

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

Makoto Hamana

Department of Computer Science,
Gunma University, Japan
2005, April, JAIST

What is the metalanguage in CS and Logic?

Example: $\lambda\mu_n$ -calculus

$$(\beta_{\rightarrow}) \quad (\lambda x^A.M)N = M[N/x]$$

$$(\eta_{\rightarrow}) \quad (\lambda x^A.Mx) = M \quad x \notin \text{FV}(M)$$

$$(\beta_{\wedge}) \quad \pi_i \langle M_1, M_2 \rangle = M_i$$

$$(\eta_{\mu}) \quad \mu \alpha^A. [\alpha]M = M \quad \alpha \notin \text{FV}(M)$$

$$(\eta_{\vee}) \quad \mu(\alpha^A, \beta^B). [\alpha, \beta]M = M \quad \alpha, \beta \notin \text{FV}(M)$$

$$(\xi_{\rightarrow}) \quad (\mu \alpha^{A \rightarrow B}. M)N = \mu \beta^B. M[[\beta](-)N/[\alpha](-)]$$

...

Features of the metalanguage

- ▷ bindings
- ▷ object variables ... capture avoiding substitutions
- ▷ metavariables ... possibly capturing substitutions

HOAS approach (Higher-Order Abstract Syntax)

[Pfenning-Elliot'88, Miller'91, ...]

- ▷ Typed λ -calculus as a metalanguage
- ▷ Problem on induction

Abstract syntax with binding [LICS'99]

- ▷ Fiore-Plotkin-Turi Binding algebras, Σ -monoids
- ▷ Gabbay-Pitts FM-set theory
- ▷ Hofmann Tripos

Part I. Construction of free Σ -monoids

- ▷ Free construction
- ▷ monad

Part II. Analysis of the language of the free Σ -monoid

- ▷ metavariables
- ▷ λ -calculus extended with holes
- ▷ explicit environments

Strong relationship to the series of Sato et al.'s work

Framework: Categorical universal algebra

A Σ -monoid models a language with binding with **substitutions**.

$$(\lambda x.M)N = M[N/x]$$

Definition

Let Σ be a binding signature . A **Σ -monoid** consists of

- ▷ a monoid object $M = (M, \eta, \mu)$
in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$ with
- ▷ a Σ -binding algebra $\alpha : \Sigma M \rightarrow M$ such that

$$\begin{array}{ccc}
 \Sigma(M) \bullet M & \xrightarrow{st} & \Sigma(M \bullet M) \xrightarrow{\Sigma\mu_M} \Sigma M \\
 \alpha \bullet M \downarrow & & \downarrow \alpha \\
 M \bullet M & \xrightarrow{\mu_M} & M
 \end{array}$$

commutes. This defines the category **Σ -Mon**.

Theorem [FPT'99] $T_{\Sigma} \mathbf{V}$ is an initial Σ -monoid.

Free Σ -monoid

Q. Is there a **free** Σ -monoid? i.e.

for a given $X \in \mathbf{Set}^{\mathbb{F}}$,

a Σ -monoid $M_{\Sigma}X$ having **universality**

A. Yes. We give a construction M_{Σ} :

▷ $M_{\Sigma}X$ is a free Σ -monoid over X .

▷ the *construction* $M_{\Sigma} : \mathbf{Set}^{\mathbb{F}} \rightarrow \mathbf{Set}^{\mathbb{F}}$ is a monad.

How to give a free construction of Σ -monoid?

- ▷ $X \longmapsto T_\Sigma X$ gives a free Σ -algebra but not a free Σ -monoid.
- ▷ A **monoid** $M = (M, \mu, \eta)$ in a monoidal category $\mathcal{C} = (\mathcal{C}, \otimes, I)$ consists of
 - an object $M \in \mathcal{C}$,
 - a *unit* $\eta : I \rightarrow M$,
 - a *multiplication* $\mu : M \otimes M \rightarrow M$, andsatisfying unit and associative law diagrams.
- ▷ General construction of free monoid objects
Dubuc: “Free Monoids”, Journal of Algebra 29, 1974
 - rather abstract, and far from a “language”
- ▶ Another direction:
monoid in $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V}) \sim$ **operad** \rightsquigarrow **free multicategory**
“concrete description”

How construct a free monoid in $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$?

- ▷ A Σ -monoid is a **monoid** in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$.
- ▷ unit: $\mathbf{V} : \mathbb{F} \rightarrow \mathbf{Set}; \mathbf{V}(n) = n$
- ▷ monoidal product \bullet is a “substitution monoidal product” defined by, for $A, B \in \mathbf{Set}^{\mathbb{F}}$,

$$(A \bullet B)(n) \triangleq \left(\prod_{m \in \mathbb{N}} A(m) \times B(n)^m \right) / \approx$$

where \approx is the equivalence relation generated by

$$(t; u_{\rho_1}, \dots, u_{\rho_m}) \sim (A(\rho)(t); u_1, \dots, u_l)$$

for $\rho : m \rightarrow l \in \mathbb{F}$.

Construction of free Σ -monoid

- ▷ given $X \in \mathbf{Set}^{\mathbb{F}}$
- ▷ construct a Σ -monoid $M_{\Sigma}X$ by:
 - (1) construct the presheaf $M_{\Sigma}X \in \mathbf{Set}^{\mathbb{F}}$
 - (2) show $M_{\Sigma}X$ has a Σ -algebra structure
 - (3) construct a monoid $(M_{\Sigma}X, \nu, \beta)$ in $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$
 - (4) check Σ -monoid law
- ▷ show universality
- ▶ $M_{\Sigma}X$ is a free Σ -monoid.

Binding signature

$$f : \langle n_1, \dots, n_l \rangle \in \Sigma$$

f has l arguments and binds n_i variables in the i -th argument ($1 \leq i \leq l$).

e.g.

fun. sym.	term
-----------	------

$\lambda : \langle 1 \rangle$	$\lambda([1]\text{nam}(1))$ <i>or, simply</i> $\lambda(\text{nam}(1))$
-------------------------------	--

$@ : \langle 0, 0 \rangle$	$@(t, t)$
----------------------------	-----------

- ▷ Object variables: natural numbers $1, \dots, n$
- ▷ Object variable terms: $\text{nam}(i)$

Convention: a set n of object variables

$$n = \{1, \dots, n\}$$

(I) The presheaf $M_\Sigma X$

▷ given $X \in \mathbf{Set}^{\mathbb{F}}$

▷ construct the set $\bar{M}_\Sigma X(n)$ indexed by $n \in \mathbb{N}$ as follows:

$$\frac{i \in n}{\text{nam}(i) \in \bar{M}_\Sigma X(n)}$$

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

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

▷ axiom

$$[x] \langle t_{\rho 1}, \dots, t_{\rho l} \rangle \doteq [X(\rho)(x)] \langle t_1, \dots, t_m \rangle$$

i.e. define the equivalence relation for every $\rho : l \rightarrow m \in \mathbb{F}$.

▷ The presheaf $M_\Sigma X \in \mathbf{Set}^{\mathbb{F}}$ is

$$M_\Sigma X(n) \triangleq \bar{M}_\Sigma X(n) / \doteq$$

Notation

$$\lceil x \rceil \equiv \lceil x \rceil \langle 1, \dots, n \rangle$$

$$f(t_1, \dots, t_l) \equiv f(\lceil n+1, \dots, n+i \rceil t_1, \dots, \lceil n+1, \dots, n+l \rceil t_l)$$

because the binders are clear from the function symbol $f \in \Sigma$.

(III) Monoid

$(M_\Sigma X, \nu, \beta)$ is a monoid in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathbf{V})$.

▷ unit $\nu : \mathbf{V} \rightarrow M_\Sigma X \in \mathbf{Set}^{\mathbb{F}}$

$$\begin{aligned} \nu(n) : \mathbf{V}(n) &\longrightarrow M_\Sigma X(n) \\ i &\longmapsto \text{nam}(i) \end{aligned}$$

▷ multiplication $\beta : M_\Sigma X \bullet M_\Sigma X \rightarrow M_\Sigma X$

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

$$\beta(n)([x] \langle s_1, \dots, s_l \rangle; \vec{t}) = [x] \langle \beta(n)(s_1, \vec{t}), \dots, \beta(n)(s_l, \vec{t}) \rangle \quad (x \in X(l))$$

$$\beta(n)(\text{nam}(i); \vec{t}) = t_i$$

$$\beta(n)(f(s_1, \dots, s_l); \vec{t}) = f(\beta(n+i_1)(s_1; \vec{t}), \dots, \beta(n+i_l)(s_l; \vec{t}))$$

where $f : \langle i_1, \dots, i_l \rangle \in \Sigma$ and \vec{t} denotes t_1, \dots, t_m .

▷ monoid laws: proved by induction at each component

(IV) Σ -monoid

Show that the monoid $(M_\Sigma X, \nu, \beta)$ makes the following commutative:

$$\begin{array}{ccc}
 \Sigma(M_\Sigma X) \bullet M_\Sigma X & \xrightarrow{st} & \Sigma(M_\Sigma X \bullet M_\Sigma X) \xrightarrow{\Sigma\beta} \Sigma M_\Sigma X \\
 \alpha \bullet M_\Sigma X \downarrow & & \downarrow \alpha \\
 M_\Sigma X \bullet M_\Sigma X & \xrightarrow{\beta} & M_\Sigma X
 \end{array}$$

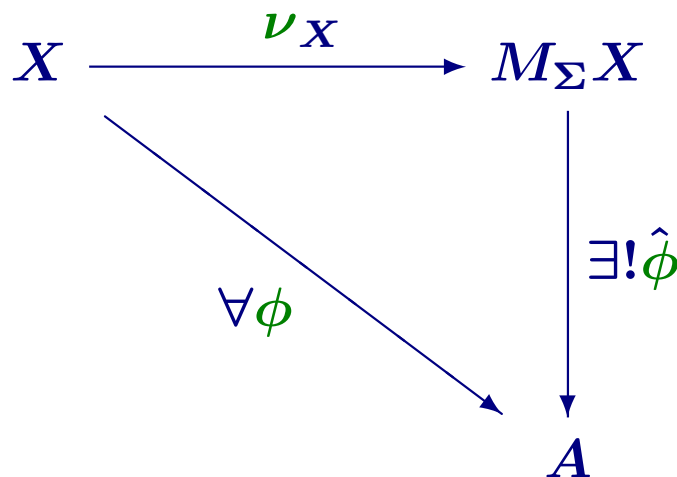
Proof sketch: Instantiating this diagram at $n \in \mathbb{F}$ and chasing an element, this eventually becomes the equality

$$\beta(n)(f(s_1, \dots, s_l); \vec{t}) = f(\beta(n + i_1)(s_1; \vec{t}), \dots, \beta(n + i_l)(s_l; \vec{t})).$$

Proposition $M_\Sigma X$ is a Σ -monoid.

Theorem (universality)

$M_\Sigma X$ is a **free Σ -monoid** over $X \in \mathbf{Set}^{\mathbb{F}}$, i.e.



any Σ -monoid A

$\hat{\phi} : \Sigma$ -monoid map

Proof sketch: define the “extension” $\hat{\phi}$ and use induction on terms.

This universality can be rephrased as an adjunction.

Corollary

“forgetful” functor $U : \Sigma\text{-Mon} \rightarrow \mathbf{Set}^{\mathbb{F}}$

$$\mathbf{Set}^{\mathbb{F}} \begin{array}{c} \xrightarrow{M_{\Sigma}} \\ \perp \\ \xleftarrow{U} \end{array} \Sigma\text{-Mon}$$

Corollary

The free Σ -monoid construction M_{Σ} is a monad on $\mathbf{Set}^{\mathbb{F}}$.

► Another substitution structure

Definition

A monad (M, η, μ) consists of

- ▷ a functor $M : \mathcal{C} \rightarrow \mathcal{C}$,
- ▷ a *unit* $\eta : \text{Id} \rightarrow M$,
- ▷ a *multiplication* $\mu : M \circ M \rightarrow M$,

satisfying unit and associative law diagrams.

M_Σ is a Monad

▷ functor: $M_\Sigma : \mathbf{Set}^{\mathbb{F}} \longrightarrow \mathbf{Set}^{\mathbb{F}}$ is defined by the construction

▷ unit: $\eta : \text{Id} \rightarrow M_\Sigma$

$$\begin{aligned} \eta_X(n) : X(n) &\longrightarrow M_\Sigma X(n) \\ x &\longmapsto [x] \end{aligned}$$

▷ multiplication $\mu : M_\Sigma \circ M_\Sigma \rightarrow M_\Sigma$

$$\begin{aligned} \mu_X(n) : M_\Sigma(M_\Sigma X)(n) &\longrightarrow M_\Sigma X(n) \\ \text{nam}(i) &\longmapsto \text{nam}(i) \\ f(t_1, \dots, t_l) &\longmapsto f(\mu_X(n + i_1)(t_1), \dots, \mu_X(n + i_l)(t_l)) \\ [s] \langle t_1, \dots, t_l \rangle &\longmapsto \beta(n)(s; \mu_X(n)(t_1), \dots, \mu_X(n)(t_l)) \end{aligned}$$

where $f : \langle i_1, \dots, i_l \rangle \in \Sigma$, and $s \in M_\Sigma X$, $t_1, \dots, t_l \in M_\Sigma M_\Sigma X$.

▷ monad laws: can be checked by induction.

The **associative law** is important \rightsquigarrow λ -calculus with holes

Part II.

Analysis of the language of the free Σ -monoid

Multiplications as Substitution operations

Two-level substitution structure by multiplications:

(1) $M_{\Sigma}X$ is a Σ -monoid:

β performs a substitution of object variables,
i.e. **capture avoiding** substitution.

(2) M_{Σ} is a monad:

μ performs substitution of metavariables
i.e. **possibly capturing** substitution.

Multiplications as Substitution operations

Remember: the **monad** multiplication μ

$$\mu_X(n) : M_\Sigma(M_\Sigma X)(n) \longrightarrow M_\Sigma X(n)$$

$$f(t_1, \dots, t_l) \longmapsto f(\mu_X(n + i_1)(t_1), \dots, \mu_X(n + i_l)(t_l))$$

$$\lceil s \rceil \longmapsto s$$

Performing the multiplication

$$\mu(f(\lceil t \rceil)) = f(t)$$

can be rewritten as “applying a substitution”

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

Multiplications as Substitution operations

We have

$$\mu(\lambda([a][\text{nam}(a)])) = \lambda([a]\text{nam}(a)).$$

This is rewritten as

$$\mu(\lambda([a][*])\{ * \mapsto \text{nam}(a) \}) = \lambda([a]\text{nam}(a)).$$

The object variable a is **captured** by the binder “[a]”. Hence,

- ▷ μ performs **possibly capturing** substitution, and
- ▷ [x] is considered as a **metavariable**.

cf.

Sato, Sakurai, Kameyama, Igarashi: *Calculi of Meta-variables*, CSL'03

How to give a presheaf X of metavariables?

- ▷ For a given $X \in \mathbf{Set}^{\mathbb{F}}$, we construct $M_{\Sigma}X$.
- ▷ First-order case: a given variable set. ★
- ▷ Guess: each $X(n) \cdots$ variable set
 - What is n ?
 - How to define the arrow part $X(\rho)$?

Hint: Staged variables

Plotkin: *Another Meta-Language for Programming with Bound Names Modulo Renaming*, 2000.

- ▷ The notion of **staged variables**
- ▷ A variable x has a “stage” $n = \{1, \dots, n\}$ of object variables, notation: $x : n$
- ▷ A judgment $x : n \vdash t : n$ ★
e.g. $x : 2 \vdash f(x, \text{nam}(2)) : 2$

Construction of a presheaf of metavariables (1)

- ▷ Define an \mathbb{N} -indexed set \mathbf{X} by setting

$$x \in \mathbf{X}(n) \text{ iff } x : n.$$

- ▷ The \mathbb{N} -indexed set \mathbf{X} is “almost” a presheaf, but it lacks the arrow part.
- ▷ Abstracting the problem: find the functor

$$(\hat{-}) : \mathbf{Set}^{\mathbb{N}} \longrightarrow \mathbf{Set}^{\mathbb{F}}.$$

- ▶ Kan extension

Theorem (Hom-tensor adjunction) [Mac Lane-Moerdijk'94]

Let \mathcal{E} be a cocomplete category and $A : \mathcal{C}^{\text{op}} \rightarrow \mathcal{E}$ a functor. Then,

$$\text{Set}^{\mathcal{C}} \begin{array}{c} \xrightarrow{L_A} \\ \perp \\ \xleftarrow{\mathcal{E}(A(=), -)} \end{array} \mathcal{E}$$

where $L_A = \text{Lan}_y A$.

i.e. “the left Kan extension of A along the Yoneda embedding y ”

Construction of a presheaf of metavariables:

▷ $\mathcal{C} = \mathbb{N}$, $\mathcal{E} = \mathbf{Set}^{\mathbb{F}}$

▷ $A : \mathbb{N}^{\text{op}} \rightarrow \mathbf{Set}^{\mathbb{F}}$; $A(i) = \mathbb{F}(i, -)$

$$(\hat{-}) = L_A : \mathbf{Set}^{\mathbb{N}} \longrightarrow \mathbf{Set}^{\mathbb{F}}.$$

Calculation:

$$\hat{X}(n) = \text{Lan}_y A (X)(n) = \left(\int^{k \in \mathbb{N}} X(k) \cdot \mathbb{F}(k, -) \right)(n) = \coprod_{k \in \mathbb{N}} X(k) \times \mathbb{F}(k, n)$$

What is \hat{X} ?

- ▷ How to give a presheaf of metavariables
 - Give \mathbb{N} -indexed set X of staged variables.
 - $\hat{X} \in \mathbf{Set}^{\mathbb{F}}$
 $\hat{X}(n) = \coprod_{k \in \mathbb{N}} X(k) \times \mathbb{F}(k, n)$.

▷ So, write

$$x^{\xi} \in \hat{X}(n) \quad \text{where } x : k \text{ (i.e. } x \in X(k)) \\ \text{and } \xi : k \rightarrow n \in \mathbb{F}.$$

i.e. “variables decorated with substitutions”.

► Useful?

Yes: relationship to λ -calculus extended with contextual holes

λ -calculus extended with contextual holes

[Talcott'93][Mason'96]

Sands: *Computing with Contexts: A simple approach*, 1998.

Hashimoto,Ohori: *A Typed Context Calculus*, 1998.

Sato et al.: *A Simply Typed Context Calculus with First-Class Environments*, '01.

- ▷ The syntactic construct “holes decorated with substitutions” is important to ensure

commutativity of β -reduction and hole-filling operation.

- ▷ Consider λ -calculus + \square :

$$(\lambda x. \square)y \rightarrow_{\beta} \square.$$

- ▷ Problem:

$$\begin{array}{ccc} (\lambda x. \square)I & \xrightarrow{\beta\text{-red.}} & \square \\ \text{fill with } x \downarrow & & \downarrow \text{fill with } x \\ (\lambda x. x)I & \xrightarrow{\beta\text{-red.}} & I \neq x \end{array}$$

- ▷ So, we need

$$(\lambda x. \square)I \rightarrow_{\beta} \square\{I/x\}.$$

Relationship between λ -calculus+ \square and M_Σ

$$\begin{array}{ccc}
 (\lambda x. \square)I & \xrightarrow{\beta\text{-red.}} & \square\{I/x\} \\
 \text{fill with } x \downarrow & & \downarrow \text{fill with } x \\
 (\lambda x. x)I & \xrightarrow{\beta\text{-red.}} & I
 \end{array}$$

Correspondence 1.

$\square\{I/x\}$... “a metavariable decorated with substitution”

Correspondence 2.

The associative law of the monad M_Σ implies

$$\begin{array}{ccc}
 (s; \mu(\vec{t})) \in MMX \bullet MMX & \xrightarrow{\beta} & MMX \\
 \mu \bullet \mu \downarrow & & \downarrow \mu \\
 MX \bullet MX & \xrightarrow{\beta} & MX
 \end{array}$$

In particular, take $(s; \mu(\vec{t})) = ([x]; I)$. \star

Summary

Definition A higher-order syntax with metavariables is specified by

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

Then, terms are expressed as the following BNF:

$$\begin{aligned} M_{\Sigma} \hat{X}(n) &\ni t ::= \text{nam}(i) \mid f(t_1, \dots, t_l) \mid [x] \langle t_1, \dots, t_l \rangle \\ X(n) &\ni x \end{aligned}$$

where $i \in n$, $\hat{X}(n) = \coprod_{k \in \mathbb{N}} X(k) \times \mathbb{F}(k, n)$.

Claim $[x] \langle t_1, \dots, t_l \rangle$ is an “explicit environment”.

cf. Sato, Sakurai, Burstall: *Explicit Environments*, TLCA'99

$\lceil x \rceil \langle t_1, \dots, t_l \rangle$ as Explicit environments

▷ $\lceil x \rceil \langle t_1, \dots, t_l \rangle$ informally means

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

▶ “suspended” substitution

▶ the substitution process only happens after instantiating x

▷ explicit substitution?

Remember:

$$\lceil s \rceil \langle t_1, \dots, t_l \rangle \xrightarrow{\mu_X(n)} \beta(n)(s; \mu_X(n)(t_1), \dots, \mu_X(n)(t_l))$$

$\lceil x \rceil \langle t_1, \dots, t_l \rangle$ as Explicit environments

- ▷ comparison: λ -calculus with named explicit substitutions $\lambda_{\mathbf{X}}$

$$t ::= a \mid tt \mid \lambda a.t \mid t\langle a := t \rangle$$

where a denotes an arbitrary $\lambda_{\mathbf{X}}$'s variable [Bloo-Rose'95].

- ▷ difference:

$$\lambda_{\mathbf{X}}: \quad a' \langle a := t \rangle \quad (a': \text{object variable})$$

$$M_{\Sigma \mathbf{X}}: \quad \lceil x \rceil \langle t_1, \dots, t_l \rangle \quad (\lceil x \rceil: \text{metavariable})$$

(regarded as $\square \langle t_1, \dots, t_l \rangle$)

An explicit environment ρ satisfy the following [Sato et al.'99]:

- (1) It has an operation $\llbracket t \rrbracket_{\rho}$, which evaluates a term t under an environment ρ .
- (2) It is a “first-class value”.

Note: Explicit substitutions have the property (1) but not (2).