ be the result of the hole-filling operation. In $\lambda\mathcal{M}$, the context C is represented as a term $\overline{C} \equiv (\lambda X^1{:}0.\, C[X^1])$, and $C[M]$ as an application $(\overline{C}M)$, which reduces to $C[M]$ by the textual substitution in $\lambda\mathcal{M}$.

Let us take an example from Hashimoto and Ohori's context calculus [6]. Consider the context $C \equiv (\lambda u.\, (\lambda x.\, [\,])u + y)3$ and the term $M \equiv (\lambda z.\, C[x+z])x$ in lambda calculus. They can be written in $\lambda\mathcal{M}$ (using our notation for object language) as:

$$
\begin{aligned}
\overline{C} &\equiv \lambda X^1{:}0.\, \mathtt{(}u^0\mathtt{)}\,\mathtt{[}(\mathtt{(}x^0\mathtt{)}\,\mathtt{[}X^1\mathtt{]} \cdot u^0) + y^0\mathtt{]} \cdot 3\\
\overline{M} &\equiv \mathtt{(}z^0\mathtt{)}\,\mathtt{[}\overline{C}\ (x^0 + z^0)\mathtt{]} \cdot x^0
\end{aligned}
$$

We can reduce $\overline{M}$ as:

$$
\begin{aligned}
\overline{M} &\rightarrow \mathtt{(}z^0\mathtt{)}\,\mathtt{[}[X^1 := x^0 + z^0](\mathtt{(}u^0\mathtt{)}\,\mathtt{[}(\mathtt{(}x^0\mathtt{)}\,\mathtt{[}X^1\mathtt{]} \cdot u^0) + y^0\mathtt{]} \cdot 3)\mathtt{]} \cdot x^0\\
&\equiv \mathtt{(}z^0\mathtt{)}\,\mathtt{[}\mathtt{(}u^0\mathtt{)}\,\mathtt{[}(\mathtt{(}x^0\mathtt{)}\,\mathtt{[}x^0 + z^0\mathtt{]} \cdot u^0) + y^0\mathtt{]} \cdot 3\mathtt{]} \cdot x^0\\
&\overset{*}{\rightarrow} 3 + x^0 + y^0
\end{aligned}
$$

Note that, by the side-condition of the β-reduction, we cannot reduce the outermost level-0 redex first. If it would be reduced first, we would get $3+x^0+x^0$ as a result, so the Church-Rosser property would be broken. Hashimoto and Ohori's context calculus has a similar restriction that the β-reduction is prohibited when the redex contains a hole.

[3] We changed the level-1 variable x^1 to y^1 for readability. Formally this renaming is justified by the α-equivalence.

A good point of our representation of contexts is that, since contexts are functions, they can be *composed*, in other words, we can fill a context in another context. As a simple example, let C and D be the contexts $\lambda x.[\,]$ and $\lambda y.[\,]$ in lambda calculus, and consider hole-filling of D in C, i.e. $C[D]$. The contexts C and D are represented in $\lambda\mathcal{M}$ as $\overline{C} \equiv \lambda X^1.\,(x^0)\,[X^1]$ and $\overline{D} \equiv \lambda X^1.\,(y^0)\,[X^1]$, then we can compose them in the same way as composition of two functions:

$$\overline{C} \circ \overline{D} \equiv \lambda X^1.\overline{C}\;(\overline{D}\;X^1)$$

$\overline{C} \circ \overline{D}$ reduces to $\lambda X^1.\,(x^0)\,[(y^0)\,[X^1]]$ which represents the context $\lambda x.\,\lambda y.[\,]$. It should be noted that, in several existing context calculi including Hashimoto-Ohori's and our previous work [11], contexts cannot be composed, since these calculi keep track of possible bound variables in a hole.

6 Properties of $\lambda\mathcal{M}$ and λm

We have a number of desirable properties for the calculi $\lambda\mathcal{M}$ and λm, that is, they enjoy subject reduction, confluence, and strong normalization properties. In the following, we focus on the properties of $\lambda\mathcal{M}$, but the modification for λm is straightforward.

We can prove the subject reduction property in the standard way.

Theorem 1 (Subject Reduction Property). *If $\Gamma \vdash M : \alpha$ is derived and $M \xrightarrow{*} N$, then $\Gamma \vdash N : \alpha$ can be derived.*

Note that even if $\Gamma \vdash (\lambda x^i.\,M)N : \alpha$ is derived, $[x^i := N]M$ is not necessarily defined. But we can prove from Theorem 2 and 3 that if $|\Gamma| \leq i$ then there exist terms M' and N' such that $M \xrightarrow{*} M'$, $N \xrightarrow{*} N'$, and $[x^i := N']M'$ is defined.

Lemma 1. *Suppose $\Gamma \vdash M : \alpha$ is derived and the highest level of the redices of M is i. Then, any reduction sequence that reduces only level-i redices leads to a term that does not have redices of level-i or higher.*

We can prove this lemma by reducing it to the strong normalizability of the simply typed λ-calculus. We translate $\lambda\mathcal{M}$ to the simply typed λ-calculus by mapping a level-i abstraction $\lambda x^i.\,M$ to an abstract of the simply typed λ-calculus and a level-j ($j < i$) abstraction $\lambda x^j.\,M$ to a pair of x^j and M. Since we prove a stronger property in Theorem 6, we omit the details here.

Note that, we cannot simply map λ-abstractions of $\lambda\mathcal{M}$ to those of the simply typed λ-calculus, since the textual substitution cannot be simulated by the capture-avoiding substitution in the standard calculi.

Theorem 2. *If $\Gamma \vdash M : \alpha$ is derived, then M has a meta-normal form.*

Proof. By repeatedly using Lemma 1. □

Theorem 3. *If $\Gamma \vdash M : \alpha$ is derived and M is meta-normal, then $|M| \leq \max(|\Gamma|, |\alpha|)$.*

Proof. Suppose M is a meta-normal term. We prove the theorem by induction on M.

If M is $(\lambda x^i{:}\beta_1.\,N_0)N_1 \cdots N_n$ for some $i \geq 0, x^i, \beta_1, n \geq 0, N_0, \cdots, N_n$, then the arity of N_0 should be $\gamma \equiv \beta_2 \to_{j_2} \cdots \to_{j_n} \beta_n \to_i \alpha$ where β_k is the arity of N_k for $k = 2, \cdots, n$. Since N_0 is meta-normal, we have $|N_0| \leq \max(|\Gamma|, i, |\gamma|)$ by induction hypothesis, and $|\gamma| \leq i$ by the side-condition of rule (abs). Therefore, we have $|\lambda x^i{:}\beta_1.\,N_0| = \max(i, |N_0|) \leq \max(i, |\Gamma|)$. Now, we have two cases.

1. $n = 0$. $|M| \leq \max(i, |\Gamma|) = \max(|\beta_1 \to_i \gamma|, |\Gamma|)$.
2. $n > 0$. We have, by induction hypothesis, $|N_k| \leq \max(|\Gamma|, |\beta_k|)$ for $1 \leq k \leq n$. Hence $|N_1| \leq \max(|\Gamma|, i)$. Since M is meta-normal, $\max(|N_0|, |N_1|) > i$. Hence $|\Gamma| > i$. Since $|\gamma| \geq |\beta_k|$, we have $|N_k| \leq |\Gamma|$ for $1 \leq k \leq n$. Hence $|M| \leq |\Gamma|$.

We omit the proof of the case where M is of the form $x^i N_1 \cdots N_n$ and we can easily prove the claim in other cases (M is a constant c, $(x^0)\,[N_0]$, or $\langle N_0, N_1 \rangle$) by induction hypothesis. □

The following theorem ensures that meta-level evaluation in $\lambda\mathcal{M}$ does not get stuck.

Theorem 4 (Normal Form Property). *Suppose $\Gamma \vdash M : \alpha$ is derived and M is meta-normal. (1) If $|\Gamma| = |\alpha| = 0$, then $|M| = 0$. (2) If $|\Gamma| < |\alpha|$, then M is a λ-abstraction.*

Proof. (1) Clear from Theorem 3. (2) If M is not a λ-abstraction, then it must be of the form $x^i N_1 N_2 \cdots N_n$ or $(\lambda x^i{:}\gamma.\,N_0)N_1 N_2 \cdots N_n$ for $n \geq 0$. In the former case, $|\alpha| \leq |x^i| \leq |\Gamma|$. Contradiction. In the latter case, let β be the arity of N_0. From Theorem 3, $|(\lambda x^i : \gamma.N_0)N_1| \leq |\beta|$. By the side-condition of rule (abs), $|\beta| \leq i$. Hence $|N_0| \leq i$ and $|N_1| \leq i$, which implies $(\lambda x^i{:}\gamma.\,N_0)N_1$ is a redex. Contradiction. □

We can prove the confluence and strong normalizability of $\lambda\mathcal{M}$. The confluence can be proved using the standard technique (parallel reduction), but the strong normalizability is not trivial. Since space is limited, we just give the idea of the proof here. For the detailed proof, see the full version of this paper.

Lemma 2 (Substitution Lemma). *Let x^i and y^i be distinct variables, $|M_0| \leq i$, $|M_1| \leq i$, and $|N| \leq i$. Then we have*

$$[x^i := N][y^i := M_1]M_0 \equiv [y^i := [x^i := N]M_1][x^i := N]M_0$$

Theorem 5 (Confluence). *The meta-reduction is confluent.*

Theorem 6 (Strong Normalizability). *If $\Gamma \vdash M : \alpha$ is derived, then M is strongly normalizable with respect to the meta-reduction.*

Proof. (idea) We can prove this theorem by using the reducibility method, but we cannot follow the standard way because substitution operations can be applied only when the level restriction is satisfied. In other words, we cannot express the lemma, which is usually claimed in proof by the reducibility method, that

If $\Gamma \vdash M : \alpha$ is derived, then a term that is obtained by substituting reducible terms for free variables in M is reducible.

To deal with this difficulty, we extend $\lambda\mathcal{M}$ so that it has 'postponed substitutions', that is, we define an *extended judgment* of the form $\Gamma_1, x^i := N : \alpha, \Gamma_2 \vdash M$ and give a reduction rule that substitutes N for x in Γ_2 or M under some conditions. Then, we define reducibility sets that consist of extended judgments and prove lemmas similar to the ones in the case of simply typed lambda calculus. □

We remark that, for many object languages such as the simply typed lambda calculus, these theorems still hold if we replace the meta-reduction by the union of the meta-reduction and the reductions in the object language.

7 Related Work and Conclusion

We have proposed two formal systems $\lambda\mathcal{M}$ and λm that formalize the notion of meta-variable, motivated by the observations how meta-variables are used in textbooks of logic.

Edinburgh Logical Framework (LF) [5] gives a typed framework by which we can define various logical systems as object calculi. Unlike LF and its descendants, we clearly distinguish the meta-levels from the object-level. The textual substitution in $\lambda\mathcal{M}$ is another characteristic feature.

Geuvers and Jogjov [3] have introduced the notion of meta-variable into the proof assistant system, so that they can describe open proofs. Their motivation is similar to ours, so they have also encountered the problem that free variables are captured when a meta-variable is instantiated. Their solution to this problem is similar to the one in our work on the calculus of context [11].

Recently much effort has been devoted to formalize the notion of context in lambda calculi, and various calculi of context have been proposed. In the work of Talcott [13] and Mason [8], the notion of contexts are formalized outside of the object language, because contexts are meta-level objects in nature and should be characterized independently of the object language. In other work such as Dami [1], Hashimoto-Ohori [6], Sands [9], and Sato-Sakurai-Kameyama [11], contexts are built into the system, so that context manipulations and object-level calculations can be carried out in the same framework. By representing contexts in $\lambda\mathcal{M}$, we have integrated the two approaches.

It turns out that λm is similar to the calculi for binding-time analysis in off-line partial evaluation with multiple computation stages [4, 2]. In those calculi, types are stratified like ours but, there, levels represent binding time—i.e., when certain expression can be computed. Reduction is similar to ours in that the order of computation is also determined by levels. Aside from the base language (for which they use typed or untyped λ-calculus), one subtle difference is that, in those calculi, variables can range over expressions of the *same* level, resulting in a relaxed condition on function types: $\alpha \to_i \beta$ is a type if $|\alpha| \leq i$ (not $|\alpha| < i$), and $|\beta| \leq i$.

Davies [2] also presents a reformulation $\lambda^{\bigcirc}$ of the calculus for binding-time analysis and shows that it corresponds to a proof system of linear-time temporal logic with a modal operator. Actually, our earlier attempt to formalize meta-variables [14] was done in a close style. In these calculi, terms are annotated with information that indicates levels of subexpressions, hence λm is a simpler calculus to formalize meta-variables.

As far as we know, the calculus $\lambda\mathcal{M}$ is the first one which formalizes non-standard (textual) substitution and is confluent and strongly normalizing. We believe that our calculi can be a basis of formalizing logical and computation systems naturally and directly, but their further development is left for future work.

Acknowledgements: We thank anonymous referees for helpful comments. The first author thanks Henk Barendregt and Jan Willem Klop for having discussions on meta-variables with him. This work is supported in part by Grant-in-Aid for Scientific Research on Priority Areas Research No. 15017247 (Sato, Sakurai) and Grant-in-Aid for Young Scientists (B) No. 15700011 (Igarashi), from MEXT of Japan.

References

1. L. Dami. A Lambda-Calculus for Dynamic Binding. *Theoretical Computer Science*, 192:201–231, 1998.
2. R. Davies. A Temporal-Logic Approach to Binding-Time Analysis. In *11th Annual IEEE Symposium on Logic in Computer Science (LICS'96)*, pages 184–195, 1996.
3. H. Geuvers and G. Jojgov. Open Proofs and Open Terms: A Basis for Interactive Logic. In *CSL 2002*, LNCS 2471, pages 537–552, 2002.
4. R. Glück and J. Jørgensen. Efficient multi-level generating extensions for program specialization. In *Programming Languages, Implementations, Logics and Programs (PLILP'95)*, LNCS 982, pages 259–278, 1995.
5. R. Harper, F. Honsell, and G. Plotkin. A Framework for Defining Logics. *Journal of the Association for Computing Machinery*, 40(1):143–194, 1993.
6. M. Hashimoto and A. Ohori. A Typed Context Calculus. *Theoretical Computer Science*, 266(1-2):249–272, 2001.
7. S. C. Kleene. *Introduction to Metamathematics.* North-Holland, 1952.
8. I. Mason. Computing with Contexts. *Higher-Order and Symbolic Computation*, 12:171–201, 1999.
9. D. Sands. Computing with Contexts - a Simple Approach. *Electronic Notes in Theoretical Computer Science*, 10, 1998.
10. M. Sato. Theory of Judgments and Derivations. In *Discovery Science*, LNAI 2281, pages 78–122, 2001.
11. M. Sato, T. Sakurai, and Y. Kameyama. A Simply Typed Context Calculus with First-Class Environments. *Journal of Functional and Logic Programming*, 2002(4):1–41, 2002.
12. J. R. Shoenfield. *Mathematical Logic.* Addison-Wesley, 1967.
13. C. Talcott. A Theory of Binding Structures and Applications to Rewriting. *Theoretical Computer Science*, 112(1):99–143, 1993.
14. K. Yamamoto, A. Okamoto, M. Sato, and A. Igarashi. A Typed Lambda Calculus with Quasi-quotation (in Japanese). In *Informal Proceedings of the 4th JSSST Workshop on Programming and Programming Languages*, pages 87–102, 2003.