Free Σ-monoids:
A Higher-Order Syntax with Metavariables

Makoto Hamana

Department of Computer Science, Gunma University, Japan
hamana@cs.gunma-u.ac.jp

Abstract. The notion of Σ-monoids is proposed by Fiore, Plotkin and Turi, to give abstract algebraic model of languages with variable binding and substitutions. In this paper, we give a free construction of Σ-monoids. The free Σ-monoid over a given presheaf serves a well-structured term language involving binding and substitutions. Moreover, the free Σ-monoid naturally contains interesting syntactic objects which can be viewed as "metavariables" and "environments". We analyse the term language of the free Σ-monoid by relating it with several concrete systems, especially the λ-calculus extended with contexts.

1 Introduction

In theory of programming languages, we often use some extension of the λ-calculus. When we develop such a theory, we usually use a formal language consisting of both λ-calculus and its meta-language. For example, when we write like "for a term $\lambda x.M \cdots$", this "M" is a *metavariable* denoting some λ-term and is not a (object) variable of the λ-calculus. Since we can also instantiate this M by substitution at the meta-level, we can see that an operation similar to the β-reduction also happens at the meta-level. Sato et al. proposed a series of λ-calculi that formalises both the object-level and such a notion of meta-levels [SSB99, SSK01, SSKI03].

A natural question is what is a good semantics of such a kind of calculus, and clearly existing semantics of the λ-calculus has not covered this object/meta features. This is an interesting mathematical problem. We also expect that it may be a step to theoretical foundation of meta-programming features considered in the traditional declarative languages (such as Lisp's quote and backquote, Prolog's assert) and also revisited recently in the modern programming languages (such as MetaML, FreshML, Boost C++ Library etc.)

This paper shows that the notion of *free Σ-monoids* can provide a unified view of the structure of object and meta-levels in syntax. The structure of Σ-monoids is an abstract algebraic model of syntax with variable binding and substitutions proposed by Fiore, Plotkin and Turi [FPT99]. Although the notion of meta-level was originally not considered for Σ-monoids, we will see that the free construction of them explored here can naturally induce the notions of object and meta-levels. Moreover, the syntax extracted from the free Σ-monoids can be used to analyse and relate existing work on extensions of λ-calculus enriched by several new constructs: contexts, metavariables, and environments.

Organisation and results. Our contribution of this paper is summarised as follows.

(i). To give an explicit free construction M_Σ of Σ-monoid (Sect. 3), and show it is a monad (Sect. 4).

(ii). To show the free Σ-monoid $M_\Sigma X$ naturally has two-level substitution structure (Sect. 5.1):
- the Σ-monoid multiplication β gives capture-avoiding substitution, and
- the monad multiplication μ gives possibly capturing substitution.

(iii). The generators X of the free Σ-monoid can be seen as "metavariables" in the sense of Sato et al. [SSKI03]. (Sect. 5.1).

(iv). To show a link between Plotkin's staged variables [Plo00] and variables decorated with substitutions in contextual calculi [Tal93, Mas99, HO01, San98, SSK01] via a Kan extension (Sect. 5.2, 5.3).

(v). The construct $\lceil x \rceil \langle t_1, \ldots, t_l \rangle$ in the free Σ-monoid which is naturally arisen by the free construction can be seen as an "explicit environment" in the sense of Sato, Sakurai and Burstall [SSB99] (Sect. 5.4).

How to read the paper. This paper is based on the work on semantics of abstract syntax with variable binding by Fiore el al. [FPT99] in the framework of categorical algebra. Basic knowledge of category theory is assumed for reading Sect. 2-4, especially monoids, monads and algebras (e.g. [Mac71] Chap. VI, VII). But the construction of free Σ-monoid (Sect. 3.1 (I)) is purely syntactic, so one can understand it without the knowledge of category theory. Also, in Sect. 5, we see how the notion of Σ-monoids gives a unifying point of view on various syntactic manipulations and constructs, as variable binding, holes of contexts, explicit environments, and substitution of object and meta-variables by terms. So, the reader who is interested in these syntactic aspect can start from Sect. 5 after the preliminaries of Sect. 2.

2 Preliminaries

2.1 Matavariables and Object Variables

In this paper, we precisely distinguish "object-level variables" and "meta-level variables". Before going to the main part, we firstly explain this distinction to avoid confusion.

For example, consider the following λ-term in a certain mathematical context:

$$\lambda a.\lambda b.Ma$$

where M is a λ-term. At the level of text, this M is a meta-level variable "M". The variable a itself is an actual object-level variable "a" [1]. Moreover, there is also an important difference between metavariables and object variables in view of substitutions. If we substitute a term bb for the (object) variable a in the above term, this is actually impossible because usually we assume "capture-avoiding substitutions" in the λ-calculus, i.e.

$$(\lambda a.\lambda b.Ma)[a := bb] = (\lambda a'.\lambda b.Ma')[a := bb] = \lambda a'.\lambda b.Ma'.$$

But the situation in the case of metavariables differs. If we want to substitute a term bb for M, we have

$$(\lambda a.\lambda b.Ma)\{M \mapsto bb\} = \lambda a.\lambda b.(bb)a$$

where $\{- \mapsto -\}$ denotes a meta-level substitution. In this case, although the (object) variable b is *captured* by the binder, usually it does not matter.

If we view these phenomena at the extra meta-level (i.e. meta-meta-level), these two classes of variables are classified by the distinction of substitutions: capture-avoiding and possibly capturing. We use the notions of "object variables" and "metavariables" in this sense (cf. [SSKI03]) and use the following terminology: *metavariable* or simply *variable* for the notion of metavariable, and *object variable* for the notion of object-level variable.

We do not call "object variable" simply variable except for a particular case in this paper.

2.2 Binding Algebras

Now we are going to technical preliminary. We review the notion of binding algebras by Fiore, Plotkin, and Turi. For detail, see [FPT99].

A *binding signature* Σ is consisting of a set Σ of function symbols with an arity function $a : \Sigma \to \mathbb{N}^*$. A function symbol of arity $\langle n_1, \ldots, n_l \rangle$, denoted by $f : \langle n_1, \ldots, n_l \rangle$, has l arguments and binds n_i variables in the i-th argument $(1 \leq i \leq l)$.

Example 1. The signature of the λ-calculus has a function symbol λ of arity $\langle 1 \rangle$, *viz.* λ-abstraction with one argument and binding one variable, and function symbol @ of arity $\langle 0, 0 \rangle$, *viz.* application with two arguments and binding no variables. Hereafter, we refer to this signature as "the signature of the λ-calculus".

The free Σ-monoid we will give in this paper consists of terms constructed by this kind of binding signature. For example, by using the construction rules (I) in Sect. 3.1, the term $\lambda([1]\mathsf{ovar}(1))$ can be constructed. This is an encoding of the λ-term $\lambda x.x$ by the method of de Bruijn levels [FPT99], where [-] denotes a binder and $\mathsf{ovar}(i)$ an object variable i.

[1] Indeed, this can also be seen as a metavariable a denoting some object-level variable. But assuming a itself is an object variable is simpler and there is no difference in reasoning about them (cf. [Pit03])

Let $\mathbb{F}$ be the category which has finite cardinals $n = \{1, \dots, n\}$ (n is possibly 0) as objects, and all functions between them as arrows $m \to n$. This is the category of object variables by the method of de Bruijn levels (i.e. natural numbers) and their renamings. The functor category $\mathbf{Set}^{\mathbb{F}}$ plays an central role in this paper. The objects of it are functors $\mathbb{F} \to \mathbf{Set}$ and the arrows are natural transformations between them. An object $A \in \mathbf{Set}^{\mathbb{F}}$ is often called a *presheaf*.

We define the functor $\delta : \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ as follows: for $L \in \mathbf{Set}^{\mathbb{F}}, n \in \mathbb{F}, \rho \in$ arr $\mathbb{F}$,

$$(\delta L)(n) = L(n+1), \quad (\delta L)(\rho) = L(\rho + \mathrm{id}_1).$$

To a binding signature Σ, we associate the *signature functor* $\Sigma : \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ given by

$$\Sigma A \triangleq \coprod_{f : \langle n_1, \dots, n_l \rangle \in \Sigma} \prod_{1 \le i \le l} \delta^{n_i} A.$$

A Σ-*binding algebra* (or simply Σ-*algebra*) is a pair (A, α) consisting of a presheaf $A \in \mathbf{Set}^{\mathbb{F}}$ and a map[2] $\alpha = [f_A]_{f \in \Sigma} : \Sigma A \longrightarrow A$ called *algebra structure*, where f_A is an *operation*

$$f_A : \delta^{n_1} A \times \dots \times \delta^{n_l} A \longrightarrow A$$

defined for each function symbol $f : \langle n_1, \dots, n_l \rangle \in \Sigma$.

Define the presheaf $\mathrm{V} \in \mathbf{Set}^{\mathbb{F}}$ by

$$\mathrm{V}(n) = n; \quad \mathrm{V}(\rho) = \rho \quad (\rho : m \to n \in \mathbb{F}).$$

In [FPT99], this V is called "the presheaf of variables". More precisely, this V means the presheaf of *object variables*.

Proposition 2. *(***[FPT99]***)* $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathrm{V})$ *forms a monoidal category [Mac71], where the monoidal product is defined as follows. For presheaves A and B,*

$$(A \bullet B)(n) \triangleq (\coprod_{m \in \mathbb{N}} A(m) \times B(n)^m) / \sim \tag{1}$$

where $\sim$ is the equivalence relation generated by

$$(t; u_{\rho 1}, \dots, u_{\rho m}) \sim (A(\rho)(t); u_1, \dots, u_l)$$

for $\rho : m \to l \in \mathbb{F}$. The arrow part of $A \bullet B$ is defined by

$$(A \bullet B)(\rho)(t; u_1, \dots, u_l) \triangleq (t; B(\rho)(u_1), \dots, B(\rho)(u_l))$$

for $\rho : n \to k \in \mathbb{F}$. This is certainly well-defined because the equivalence relation $\sim$ is preserved by the map $(A \bullet B)(\rho)$.

Here and throughout this paper, we use the following notation.

Notation 3. An element of $A(m) \times B(n)^m$ is denoted by $(t; u_1, \dots, u_m)$ where $t \in A(m)$ and $u_1, \dots, u_m \in B(m)$. A representative of an equivalence class in $A \bullet B(n)$ is also denoted by this notation.

[2] [ ] denotes a copair of coproducts.

3 Free Σ-monoids

Definition 4. Let Σ be a signature functor with strength *st* defined by a binding signature. A Σ-*monoid* consists of a *monoid object* [Mac71] $M = (M, \eta : \mathrm{V} \to M, \mu : M \bullet M \to M)$ in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathrm{V})$ with a Σ-binding algebra $\alpha : \Sigma M \to M$ such that

$$\begin{tikzcd}
\Sigma(M) \bullet M \arrow[r, "st"] \arrow[d, "\alpha \bullet M"'] & \Sigma(M \bullet M) \arrow[r, "\Sigma\mu"] & \Sigma M \arrow[d, "\alpha"] \\
M \bullet M \arrow[rr, "\mu"'] & & M
\end{tikzcd}$$

commutes; a morphism of Σ-monoids $(M, \alpha) \longrightarrow (M', \alpha')$ is a morphism in $\mathbf{Set}^{\mathbb{F}}$ which is both Σ-algebra homomorphism and monoid morphism. This defines the category Σ-**Mon** of Σ-monoids.

Theorem 5. ([**FPT99**]) *A free* Σ*-algebra* TV *over* V *is an initial* Σ*-monoid, with the multiplication given in [Prop. 3.5 of [FPT99]]*

A natural question is an existence of a *free* Σ-*monoid*. This means whether is there a Σ-monoid which is freely generated from a given *arbitrary* $X \in \mathbf{Set}^{\mathbb{F}}$ and has universality. TV is an example of it but is not a full answer because this Σ-monoid is only generated[3] from a *particular* presheaf.

Now, our aim is to give a free Σ-monoid, denoted by $M_\Sigma X$, generated from arbitrary $X \in \mathbf{Set}^{\mathbb{F}}$. In the following, we explicitly construct it with substitution monoidal structure as a "language" equipped with the feature of variable binding and substitutions. In contrast to V of the presheaf of object variables, we will show that the presheaf X of generators is regarded as the presheaf of *metavariables*, which was not considered in [FPT99]. The actual situation will be clear in Sect. 5.

Remark 6. Notice that the relationship between the notions of free Σ-algebra and free Σ-monoid is *not* a simple implication. Namely, although TV is a free Σ-algebra over V (equivalently, an initial V + Σ-algebra), it is *not* a free Σ-monoid *over* V. Correctly, TV is a free Σ-monoid *over* $0 \in \mathbf{Set}^{\mathbb{F}}$ of the empty set functor, i.e. in our notation, $\mathrm{TV} = M_\Sigma 0$. This is easily checked because the explicit construction of TV ([FPT99] Sect. 2) is the same as the construction (I) of $M_\Sigma X$ without using the rule (⧫) (because of $X = 0$) below.

3.1 Construction

Let Σ be a binding signature and X an arbitrary presheaf in $\mathbf{Set}^{\mathbb{F}}$. We construct the Σ-monoid $M_\Sigma X$ by the following four steps and show that it is free.

[3] See Remark 6.

(I) The presheaf $M_\Sigma X$. First we define the set $\bar{M}_\Sigma X(n)$ indexed by $n \in \mathbb{N}$ by the following construction rules (by starting from the first rule).

$$\frac{i \in \mathrm{V}(n)}{\mathsf{ovar}(i) \in \bar{M}_\Sigma X(n)}\ \nu(n)$$

$$\frac{f : \langle i_1, \dots i_l\rangle \in \Sigma \quad t_1 \in \bar{M}_\Sigma X(n{+}i_1) \ \cdots \ t_l \in \bar{M}_\Sigma X(n{+}i_l)}{f([n{+}1, \dots, n{+}i_1]t_1, \dots, [n{+}1, \dots, n{+}i_l]t_l) \in \bar{M}_\Sigma X(n)}\ f_{M_\Sigma X}(n)$$

$$\frac{x \in X(l) \quad t_1, \dots, t_l \in \bar{M}_\Sigma X(n)}{\lceil x \rceil \langle t_1, \dots, t_l\rangle \in \bar{M}_\Sigma X(n)} \qquad (\blacklozenge)$$

Note that any $l \in \mathbb{N}$ with $X(l) \neq \varnothing$ is possible to apply the rule ($\blacklozenge$) because X is a functor $\mathbb{F} \to \mathbf{Set}$. So, l is possibly 0, i.e. if $x \in X(0)$, $\lceil x \rceil\langle\rangle \in \bar{M}_\Sigma X(n)$ for all $n \in \mathbb{N}$.

Notation 7. We often simply write $\lceil x \rceil$ for $\lceil x \rceil\langle \mathsf{ovar}(1), \dots, \mathsf{ovar}(l)\rangle$. Since the binders are clear from the arity of function symbol $f \in \Sigma$, we often abbreviate $f([n{+}1, \dots, n{+}i]t_1, \dots, [n{+}1, \dots, n{+}l]t_l)$ as just $f(t_1, \dots, t_l)$.

Define the equivalence relation $\doteq$ on $\bar{M}_\Sigma X(n)$ generated by context (by function symbols in Σ) closure of "the axiom"

$$\lceil x \rceil\langle t_{\rho 1}, \dots, t_{\rho l}\rangle \ \doteq\ \lceil X(\rho)(x) \rceil\, \langle t_1, \dots, t_m\rangle \tag{2}$$

for every $\rho : l \to m \in \mathbb{F}, x \in X(l), t_1, \dots, t_m \in \bar{M}_\Sigma X(n)$ (so, for example, $f(\lceil x \rceil\langle t_2, t_1\rangle) \doteq f(\lceil X(\rho)(x) \rceil\, \langle t_1, t_2\rangle)$). The presheaf $M_\Sigma X \in \mathbf{Set}^{\mathbb{F}}$ is defined by $M_\Sigma X(n) \triangleq \bar{M}_\Sigma X(n) \,/ \doteq$, and for $\rho : n \to n' \in \mathbb{F}$, the arrow part is defined by structural induction (see Theorem 8):

$$\begin{aligned}
M_\Sigma X(\rho)(\mathsf{ovar}(i)) &= \mathsf{ovar}(\rho(i))\\
M_\Sigma X(\rho)(f(t_1, \dots, t_l)) &= f(M_\Sigma X(\rho)(t_1), \dots, M_\Sigma X(\rho)(t_l))\\
M_\Sigma X(\rho)(\lceil x \rceil\langle t_1, \dots, t_l\rangle) &= \lceil x \rceil\langle M_\Sigma X(\rho)(t_1), \dots, M_\Sigma X(\rho)(t_l)\rangle.
\end{aligned}$$

The most important point of this construction is the invention of the rule ($\blacklozenge$), and the related construct $\lceil x \rceil\langle t_1, \dots, t_n\rangle$ and axiom (2). Syntactic meanings of this construct will be discussed in detail in Sect. 5. The idea of rule ($\blacklozenge$) comes from a construction of free symmetric multicategory. If the reader is familiar with categorical type theory, this rule may be able to be understood as the (formal) composition of arrows with the principle of "substitution as composition".

(II) Σ-algebra. $M_\Sigma X$ has the $\mathrm{V} + \Sigma$-algebra structure

$$[\ \nu, [f_{M_\Sigma X}]_{f\in\Sigma}\ ] : \mathrm{V} + (\Sigma M_\Sigma X) \longrightarrow M_\Sigma X$$

where the maps $\nu, f_{M_\Sigma X}$ are defined by the construction rule of the presheaf $\bar{M}_\Sigma X$ (the mappings from the upper to the lower in the construction rules). Define the map $\sigma : X \bullet M_\Sigma X \to M_\Sigma X$ by

$$(x; t_1, \dots, t_l) \longmapsto \lceil x \rceil\langle t_1, \dots, t_l\rangle$$

This is well-defined because it maps a $\sim$-equivalnce class to a $\doteq$-equivalnce class. So this is also an algebra structure. Moreover, we can show the following.

Theorem 8. $(M_\Sigma X, [\,\nu, [f_{M_\Sigma X}]_{f\in\Sigma}, \sigma\,])$ *is an initial* $(V+\Sigma+X\bullet -)$*-algebra.*

Proof. This is proved by standard argument of showing uniquness and structural induction. □

Hence, we can use *structual induction* on terms of $M_\Sigma X$ (e.g. when we define a map whose domain is $M_\Sigma X$).

Remark 9. A $(V+\Sigma+X\bullet-)$-algebra always satisfies "the axiom" (2) although it is simply an algebra defined by an signature functor (i.e. it is not defined as a (Σ, E)-algebra). This is because the functor $X\bullet -$ is defined by a quotient.

(III) Monoid. We construct the monoid $(M_\Sigma X, \nu, \beta)$ in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathrm{V})$. The unit $\nu : \mathrm{V} \to M_\Sigma X$ is already defined in (I). The multiplication $\beta : M_\Sigma X \bullet M_\Sigma X \to M_\Sigma X$ is defined inductively as follows. Here, $[n{+}1, \dots, n{+}k]t$ is abbreviated as $[n{+}\vec{k}]t$.

$$\beta(n) : \coprod_m M_\Sigma X(m) \times M_\Sigma X(n)^m / \sim \longrightarrow M_\Sigma X(n)$$

$$\begin{aligned}
\beta(n)(\mathsf{ovar}(i);\, \vec{t}) &= t_i \\
\beta(n)(\lceil x\rceil\langle s_1,\dots,s_l\rangle;\, \vec{t}) &= \lceil x\rceil\langle \beta(n)(s_1;\vec{t}),\dots,\beta(n)(s_l;\vec{t})\rangle \quad (x\in X(l)) \\
\beta(n)(f([n{+}\vec{i_1}]s_1,\dots,[n{+}\vec{i_l}]s_l);\, \vec{t}) & \\
= f([n{+}\vec{i_1}]\,\beta(n{+}i_1)(s_1;\, &\mathsf{up}_{i_1}(\vec{t}), \mathsf{ovar}(n{+}1),\dots,\mathsf{ovar}(n{+}i_1)),\dots \qquad (3)\\
[n{+}\vec{i_l}]\,\beta(n{+}i_l)(s_l;\, &\mathsf{up}_{i_l}(\vec{t}), \mathsf{ovar}(n{+}1),\dots,\mathsf{ovar}(n{+}i_l))
\end{aligned}$$

where $f : \langle i_1,\dots,i_l\rangle \in \Sigma$ and $\vec{t}$ denotes $t_1,\dots,t_m$, and the weakening map from $M_\Sigma X(n)$ to $M_\Sigma X(n{+}i)$ is defined by $\mathsf{up}_i \triangleq M_\Sigma(\mathrm{id}_n + \mathsf{w}_i)$ where $\mathsf{w}_i : 0 \to i$. This is well-defined, and the naturality of β follows from the definitions of $M_\Sigma X(\rho)$. The isomorphisms $\mathrm{V}\bullet M_\Sigma X \cong M_\Sigma X$, $M_\Sigma X \bullet \mathrm{V} \cong M_\Sigma X$, $(M_\Sigma X \bullet M_\Sigma X)\bullet M_\Sigma X \cong M_\Sigma X \bullet (M_\Sigma X \bullet M_\Sigma X)$ of monoid are defined by

$$(i;\vec{t}) \mapsto t_i\ ,\ (t;1,\dots,l) \mapsto t$$
$$((s;\vec{t});\vec{u}) \mapsto (s;(t_1;\vec{u}),\dots,(t_l;\vec{u}))$$

The inverse mappings are obvious except for $u : M_\Sigma X \to \mathrm{V}\bullet M_\Sigma X$. This u is defined by $t \mapsto (1;t)$ because in $\mathrm{V}\bullet M_\Sigma X(n) = \coprod_m m \times M_\Sigma X(n)^m/\sim$, always

$$(i;t_1,\dots,t_l) \sim (1;t_i)$$

holds by taking $\rho : 1 \to m, 1 \mapsto i$ in the definition of $\sim$; thus u is an isomorphism. The monoid laws are proved by induction on the terms in $M_\Sigma X$.

(IV) Σ-monoid. The remaining task is to show that the monoid $(M_\Sigma X, \nu, \beta)$ makes the following diagram of Σ-monoid law commutative. The strength st is the one defined in [FPT99].

$$\begin{tikzcd}
\Sigma(M_\Sigma X) \bullet M_\Sigma X \arrow[r, "st"] \arrow[d, "\alpha \bullet M_\Sigma X"'] & \Sigma(M_\Sigma X \bullet M_\Sigma X) \arrow[r, "\Sigma\beta"] & \Sigma M_\Sigma X \arrow[d, "\alpha"] \\
M_\Sigma X \bullet M_\Sigma X \arrow[rr, "\beta"'] & & M_\Sigma X
\end{tikzcd}$$

Instantiating this diagram at $n \in \mathbb{F}$ and chasing an element, this eventually becomes the equality

$$\begin{aligned}
&\beta(n)(f([n{+}\vec{i_1}]s_1, \ldots, [n{+}\vec{i_l}]s_l);\, \vec{t}) \\
&\quad = f([n{+}\vec{i_1}]\,\beta(n{+}i_1)(s_1;\, \mathsf{up}_{i_1}(\vec{t}), \mathsf{ovar}(n{+}1), \ldots, \mathsf{ovar}(n{+}i_1)), \ldots)
\end{aligned}$$

In fact, this is true because it is nothing but the equation (3) [4] in the definition of β. Hence, we have the following.

Proposition 10. *$(M_\Sigma X, \nu, \beta)$ is a Σ-monoid.*

3.2 Universality

Definition 11. An *assignment* $\phi : X \to A$ is a morphism of $\mathbf{Set}^{\mathbb{F}}$ whose target A has a Σ-monoid structure $(A, \tilde{\nu}, \tilde{\beta})$.

Then, this is extended to a Σ-monoid morphism $\phi^* : M_\Sigma X \to A$ as follows.

$$\begin{aligned}
M_\Sigma X(n) &\longrightarrow A(n) \\
\mathsf{ovar}(i) &\longmapsto \tilde{\nu}(n)(i) \\
f(t_1, \ldots, t_l) &\longmapsto f_A(\phi^*(n{+}i_1)(t_1), \ldots, \phi^*(n{+}i_l)(t_l)) \\
\ulcorner x \urcorner \langle t_1, \ldots, t_l \rangle &\longmapsto \tilde{\beta}(n)(\,\phi(l)(x);\, \phi^*(n)(t_1), \ldots \phi^*(n)(t_l)\,)
\end{aligned}$$

where $f : \langle i_1, \ldots, i_l \rangle \in \Sigma$. Checking this is certainly a Σ-monoid morphism is straightforward.

Lemma 12.
*Let $\phi : X \to A$ and $\psi : X \to B$ be assignments. For a Σ-monoid morphism $h : B \to A$ such that $\phi = h \circ \psi$ in $\mathbf{Set}^{\mathbb{F}}$, the right diagram commutes in Σ-**Mon**.*

$$\begin{tikzcd}
M_\Sigma X \arrow[r, "\psi^*"] \arrow[dr, "\phi^*"'] & B \arrow[d, "h"] \\
& A
\end{tikzcd}$$

Proof. Instantiating the diagram at $n \in \mathbb{F}$, use induction on the structure of terms. □

[4] This just expresses an inductive application of β, i.e. roughly, $\beta(n)(f(s_1, \ldots, s_l);\, \vec{t}) = f(\beta(n+i_1)(s_1;\, \vec{t}), \ldots, \beta(n+i_l)(s_l;\, \vec{t}))$.

Finally, we show the Σ-monoid $M_\Sigma X$ has the following universality, which means freeness.

Proposition 13.
Let A be a Σ-monoid and $\phi : X \to A$ an assignment. Then, there exists a unique Σ-monoid morphism $\hat{\phi}$ that makes the right diagram commutative where $\eta_X(n)$ maps x to $\lceil x \rceil$.

$$\begin{tikzcd} X \arrow[r, "\eta_X"] \arrow[dr, "\phi"'] & M_\Sigma X \arrow[d, "\hat{\phi}"] \\ & A \end{tikzcd}$$

Proof. Actually, such $\hat{\phi}$ exists by taking $\hat{\phi} = \phi^*$. For uniqueness, take $B = M_\Sigma X$ in Lemma 12. □

Theorem 14. *$(M_\Sigma X, \nu, \beta)$ is the free Σ-monoid over $X \in$ **Set**$^{\mathbb{F}}$.*

This universality can be rephrased as an adjunction:

Corollary 15. *The functor U : Σ-**Mon** → **Set**$^{\mathbb{F}}$ that forgets the Σ-monoid structure has the left adjoint M_Σ, i.e. there is an adjunction*

$$\begin{tikzcd} \textbf{Set}^{\mathbb{F}} \arrow[r, shift left, "M_\Sigma"] \arrow[r, phantom, "\perp"] & \Sigma\text{-}\textbf{Mon} \arrow[l, shift left, "U"] \end{tikzcd}$$

4 Free Σ-Monoid Construction is a Monad

By Corollary 15 and the basic theorem of category theory (every adjunction gives a monad [Mac71]), we know that UM_Σ gives a monad on **Set**$^{\mathbb{F}}$. We state this with the usual identification regarding M_Σ as the endofunctor UM_Σ on **Set**$^{\mathbb{F}}$.

Theorem 16. *(M_Σ, η, μ) is a monad on **Set**$^{\mathbb{F}}$.*

It is valuable to concretely describe this monad and show the theorem to know the internal structure of the language given by the free Σ-monoid. So, in this section, we explicitly show that M_Σ gives a monad on **Set**$^{\mathbb{F}}$.

Functor. First, we describe the functor M_Σ : **Set**$^{\mathbb{F}}$ → **Set**$^{\mathbb{F}}$. In Sect. 3.1 (I), we have defined the object part of M_Σ by giving the construction of the free Σ-monoid over a presheaf X. Now, we define the arrow part: for $\phi : X \to Y \in$ arr **Set**$^{\mathbb{F}}$, $M_\Sigma(\phi) : M_\Sigma X \to M_\Sigma Y$ is defined as follows:

$$\begin{aligned} M_\Sigma\phi(n)(\mathsf{ovar}(i)) &= \mathsf{ovar}(i) \\ M_\Sigma\phi(n)(f(t_1,\ldots,t_l)) &= f(M_\Sigma\phi(n+i_1)(t_1),\ldots,M_\Sigma\phi(n+i_l)(t_l)) \\ M_\Sigma\phi(n)(\lceil x\rceil\langle t_1,\ldots,t_l\rangle) &= \lceil \phi(n)(x)\rceil\langle M_\Sigma\phi(n)(t_1),\ldots,M_\Sigma\phi(n)(t_l)\rangle \end{aligned}$$

where $f : \langle i_1,\ldots,i_l\rangle \in \Sigma$.

Unit. The unit $\eta : \mathsf{Id} \to M_\Sigma$ is defined by $\eta_X(n) : X(n) \to M_\Sigma X(n),\ x \mapsto \lceil x \rceil$.

Multiplication. The multiplication $\mu : M_\Sigma \circ M_\Sigma \to M_\Sigma$ is defined by

$$\begin{aligned}
\mu_X(n) : M_\Sigma(M_\Sigma X)(n) &\longrightarrow M_\Sigma X(n)\\
\mathsf{ovar}(i) &\longmapsto \mathsf{ovar}(i)\\
f(t_1,\ldots,t_l) &\longmapsto f(\mu_X(n+i_1)(t_1),\ldots,\mu_X(n+i_l)(t_l))\\
\lceil s\rceil\langle t_1,\ldots,t_l\rangle &\longmapsto \beta(n)(s;\mu_X(n)(t_1),\ldots\mu_X(n)(t_l))
\end{aligned}$$

where $f : \langle i_1,\ldots,i_l\rangle \in \Sigma$, and $s \in M_\Sigma X(m),\ t_1,\ldots,t_l \in M_\Sigma M_\Sigma X(n)$.

Monad laws. We write M for M_Σ for simplicity. The unit laws of the monad is proved straightforwardly by induction of terms in $MX(n)$. The associative law of the monad is also proved by induction, but more cumbersome. The associative law at $X \in \mathbf{Set}^{\mathbb{F}}$ and $n \in \mathbb{F}$ is

$$\begin{tikzcd}
MMMX(n) \arrow[r, "M\mu_X(n)"] \arrow[d, "\mu_{MX}(n)"'] & MMX(n) \arrow[d, "\mu_X(n)"] \\
MMX(n) \arrow[r, "\mu_X(n)"'] & MX(n)
\end{tikzcd}$$

Namely, we need to prove the equation

$$\mu_X(n) \circ M\mu_X(n)\ (w) = \mu_X(n) \circ \mu_{MX}(n)\ (w) \qquad (4)$$

for all $w \in MMMX(n)$. We proceed by induction on the structure of w. The cases $w = \mathsf{ovar}(i), f(\vec{t})$ are straightforward chasing. The case $w = \lceil s\rceil\langle\vec{t}\rangle$ is again cumbersome. For simplicity, hereafter we omit subscripts and the component parameter n of the natural transformations μ, β. By using $M\mu$ as an arrow part of the functor M_Σ defined above, the equation (4) becomes

$$\begin{aligned}
\text{lhs} &= \mu(M\mu(\lceil s\rceil\langle\vec{t}\rangle)) = \mu(\ \lceil\mu(s)\rceil\langle M\mu(\vec{t})\rangle\ )\\
&= \beta(\ \mu(s);\ \mu(M\mu(\vec{t}))\ ) \overset{\text{I.H.}}{=} \beta(\ \mu(s);\ \mu\mu(\vec{t})\ ),\\
\text{rhs} &= \mu(\ \beta(s;\ \mu(\vec{t}))\ ).
\end{aligned}$$

So, we need to prove

$$\beta(\ \mu(s);\ \mu\mu(\vec{t})\ ) = \mu(\ \beta(s;\ \mu(\vec{t}))\ ). \qquad (5)$$

Conceptually, this means a commutation of the monoid multiplication β and the monad multiplication μ. This point will be discussed with a relationship to contextual calculi in Sect. 5.2. The equation (5) is proved again by induction on the structure of $s \in MMX(n)$. The case $s = u\langle\vec{v}\rangle$ is the most complicated case, but a careful equational calculation shows it. Hence, we have shown Th. 16.

A structure similar to this monoad M_Σ and monoid $M_\Sigma X$ is also considered by Ghani and Uustalu [GU03] from the viewpoint of combination of two signatures. They used finitary monads in the category $[\mathbf{Set}, \mathbf{Set}]_f$ of finitary functors, instead of monoids in $\mathbf{Set}^{\mathbb{F}}$, to model substitutions. This finitary monad approach is equivalent to monoid approach in $\mathbf{Set}^{\mathbb{F}}$ used here because the equivalence of categories $[\mathbf{Set}, \mathbf{Set}]_f \simeq \mathbf{Set}^{\mathbb{F}}$. Hence, our M_Σ can also be seen as a structure having monads in two-levels, i.e. it induces the monads on $[\mathbf{Set}, \mathbf{Set}]_f$ and on $\mathbf{Set}$.

5 Analysis on the Term Language of Free Σ-monoids

5.1 Multiplications as Substitution operations

An intuition of the multiplication μ of the monad M_Σ is the operation that "erases" (or "collapse") all the outermost brackets "$\lceil - \rceil$". This operation can also be considered as a substitution. Because, for example, performing the multiplication

$$\mu_X(n)(f(\lceil t \rceil)) = f(t)$$

can be rewritten as "applying a substitution"

$$\mu_X(n)(\ f(\lceil * \rceil)\{* \mapsto t\}\ ) = f(t)$$

Here, the notation $f(\lceil \_ \rceil)\{\_ \mapsto \_\}$ is another representation of an element of $M_\Sigma \circ M_\Sigma$, which is a syntactic form of (suspended) substitution. Although we can more rigorously define this identification, we keep this informal for simplicity.

There are two important properties of μ as substitution by this understanding. We state this:

(i). μ performs a substitution of *metavariables*.
(ii). μ performs a *possibly capturing* substitution.

The above examples shows (i). Why this is metavariable substitution is that the construct $\lceil t \rceil$ is considered as a metavariable in $M_\Sigma(M_\Sigma X)$ (or $*, t$ are metavariables in $M_\Sigma X$). In this case, the generators $M_\Sigma X$ is considered as metavariables.

Interestingly and importantly, we see that μ has the property (ii). For example, we have

$$\mu_X(0)(\ \lambda([1]\lceil \mathsf{ovar}(1) \rceil)\ ) = \lambda([1]\mathsf{ovar}(1))$$

by using the signature of the λ-calculus given in Example 1. Here, the object variable 1 is *captured* by the binder "[1]".

Notice that this kind of capturing cannot happen in the case of the Σ-monoid multiplication $\beta : M_\Sigma X \bullet M_\Sigma X \to M_\Sigma X$. Consider a similar try:

$$\beta(\ \lambda([1]\mathsf{ovar}(1)); \mathsf{ovar}(2)\ ).$$

This expresses that we want to replace the object variable $\mathsf{ovar}(a)$ inside λ with the free object variable $\mathsf{ovar}(b)$. But actually the term is not well-formed, i.e.

$$(\lambda([1]\mathsf{ovar}(1)); \mathsf{ovar}(2)) \notin M_\Sigma X \bullet M_\Sigma X(2).$$

Hence, we state the following by the definition of β and this observation:

(iii). β performs a substitution of *object variables*.
(iv). β performs a *capture-avoiding* substitution.

5.2 Staged Variables and a Construction of a Presheaf of Metavariables

We have obtained a construction of the free Σ-monoid $M_\Sigma X$ over $X \in \mathbf{Set}^{\mathbb{F}}$ and shown that M_Σ is a monad. As a consequence, we can think of X as a kind of "variables" as in the case of first-order universal algebra.

But what is a presheaf X of variables? In the case of first-order universal algebra, elements in the set X of generators can be just considered as syntactic constants of variables. In the case of binding algebra, this is not so simple, because X itself is a presheaf. One may guess that the component $X(n)$ for each n is a set of variables. This seems feasible but what is a concrete meaning of a "variable" x in a particular component $X(n)$ is still not clear, especially we should know the meaning of the index n. Also, since X must be a functor, we need to consider the functoriality of X.

A hint to answer this problem can be found in Plotkin's "Metalanguage for programming with bound object variables" [Plo00]. He considered the notion of "staged variables" for this metalanguage. Let us see this notion by adapting it for our term language. A variable x has a "stage" n that is a set $\{1, \ldots, n\}$ of object variables, and it is denoted by $x : n$. A typing judgment

$$x : n \vdash t : n$$

means that as usual, t depends on a variable x, and also the term t depends only *on object variables* in n. A staged variable $x : n$ is a variable which means that it is only instantiated by a term having free object variables from 1 to n. The staged variables forms an $\mathbb{N}$-indexed set X by setting $x \in X(n)$ iff $x : n$.

An $\mathbb{N}$-indexed set X is "almost" a presheaf in $\mathbf{Set}^{\mathbb{F}}$. But clearly it is not sufficient because it lacks the arrow part. So, we need to seek some canonical way of constructing a presheaf $X' \in \mathbf{Set}^{\mathbb{F}}$ from an $\mathbb{N}$-indexed set X of staged variables.

Fortunately, there is a construction for it. The category of $\mathbb{N}$-index sets can be expressed as the presheaf category $\mathbf{Set}^{\mathbb{N}}$ (by considering $\mathbb{N}$ as a discrete category). Thus, this problem can be abstracted to that of finding a construction of a presheaf in $\mathbf{Set}^{\mathbb{F}}$ from a given presheaf in $\mathbf{Set}^{\mathbb{N}}$. This can be obtained by a Kan extension. Namely, for the inclusion functor J, we have the left Kan extension:

$$\begin{tikzcd}
\mathbb{N} \arrow[r, "J"] \arrow[d, "X"'] & \mathbb{F} \arrow[dl, "\mathrm{Lan}_J X"] \\
\mathbf{Set} &
\end{tikzcd}$$

We write this extension as $\hat{X} \in \mathbf{Set}^{\mathbb{F}}$ and calculate it as follows. By the coend formula of the left Kan extension (where "$\cdot$" denotes a copower) [Mac71] and discreteness of $\mathbb{N}$, we have

$$\hat{X}(n) = (\mathrm{Lan}_J X)(n) = (\int^{k \in \mathbb{N}} \mathbb{F}(Jk, -) \cdot X(k))(n) = \coprod_{k \in \mathbb{N}} \mathbb{F}(k, n) \times X(k)$$

with the obvious arrow part. We use the last coproduct formula as a *definition* of $\hat{X}$ for a given $\mathbb{N}$-indexed set X of *staged variables*.

5.3 Variables Decorated with Substitutions

Next, we consider syntactic meaning of "variables" in $\hat{X}(n)$. Now we know that an element of $\hat{X}(n)$ is a pair (ξ, x) where $x : k$ (i.e. $x \in X(k)$) and $\xi : k \to n \in \mathbb{F}$. So, we use the notation

$$x^{\xi} \in \hat{X}(n)$$

and call this syntactic construct *a variable with renamer* [Ham01, Ham03] where ξ is a renamer, because ξ is a renaming function on object variables.

By the term construction rule in Sect. 3.1 (I), the above variable with renamer x^{ξ} becomes the term

$$\lceil x^{\xi} \rceil = \lceil x^{\xi} \rceil \langle \mathsf{ovar}(1), \ldots, \mathsf{ovar}(n) \rangle \doteq \lceil x \rceil \langle \mathsf{ovar}(\xi 1), \ldots, \mathsf{ovar}(\xi k) \rangle \in M_{\Sigma} \hat{X}(n)$$

where we are using "the axiom" (2). (In particular, if $\xi : 0 \to n$, $\lceil x^{\xi} \rceil \doteq \lceil x \rceil \langle \rangle \in M_{\Sigma}\hat{X}(n)$). Hence, a variable with renamer is merely a particular form of the construct $\lceil x \rceil \langle \cdots \rangle$. Or, one may call it a variable decorated with substitution.

One may seem that this is merely a semantically derived object, or this has no actual computational usefulness. However, it is not true. The same syntactic construct, a variable decorated with substitution, has appeared in several extended λ-calculus for computing with *contexts* [Tal93, Mas99, HO01, San98, SSK01] (also, the same construct is appeared in the series of work based on Nominal Logic [Pit03], e.g. "suspension" construct in [UPG03]). Their extensions of λ-calculus with contexts have the feature of "hole variables" meaning holes of contexts. In these calculi, the notion of "holes decorated with substitutions" plays an essential role to ensure *commutativity of β-reduction and hole-filling operation.* This problem will be clear by the following argument by Sands [San98] on failure of naive extension of the λ-calculus with holes. Later, we will see that this relates to the free Σ-monoid.

Consider the λ-calculus extended with the hole $\Box$ syntactically. If we consider this extended λ-calculus naively, we have the following kind of reduction:

$$(\lambda x.\Box)y \;\to_{\beta}\; \Box.$$

But this is not adequate, since filling the hole with x does not commute with this β-reduction:

$$\begin{tikzcd}
(\lambda x.\Box)I \arrow[rr, "\beta\text{−red.}"'] \arrow[d, "\text{fill with } x"'] & & \Box \arrow[d, "\text{fill with } x"] \\
(\lambda x.x)I \arrow[r, "\beta\text{−red.}"'] & I \arrow[r, phantom, "\not\equiv"] & x
\end{tikzcd} \tag{6}$$

Notice that the operation of filling a hole allows the capture of the variable x by the binder, which is the main feature of contextual calculi. The problem of the above non-commutativity is due to the fact that the reduction step "forgets" the term I. So, the solution in contextual calculi is to syntactically decorate holes with explicit substitutions, so that e.g.

$$(\lambda x.\Box)I \ \rightarrow_\beta \ \Box^{\{I/x\}}. \tag{7}$$

Then, by using this modified β-reduction, the above diagram commutes because the both reductions go to I.

Interestingly, the above diagram has already appeared in this paper. In the proof of monad law in Sect. 4, we encountered the equation (5)

$$\beta(\,\mu(s);\,\mu\mu(\vec{t})\,) = \mu(\,\beta(s;\,\mu(\vec{t}))\,).$$

This is diagrammatically,

$$\begin{tikzcd}
(s;\,\mu(\vec{t})) \in MMX \bullet MMX \arrow[r, "\beta"] \arrow[d, "\mu \bullet \mu"'] & MMX \arrow[d, "\mu"] \\
MX \bullet MX \arrow[r, "\beta"'] & MX
\end{tikzcd}$$

and we have proved that it commutes. We claim that *this is nothing but the commutativity of the diagram (6)*. The reason is as follows. We have seen that our metavariables allow possibly capturing substitution in Sect. 5.1. Namely, our metavariables behave exactly the same as holes in contextual calculi. This means that the substitution operation μ of metavariables can be seen as the operation of filling holes. And since β-reduction is a replacement of object-level variables, it corresponds to the substitution of object variables by the monoid multiplication β. Hence, the commutativity (5) of β and μ is the same as the diagram (6)[5]. This means the diagram (6) is an instance of monad law of the free Σ-monoid.

More concretely, Sands' discussion can be formulated in our term language. Assume the beta-axiom of the λ-calculus by using the binding signature of λ-calculus:

$$\lambda([1]\lceil M\rceil)@\lceil N\rceil = \beta(\lceil M\rceil;\lceil N\rceil)$$

where metavariables $M \in X(1)$ and $N \in X(0)$. Then, using the metavariable $* \in X(1)$, we have

$$\lambda([1]\lceil *\rceil)@I = \beta(\lceil *\rceil; I) = \lceil *\rceil\langle I\rangle.$$

By regarding the bound object variable 1 as x and the metavariable $\lceil *\rceil$ as $\Box$, this *precisely* corresponds to the reduction (7).

This is an interesting link between the notions used in the different areas. We have seen that Plotkin's notion of staged variables naturally induces the notion of decorated variables that has been used for computing with contextual holes.

[5] So, our implicit intension of naming β the Σ-monoid multiplication is this similarity to β-reduction. Notice also that β-reduction is a substitution of object-level variables.

5.4 Explicit Environments

As we have seen, we can consider the Σ-monoid $M_\Sigma X$ as a "language" having the features of binding, substitutions, and metavariables. It is not only practically shown, but also theoretically justified, i.e. it has the desired universal property (Th. 13). Hence, we summarise it formally:

Definition 17. A *higher-order syntax with metavariables* is specified by

- a given binding signature Σ, and
- an indexed set $X(n)$ of staged variables[6] for each $n \in \mathbb{N}$.

The presheaf $M_\Sigma \hat{X}$ of terms is constructed by the construction rules (I) in Section 3.1 where $\hat{X}(n) = \coprod_{k \in \mathbb{N}} \mathbb{F}(k,n) \times X(k)$. Then, terms are expressed as the following BNF:

$$\begin{aligned} M_\Sigma \hat{X}(n) \ni &\quad t ::= \mathsf{ovar}(i) \mid f(t_1, \ldots, t_l) \mid \lceil x \rceil \langle t_1, \ldots, t_l \rangle \\ X(l) \ni &\quad x \end{aligned}$$

where $i \in n$.

This syntax is standard[7] except for the seemingly exotic construct $\lceil x \rceil \langle t_1, \ldots, t_l \rangle$. We call this construct an *explicit environment* which follows the terminology in Sato, Sakurai and Burstall's λ-calculus with "explicit environments" [SSB99]. Namely, $\lceil x \rceil \langle t_1, \ldots, t_l \rangle$ can be seen as a first-class representation of an environment (i.e. a list of (variable,value)-pairs).

Because the sequence part always means substitutes of object variables from 1 to l, informally a term $\lceil x \rceil \langle t_1, \ldots, t_l \rangle$ means

$$\lceil x \rceil \, \langle 1 \mapsto t_1, \ldots, l \mapsto t_l \rangle.$$

This is a "suspended" substitution because the substitution process does not happen before instantiating the variable x. Our understanding is that a term $\lceil x \rceil \langle t_1, \ldots, t_l \rangle$ is an explicit environment having a "placeholder" $\lceil \_ \rceil$ that is named "x" and it waits for an actual term to evaluate (by the multiplication μ) under the environment $\langle t_1, \ldots, t_l \rangle$.

Acknowledgments. The discussion with Masahiko Sato on the role of metavariables in "calculi of metavariables" was an important step of this work; I would like to thank him. I also thank Neil Ghani, and the anonymous referees for useful comments. This work is supported by the JSPS Grant-in-Aid for Scientific Research (16700005).

[6] These are metavariables.
[7] By regarding $\mathsf{ovar}(i)$ as an object variable.

References

[FPT99] M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proc. 14th Annual Symposium on Logic in Computer Science*, pages 193–202, 1999.

[GU03] N. Ghani and T. Uustalu. Explicit substitutions and higher order syntax. In *Proceedings of 2nd ACM SIGPLAN Workshop on Mechanized Reasoning about Languages with Variable Binding, MERLIN'03*, pages 135–146, 2003.

[Ham01] M. Hamana. A logic programming language based on binding algebras. In *4th International Symposium on Theoretical Aspects of Computer Software (TACS 2001)*, LNCS 2215, pages 243–262, 2001.

[Ham03] M. Hamana. Term rewriting with variable binding: An initial algebra approach. In *Proceedings of Fifth ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP2003)*, pages 148–159. ACM Press, 2003.

[HO01] M. Hashimoto and A. Ohori. A typed context calculus. *Theoretical Computer Science*, 266:249–271, 2001.

[Mac71] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1971.

[Mas99] Ian A. Mason. Computing with contexts. *Higher-Order and Symbolic Computation*, 12(2):171–201, 1999.

[Pit03] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Information and Computation*, 186:165–193, 2003.

[Plo00] G. Plotkin. Another meta-language for programming with bound names modulo renaming. In *Winter Workshop in Logics, Types and Rewriting*, Heriot-Watt University, February 2000. Lecture slides.

[San98] D. Sands. Computing with contexts: A simple approach. In *Second Workshop on Higher-Order Operational Techniques in Semantics (HOOTS II)*, volume 10 of *Electronic Notes in Theoretical Computer Science*, 1998.

[SSB99] M. Sato, T. Sakurai, and R. Burstall. Explicit environments. In *In Proceedings of TLCA'99*, LNCS 1581, pages 340–354, 1999.

[SSK01] M. Sato, T. Sakurai, and Y. Kameyama. A simply typed context calculus with first-class environments. In *5th International Symposium on Functional and Logic Programming (FLOPS 2001)*, LNCS 2024, pages 359–374, 2001.

[SSKI03] M. Sato, T. Sakurai, Y. Kameyama, and A. Igarashi. Calculi of meta-variables. In *Computer Science Logic and 8th Kurt Gödel Colloquium (CSL'03 & KGC)*, LNCS 2803, pages 484–497, 2003.

[Tal93] C. L. Talcott. A theory of binding structures and applications to rewriting. *Theoretical Computer Science*, 112(1):99–143, 1993.

[UPG03] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal unification. In *Computer Science Logic and 8th Kurt Gödel Colloquium (CSL'03 & KGC)*, LNCS 2803, pages 513–527, 2003.