Multiversal Polymorphic Algebraic Theories

— Syntax, Semantics, Translations, and Equational Logic —

Makoto Hamana
Dept. of Computer Science, Gunma University

Marcelo Fiore
Computer Laboratory, University of Cambridge

Abstract—We formalise and study the notion of *polymorphic algebraic theory*, as understood in the mathematical vernacular as a theory presented by equations between polymorphically-typed terms with both type and term variable binding.

The prototypical example of a polymorphic algebraic theory is System F, but our framework applies much more widely. The extra generality stems from a mathematical analysis that has led to a unified theory of polymorphic algebraic theories with the following ingredients:

- **polymorphic signatures** that specify arbitrary polymorphic operators (e.g. as in extended λ-calculi and algebraic theories of effects);
- **metavariables**, both for types and terms, that enable the generic description of meta-theories;
- **multiple type universes** that allow a notion of translation between theories that is parametric over possibly different type universes;
- **polymorphic structures** that provide a general notion of algebraic model (including the PL-category semantics of System F); and
- a **Polymorphic Equational Logic** that constitutes a sound and complete logical framework for equational reasoning.

Our work is semantically driven, being based on a hierarchical two-levelled algebraic modelling of abstract syntax with variable binding. As such, the development requires a sophisticated blend of mathematical tools: presheaf categories, the Grothendieck construction, discrete generalised polynomial functors, and aspects of categorical universal algebra.

1. Introduction

The notion of polymorphic types is one of the most remarkable inventions in programming language theory. Starting from the polymorphic λ-calculus of Girard and Reynolds and succeeding to Milner's striking application to functional programming [25], the theory of polymorphism has deepened, and the notion of polymorphism has spread everywhere. Not only functional programming languages, but also various systems have been extended to cope with polymorphism, such as π-calculus [27] and XML [**?**]. Even for object-oriented programming languages (Java, C++, etc.), a notion of polymorphism has been incorporated and regarded as a key feature of generic programming. These applications illustrate that polymorphism is not necessarily based on the λ-calculus.

Then, ***what are polymorphic types?***

This paper aims to establish an algebraic framework for analysing and reasoning about various polymorphic systems generally. We formalise and study the notion of *polymorphic algebraic theory*, as a theory presented by equations between polymorphically-typed terms.

Our approach is not based on a specific polymorphic λ-calculus. It is more general. We aim at capturing various polymorphic systems including extended λ-calculi as *particular examples* of our polymorphic algebraic theory. Our basis is the algebraic model of abstract syntax with variable binding in a presheaf category [10], [**?**].

To give necessary background for the present work, we first review this model and its subsequent developments.

I. Abstract syntax and variable binding. The aim is to model syntax involving variable binding. A typical example is the syntax for untyped λ-terms:

$$\frac{}{x_1,\ldots,x_n \vdash x_i} \qquad \frac{x_1,\ldots,x_n \vdash t \quad x_1,\ldots,x_n \vdash s}{x_1,\ldots,x_n \vdash t@s}$$

$$\frac{x_1,\ldots,x_n,x_{n+1} \vdash t}{x_1,\ldots,x_n \vdash \lambda(x_{n+1}.t)}$$

This is an abstract syntax generated by three constructors, i.e. the variable former, the application $@$, and the abstraction λ. The point is that the variable former is unary and $@$ is a binary function symbol, but λ is not merely a unary function symbol. It also makes the variable x_{n+1} bound and decreases the context. In order to model this phenomenon of variable binding generally (not only for λ-terms), Fiore, Plotkin and Turi took the presheaf category $\mathbf{Set}^{\mathbb{F}}$ to be the universe of discourse, where $\mathbb{F}$ is the category which has finite cardinals $n = \{1,\ldots,n\}$ (n is possibly 0) as objects, and all functions between them as arrows $n \to n'$. This is the category of object variables (regarded as contexts) by the method of de Bruijn index/level (i.e. natural numbers) and their renamings. A main result in [10] is that abstract syntax with variable binding is precisely characterised as the initial algebra of suitable endofunctor modelling a signature (e.g. for λ-terms).

For example, the signature endofunctor Σ_λ on $\mathbf{Set}^{\mathbb{F}}$ for abstract syntax of λ-terms is defined by $\Sigma_\lambda(A) = \mathrm{V} + A \times A + \delta A$ where each summand corresponds to the arity of constructor. Here the functor $\delta : \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ for context extension is $(\delta A)(n) = A(n+1)$, and the presheaf $\mathrm{V} \in \mathbf{Set}^{\mathbb{F}}$ of variables is $\mathrm{V}(n) = \{1,\ldots,n\}$.

For an endofunctor Σ, a *Σ-algebra* is a pair (A, α) consisting of a presheaf A and a map $\alpha : \Sigma A \to A$, called an *algebra structure*. The initial Σ_λ-algebra (Λ, in) can be constructed inductively as the presheaf Λ of all λ-terms modulo α-equivalence. The initial algebra explains more directly why presheaves are suited to model syntax with binding, namely

$$\text{a judgment } n \vdash t \quad \text{ is modelled as } \quad t \in \Lambda(n),$$

and renaming of free variables $\rho : n \to n'$ in a λ-term is modelled by the presheaf action $\Lambda(\rho) : \Lambda(n) \to \Lambda(n')$. This is a generic way of modelling abstract syntax with binding with respect to a signature functor Σ. However, the method is limited to modelling of *object-level* abstract syntax.

II. Object and meta variables. What is meant by "object-level" is the following. For example, consider the λ-term

$$\lambda x.My$$

in a certain mathematical context. Here "x" and "y" are λ-calculus variables (i.e. *object-level variables*, because now the λ-calculus is the object system), while at the level of text, "M" is a meta-level variable. When developing a theory of λ-calculus, we always use both object and meta variables. There is also an important difference between metavariables and object variables in view of substitutions. If we substitute a term xx for the object variable y in the term above, the result is not simply a textual substitution, i.e. $(\lambda x.My)[y := xx] = (\lambda x'.My)[y := xx] = \lambda x'.M(xx)$ where $x \neq x'$, because of the capture-avoiding substitution in the λ-calculus. But if we substitute a term xx for the metavariable M, we have

$$(\lambda x.My)\{M \mapsto xx\} = \lambda x.(xx)y \tag{1}$$

Although the object variables x is *captured* by the binder, it is usually allowed at the meta-level. Viewing these phenomena at the extra meta-level viewpoint, these two classes of variables are classified by the distinction of substitutions: capture-avoiding and possibly capturing.

III. Free Σ-monoids. Not only object-level abstract syntax, how can metavariables and the distinction of substitutions for object and meta variables be incorporated with the algebraic model of syntax with binding? This problem was explored in [16], [5] and a clear answer has been obtained.

A *Σ-monoid* [10] is a Σ-algebra (A, α) with a monoid structure

$$\mathrm{V} \xrightarrow{\nu} A \xleftarrow{\mu} A \bullet A$$

that is compatible with the algebra structure (i.e. $\mu \circ (\alpha \bullet \mathrm{id}) = \alpha \circ (\Sigma\mu) \circ \mathsf{strength}$) in the monoidal category $(\mathbf{Set}^{\mathbb{F}}, \bullet, \mathrm{V})$. The unit ν models the variable former, and the multiplication μ models *substitution for object variables*, where the monoidal product $\bullet$ gives precisely the arity of substitution. Monoids in the monoidal category are known as *operads* that play a crucial role in various mathematical structures (such as topology and weak ω-categories [**?**]).

For our purpose, we use *free Σ-monoids* generated by arbitrary presheaf X, where generators X is regarded as the *presheaf of metavariables*. Importantly, a free Σ-monoid over $X \in \mathbf{Set}^{\mathbb{F}}$, denoted by $\mathcal{M}X$, is constructed *inductively* as an initial $(\mathrm{V} + \Sigma + X \bullet -)$-algebra, which gives the language involving binding and *metavariables*. This characterisation shows that $\mathrm{V} + \Sigma$ models syntax with binding (as in §1-I) and the functor $X \bullet -$ models the syntactic construct of metavariables represented as a term

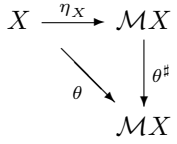
$$\mathrm{M}[t_1, \ldots, t_n]$$

where $\mathrm{M} \in X(n)$ is a *metavariable* and the index n, called also *arity*, which denotes possible free variables $1, \ldots, n$ appearing in a term substituted for M. Terms $t_1, \ldots, t_n$ are replacements of these free variables after instantiating M.

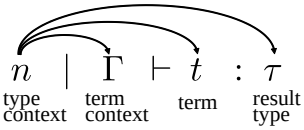
For example, we can write the above example (1) using the language of free Σ-monoid as

$$\theta^\sharp(\lambda(x.\ \mathrm{M}[x] \mathbin{@} y)) = \lambda(x.\ (x \mathbin{@} x) \mathbin{@} y) \tag{2}$$

where θ maps the metavariable M as $\theta(\mathrm{M}) = 1@1$ (where 1 is a free variable). The freeness of $\mathcal{M}X$ states that given θ, there uniquely exists the extension $\theta^\sharp$ to Σ-monoid morphism that makes the right diagram commute. Here, the notion of substitution of metavariables appears. This is syntactically understood as that θ is an *assignment for meta-substitution* and $\theta^\sharp$ is the corresponding meta-substitution on terms involving metavariables X. For (2), we put the metavariable $\mathrm{M} \in X(1)$. Σ-monoids model the syntax and semantics algebraically precisely. Therefore, it has various applications, such as second-order equational logic [8]. However, it is limited to *untyped* and *simply-typed* settings.



IV. Polymorphic abstract syntax. A previous work [20] tackled its extension to the case of polymorphic typed abstract syntax. A crucial departure from the case of $\mathbf{Set}^{\mathbb{F}}$ is the need for a finer index structure on presheaves to model polymorphic terms. Well-typed terms in a polymorphic system (e.g. System F) are formulated using judgments of the following form.



(⧫)

The arrows indicate that a type variable in $n = \{1, \ldots, n\}$ can appear in all other parts. Clearly, this dependence is more complicated than the untyped case $n \vdash t$ (where only n is a required index; therefore $\mathbf{Set}^{\mathbb{F}}$ suffices). To model this, the work [20] clarified the category $\mathsf{G}U$ of contexts and result type defined as

$$\boxed{\textstyle\int^{n \in \mathbb{F}} \mathbb{F}{\downarrow}U(n) \times U(n)} \overset{\text{def}}{=\!=} (\ n \mid \quad \Gamma \quad \vdash \quad \tau \ ) \in \mathsf{G}U$$

and a new presheaf category $\mathbf{Set}^{\mathsf{G}U}$ is suitable to model polymorphic typed *object-level* syntax. Here U is a "type universe", i.e., $U(n)$ is the set of all types, and $\mathbb{F}{\downarrow}U(n)$ is the category of term contexts, both under a type context n. As an element in $\mathsf{G}U$ shown above indicates, the Grothendieck construction "$\int$" is the key to capture the dependence (⧫).

V. This paper. We further proceed to develop polymorphic algebraic theory founded on these earlier works. We incorporate the notions of metavariables and Σ-monoids reviewed in §1-II and 1-III to the polymorphic setting reviewed in §1-IV. We allow metavariables in both types and terms. We explain why it is necessary.

For example, consider the β-axiom of typed λ-calculus:

$$\Gamma \vdash (\lambda x^\sigma.M)\, N \ = \ N[x := N] : \tau$$

Is this a *single* axiom? We usually think so, but this is actually a *schema* of axioms because M and N are metavariables, and σ and τ are *metavariables for types*. Therefore, this should be regarded as a representation of a *family* of axioms of the object system, indexed by all possible object terms M, N and object types σ, τ. This process reflects the formulation of a signature. If one follows this line, then the λ-abstraction should also be regarded as *a family* $\{\lambda_{\sigma,\tau} : (\sigma)\tau \to \sigma \Rightarrow$

$\tau \mid$ object types $\sigma, \tau\}$ of symbols. But we usually keep σ and τ meta-level types (which we call *meta-types* in this paper), and consider a single $\lambda_{\sigma,\tau}$ to develop a theory.

Although this distinction between meta and object usually receives little attention when developing a theory, this viewpoint is seriously needed when we develop a meta-theory of a theory or a mechanised formalisation (such as in Coq [2]). In such a case, we must formalise all ingredients of a theory including the meta-level treatment.

We precisely formulate the meta and object variable distinction. In polymorphic algebraic theory, the λ-abstraction is formalised as a *single* function symbol specified by

$$\mathtt{S} : *,\ \mathtt{T} : * \ \triangleright\ \ \mathsf{abs} : (\mathtt{S})\mathtt{T} \to \mathtt{S} \Rightarrow \mathtt{T}$$

where S and T are metavariables for types, and where $\triangleright$ separates a metavariable context and the arity information of a function symbol using *meta-types* (e.g. $\mathtt{S} \Rightarrow \mathtt{T}$ is a meta-type). This leads to an important clarification. Let S be the set of all metavariables for types. Up to this point, we have encountered *two* type universes for formalising a theory:

(i) The universe of all meta-types (denoted by $\mathcal{M}\underline{S}$).
(ii) The universe of all object types (denoted by $\mathcal{M}0$).

Clearly, the universe of meta-types must have almost the same structure as that of object types. What is an "almost the same structure" precisely? Moreover, when we consider a semantics of a theory, the semantics of the type universe must also have such an "almost the same structure". This ultimately becomes the question posed at the beginning of the Introduction: "*what are polymorphic types?*" Our answer is

a universe of polymorphic types should be a Σ*-monoid*,

where Σ is a signature of type constructors. Note that $\mathcal{M}\underline{S}$ is a free and $\mathcal{M}0$ is an initial Σ-monoid. Therefore, the theory naturally requires to deal with such multiple universes. In this sense, we call our theory *multiversal*.

VI. Organisation. This paper is organised as follows. After providing preliminaries in next section, we define a polymorphic signature in Section 3 and the corresponding signature functor in Section 4. We then axiomatise type-in-term substitution in Section 5. We further define polymorphic structure as a general algebraic model in Section 6. In Section 7, we define a Polymorphic Equational Logic and show its soundness and completeness. Finally, in Section 8, we exemplify how polymorphic algebraic theory can specify concrete theories.

VII. Future work. Various directions for further work are possible. One promising direction is to apply our theory to mechanised formalisation. Because a correspondence exists between a generalised polynomial functor and an inductively defined dependent type [12], [**?**], our characterisation is connected directly to a method to formalise syntax and semantics of various polymorphic systems in proof assistances based on dependent type theory. Actually, the strongly typed representation of System F syntax in Coq [2] can be obtained as an instance of our algebraic characterisation.

The type structure of our theory will be extended to allow polymorphic kinds. Preliminary results [20] indicate that this direction will be a natural extension of the present framework.

Polymorphic algebraic theory of effects, which is only exemplified in this paper, should be worked out in more detail.

An algebraic theory for the π-calculus has been given [31], [**?**]. Along this line, to seek an algebraic theory for the polymorphic π-calculus [27] using polymorphic algebraic theory is a challenging problem.

2. Preliminaries

Generalised polynomial functors on presheaves. One of the central technical tools in our development is generalised polynomial functors on presheaves [6] developed as a further generalisation of dependent polynomial functors [12].

Let $f : \mathbb{C} \to \mathbb{D}$ be a functor between small categories. For the functor $f^* = - \circ f$ there are adjunctions [22], [15]

$$f_! \ \dashv\ f^* \ \dashv\ f_* \ :\ \mathbf{Set}^{\mathbb{C}} \to \mathbf{Set}^{\mathbb{D}}$$

where for $f_!$ and f_* are left and right Kan extensions along f respectively. A *polynomial* is a diagram P in **Cat**

$$\mathbb{A} \xleftarrow{\ s\ } \mathbb{I} \xrightarrow{\ a\ } \mathbb{J} \xrightarrow{\ t\ } \mathbb{B}.$$

The *generalised polynomial functor* induced by P is

$$F_P \stackrel{def}{=} t_!\ a_*\ s^* \ :\ \mathbf{Set}^{\mathbb{A}} \to \mathbf{Set}^{\mathbb{B}}. \tag{3}$$

A polynomial diagram is *discrete* when a is given as a sum of the codiagonal $\nabla_L = [\mathrm{id}]_{l \in L} : L \cdot \mathbb{C} \to \mathbb{C}$ for finite L.

Why this is useful is that an F_P-algebra A can specify an operation on a presheaf A having complex form of arity and indices at the source and the target. By adjointness, there exists the bijective correspondences shown at the right, where the final form is the reason of the label names of the functors at the above polynomial diagram.

$$\frac{F_P A \longrightarrow A}{t_!\, a_*\, s^* A \longrightarrow A}\qquad \frac{a_*\, s^* A \longrightarrow t^* A}{a_* A(s-) \longrightarrow A(t-)}$$

Namely, s is used for computing the **s**ource index, a is used for computing the **a**rity, and t is for the **t**arget index. Another crucial fact on polynomials for our purpose is that any generalised discrete polynomial functor F_P admits *inductive construction of initial F_P-algebra* ([6] Prop. 5.1). Because of the expressiveness and constructive nature, we will employ this polynomial functor formulation to model a polymorphic signature (§4), and a type-in-term substitution (§5).

Convention on α-equivalence. In this paper, we use implicitly the method of de Bruijn levels [3] for representing bound and free variables in a term (and a type, a judgment, etc.). Any term appearing in this paper hereafter is automatically normalised to a de Bruijn level α-normal form suitably. For example, $\mathsf{abs}(x.t)$ to mean $\mathsf{abs}(1.t)$. $\forall(\alpha.\mathtt{T}[\alpha])$ to mean $\forall(1.\mathtt{T}[1])$ $f(x_1.x_2.t)$ to mean $f(1.2.t)$. $\alpha_1, \ldots, \alpha_n \vdash \forall\alpha_{n+1}.\,\tau$ to mean $n \vdash \forall(n+1.\,\tau)$.

Notational convention. We use the vector notation $\overrightarrow{a}$ for a sequence $a_1, \cdots, a_l$, and $|\overrightarrow{a}|$ for its length. We usually write the indexing of a natural transformation φ as φ_i or $\varphi(i)$, but we omit often the indices if they are inferable from contexts. We may write indices when we want to emphasise them.

3. POLYMORPHIC SIGNATURE

We start with the prototypical example of System F to illustrate how our notion of polymorphic signature can specify type and term structures.

Example 3.1 (**System F**) The polymorphic signature $\Sigma_{\mathrm{F}} = (\Sigma_{\mathrm{F}}^{\mathsf{Ty}}, \Sigma_{\mathrm{F}}^{\mathsf{Tm}})$ is given as follows: the signature $\Sigma_{\mathrm{F}}^{\mathsf{Ty}}$ for types is $\{\mathsf{b} : *,\quad \Rightarrow\colon *, * \to *,\quad \forall : \langle * \rangle * \to *\}$, which specifies type constructors. The signature $\Sigma_{\mathrm{F}}^{\mathsf{Tm}}$ for terms is

$$\begin{array}{llll}
\mathrm{S}, \mathrm{T} : * & \triangleright\ \mathsf{abs} & : (\mathrm{S})\mathrm{T} & \to\ \mathrm{S} \Rightarrow \mathrm{T} \\
\mathrm{S}, \mathrm{T} : * & \triangleright\ \mathsf{app} & : \mathrm{S} \Rightarrow \mathrm{T}, \mathrm{S} & \to\ \mathrm{T} \\
\mathrm{T} : \langle * \rangle * & \triangleright\ \mathsf{tabs} & : \langle \alpha \rangle \mathrm{T}[\alpha] & \to\ \forall(\alpha.\mathrm{T}[\alpha]) \\
\mathrm{S} : *,\ \mathrm{T} : \langle * \rangle * & \triangleright\ \mathsf{tapp} & : \forall(\alpha.\mathrm{T}[\alpha]) & \to\ \mathrm{T}[\mathrm{S}]
\end{array}$$

Let us see how this signature faithfully encodes the ordinary typing rules.

$$\frac{\Xi \mid \Gamma, x : \sigma \vdash t : \tau}{\Xi \mid \Gamma \vdash \lambda x : \sigma.\, t\ : \sigma \Rightarrow \tau} \qquad \frac{\Xi, \alpha \mid \Gamma \vdash t : \tau}{\Xi \mid \Gamma \vdash \Lambda\alpha.\, t : \forall\alpha.\, \tau}$$

The arity of abs, $(\mathrm{S})\mathrm{T}$, represents that the upper judgment of the abstraction rule (the above left) has a term context extended with an extra $x\colon \sigma$ and the result type τ. The informal metavariables σ and τ are formalised as formal metavariables $\mathrm{S}, \mathrm{T} : *$ for types in the signature. The target of abs, $\mathrm{S} \Rightarrow \mathrm{T}$, represents the target type of the lower judgment.

The arity of tabs, $\langle \alpha \rangle \mathrm{T}[\alpha]$, represents that the upper judgment of the type abstraction rule has a type context extended with an extra α and the result type τ, where τ may use α (hence written as $\mathrm{T}[\alpha]$). The declaration $\mathrm{T} : \langle * \rangle *$ represents this possible containment. A function symbol declaration is intended to be instantiated to concrete cases. For example, $\mathsf{abs} : (\mathrm{S})\mathrm{T} \to \mathrm{S} \Rightarrow \mathrm{T}$ is instantiated to $\mathsf{abs}_\theta : (\mathsf{nat})\mathsf{bool} \to \mathsf{nat} \Rightarrow \mathsf{bool}$, where $\theta = \{\mathrm{S} \mapsto \mathsf{nat}, \mathrm{T} \mapsto \mathsf{bool}\}$. Formal definitions are as follows.

Definition 3.2 A *metavariable for types* $(\mathrm{S}, \mathrm{T}, \ldots)$ of arity n is declared as $\mathrm{S} : \langle *^n \rangle *$. In the case $n = 0$, $\langle *^0 \rangle$ is omitted. A set S of metavariables forms an $\mathbb{N}$-indexed set S by $\mathrm{S} \in S(n) \ \Leftrightarrow\ \big(\mathrm{S} : \langle *^n \rangle * \big) \in S$. Given a set S of metavariables, we define $\underline{S} \in \mathbf{Set}^{\mathbb{F}}$ as $\underline{S}(n) = \coprod_{k \in \mathbb{N}} S(k) \times \mathbb{F}(k, n)$.

Definition 3.3 A *polymorphic signature* $\Sigma = (\Sigma^{\mathsf{Ty}}, \Sigma^{\mathsf{Tm}})$ consists of the following data.

- A signature Σ^{Ty} for types is a set of type constructors given by the form $c : \langle *^{n_1} \rangle *, \cdots, \langle *^{n_l} \rangle * \to *$ meaning that it has l arguments binding n_i type variables in the i-th argument $(1 \le i \le l)$.
- A signature Σ^{Tm} for terms is a set of function symbols given by the form

$$S \ \triangleright\ f : \langle k_1 \rangle(\overrightarrow{\sigma_1})\tau_1, \ldots, \langle k_l \rangle(\overrightarrow{\sigma_l})\tau_l \to \tau$$

  where $S \triangleright k_i \vdash \overrightarrow{\sigma_i}$, $S \triangleright k_i \vdash \tau_i$ $(1 \le i \le l)$, $S \triangleright 0 \vdash \tau$ (this judgment notation is defined below), meaning that f has l arguments and binds, in the i-th argument $(1 \le i \le l)$, k_i type variables and $|\overrightarrow{\sigma_i}|$ variables.

Since Σ^{Ty} is a binding signature [10], it induces the corresponding signature functor Σ^{Ty} on $\mathbf{Set}^{\mathbb{F}}$ (cf. §1-I, 1-III). Let $\mathcal{M}\underline{S}$ denote the free Σ^{Ty}-monoid over $\underline{S}$, which is the presheaf of all meta-types using S. We write $S \triangleright n \vdash \tau$ for $\tau \in \mathcal{M}\underline{S}(n)$. For a metavariable $S \triangleright 0 \vdash \mathrm{T}$, $\mathrm{T}[\,]$ of any stage n will be abbreviated as simply T.

Definition 3.4 A *type universe* $U = (U, \alpha, \nu^U, \mu^U)$ is a Σ^{Ty}-monoid. Throughout the paper, we denote by U a Σ^{Ty}-monoid.

A typical example of U is $\mathcal{M}0$ (all object types) or $\mathcal{M}\underline{T}$ (all meta-types using metavariables T). For example, $\mathcal{M}0_{\exists}$ (all existential types) will be used in Example 8.2. A non-syntactic type universe U appears in the PL-category semantics (Example 8.3).

Definition 3.5 Let S be a set of metavariables and U a Σ^{Ty}-monoid. An assignment $\theta : S \rightsquigarrow_k U$ for meta-substitution is an $\mathbb{N}$-indexed function $\theta : S \to |\delta^k U|$, where k is "the number of additional possible free variables" in the results. It maps as $S(n) \ni \mathrm{S} \overset{\theta_n}{\longmapsto} a \in U(n+k)$.

The meta-substitution of a type $S \triangleright n \vdash \tau$ by an assignment $\theta : S \rightsquigarrow_k U$, denoted by $\theta_n^\sharp(\tau)$, is defined by

$$\theta_n^\sharp(\alpha) = \nu_n^U(\alpha);\ \theta_n^\sharp(c(\overrightarrow{\alpha_1}.t_1, \ldots, \overrightarrow{\alpha_l}.t_l)) = c_n^U(\theta^\sharp_{n+|\overrightarrow{\alpha_1}|}(\tau_1), \ldots)$$
$$\theta_n^\sharp(\mathrm{S}[\overrightarrow{\tau}]) = \mu_n^U(\theta_m(\mathrm{S}); \nu_n^U(1), \ldots, \nu_n^U(k), \theta_n^\sharp(\tau_1), \ldots, \theta_n^\sharp(\tau_m))$$

where $\mathrm{Z} \in S(m)$. This θ defines a natural transformation $\widehat{\theta} : \underline{S} \to \delta^k U$, which induces the unique Σ^{Ty}-monoid morphism $\widehat{\theta}^\sharp : \mathcal{M}\underline{S} \to \delta^k U$ that extends θ by freeness of $\mathcal{M}\underline{S}$ (cf. §1-III). We will also write $\tau\theta$ for $\theta_n^\sharp(\tau)$.

Using a meta-substitution θ, a function symbol is instantiated to $f_\theta : \langle k_1 \rangle(\overrightarrow{\sigma_1}\theta)\tau_1\theta, \ldots, \langle k_l \rangle(\overrightarrow{\sigma_l}\theta)\tau_l\theta \to \tau\theta$.

Example 3.6 (**Polymorphic FPC [23]**) The signature Σ^{Ty} for types is $\Sigma_{\mathrm{F}}^{\mathsf{Ty}}$ plus $\{+, \times : *, * \to *\ ,\quad \mu : \langle * \rangle * \to *\}$. An excerpt of the signature for terms is

$$\begin{array}{llll}
\mathrm{T}_1, \mathrm{T}_2, \mathrm{T} : * & \triangleright\ \mathsf{case} & : \mathrm{T}_1 + \mathrm{T}_2, (\mathrm{T}_1)\mathrm{T}, (\mathrm{T}_2)\mathrm{T} & \to\ \mathrm{T} \\
\mathrm{T} : \langle * \rangle * & \triangleright\ \mathsf{intro} & : \mathrm{T}[\mu(\alpha.\mathrm{T}[\alpha])] & \to\ \mu(\alpha.\mathrm{T}[\alpha]) \\
\mathrm{T} : \langle * \rangle * & \triangleright\ \mathsf{elim} & : \mu(\alpha.\mathrm{T}[\alpha]) & \to\ \mathrm{T}[\mu(\alpha.\mathrm{T}[\alpha])]
\end{array}$$

An important point is that the arities and targets are written in the language of free Σ^{Ty}-monoid [16], [5] (cf. §1-III), not in an informal meta-language.

Example 3.7 (**Existential λ-calculus [11]**) We excerpt key two rules.

$$\frac{\Xi \mid \Gamma \vdash s : \sigma\{\alpha := \tau\}}{\Xi \mid \Gamma \vdash \langle \tau, s \rangle : \exists(\alpha.\sigma)} \qquad \frac{\Xi \mid \Gamma \vdash s : \exists(\alpha.\sigma) \quad \Xi, \alpha \mid \Gamma, x\colon \sigma \vdash t\colon \tau}{\Xi \mid \Gamma \vdash \mathsf{unpack}\ s\ \mathsf{as}\ \langle \alpha, x \rangle\ \mathsf{in}\ t : \tau}$$

In the unpack rule, $\alpha \notin \mathrm{FV}(\Gamma, \tau)$. The signature for types is

$$\bot : *\ ,\quad \neg : * \to *\ ,\quad \wedge : *, * \to *\ ,\quad \exists : \langle * \rangle * \to *$$

An excerpt of the signature for terms is

$$\begin{array}{lll}
\mathrm{S} : \langle * \rangle *,\ \mathrm{T} : * \ \triangleright\ \mathsf{pack} & : \mathrm{S}[\mathrm{T}] & \to\ \exists(\alpha.\mathrm{S}[\alpha]) \\
\mathrm{S} : \langle * \rangle *,\ \mathrm{T} : * \ \triangleright\ \mathsf{unpack} & : \exists(\alpha.\mathrm{S}[\alpha]), \langle \alpha \rangle \big(\mathrm{S}[\alpha]\big)\mathrm{T} & \to\ \mathrm{T}
\end{array}$$

Note that the second argument of unpack is specified under a context $\mathsf{T} : *$ thereby enforcing the side condition of the rule that α does not appear in τ.

As these examples show, our notion of (meta)types can specify not only universally quantified types, but also recursive and existential types. In this paper, we formalise polymorphic types in a broader sense, i.e., as "*variable types*" as Girard called [14], for which we mean that types involve (meta)variables and variable binding.

Example 3.8 (**Global state**) A non λ-calculus example. The signature for an algebraic theory of global state [28] is given as follows. The type signature consists of the types for locations L, expressions E and value types as $\Sigma^{\mathsf{Ty}} = \{\mathsf{L}, \mathsf{E}, \mathsf{Nat}, \mathsf{Bool} : *\}$. The term signature consists of two operations

$$\mathsf{V} : * \triangleright \mathsf{lookup} : \mathsf{L}, (\mathsf{V})\mathsf{E} \to \mathsf{E}$$
$$\mathsf{V} : * \triangleright \mathsf{update} : \mathsf{V}, \mathsf{L}, \mathsf{E} \to \mathsf{E}$$

A point is that these operations are parameterised by the metavariable V for types. In the original treatment [28], this parameterisation is only informal-level, but here we can give a formal signature in our polymorphic algebraic theory. An algebraic theory constructed by this signature is first-order with variable binding, and no higher-order types and no λ-calculus are needed, but polymorphic. This style of formalisation may be useful for finer and more flexible algebraic characterisation of effects along the line of [28], [**?**].

4. Signature Functor

Given a polymorphic signature Σ, we associate a corresponding signature functor for algebra of functor.

4.1. Polymorphic contexts

For a set T, we denote by $\mathbb{F}\downarrow T$ a comma category consisting of objects $\Gamma : n \to T$ (for contexts) and arrows $\rho : \Gamma \to \Gamma'$ (for renaming), which are functions $\rho : n \to n'$ such that $\Gamma = \Gamma' \circ \rho$.

A Σ^{Ty}*-monoid morphism* is a morphism of $\mathbf{Set}^{\mathbb{F}}$ that is both Σ^{Ty}-algebra homomorphism and monoid morphism. the category Σ^{Ty}-**Mon** consists of Σ^{Ty}-monoids and Σ^{Ty}-monoid morphisms.

Given a functor $\mathcal{F} : \mathbb{C} \to \mathbf{Cat}$, the *Grothendieck construction* [15] of $\mathcal{F}$ is a category $\int^{c\in\mathbb{C}} \mathcal{F}c$ (or simply $\int \mathcal{F}$) with objects (I, A) where $I \in \mathbb{C}$ and $A \in \mathcal{F}(I)$, and arrows $(u, \gamma) : (I, A) \to (J, B)$ where $u : I \to J$ in $\mathbb{C}$ and $\gamma : \mathcal{F}(u)(A) \to B$ in $\mathcal{F}(J)$.

Definition 4.1 The functor $\mathsf{G} : \Sigma^{\mathsf{Ty}}$-**Mon** $\to$ **Cat** is defined by

$$\mathsf{G}(X) \stackrel{def}{=} \int^{n\in\mathbb{F}} \mathbb{F}\downarrow X(n) \times X(n).$$

The *category* $\mathsf{G}U$ *for contexts and result types* (cf. §1-IV) has
- objects $(n \mid \Gamma \vdash \tau)$, where $n \in \mathbb{F}$, $\Gamma \in \mathbb{F}\downarrow Un$, $\tau \in U(n)$,
- arrows $(\rho, \pi) : (m \mid \Gamma \vdash \tau) \to (n \mid \Delta \vdash \sigma)$, where $\rho : m \to n$ in $\mathbb{F}$ such that $U(\rho)(\tau) = \sigma$, and $\pi : (\mathbb{F}\downarrow U\rho)(\Gamma) \to \Delta$ in $\mathbb{F}\downarrow(U(n))$.

The *category* $\int \mathbb{F}\downarrow U$ *for contexts* consists of objects denoted by $(n|\Gamma)$ and arrows (ρ, π). It has coproducts given by $(m|\Gamma)+(n|\Delta) = (n+m \mid \iota^n(\Gamma), \iota^m(\Delta))$, where for $\mathsf{up}(k) : 0 \to k$ in $\mathbb{F}$, $\iota^k \stackrel{def}{=} \mathbb{F} \downarrow \delta^{\mathsf{up}(k)}U : \mathbb{F}\downarrow U \to \mathbb{F} \downarrow \delta^k U$. We define the context extension $\delta^{(n|\Delta)}A \stackrel{def}{=} A\big(- + n \mid =, \Delta \vdash \equiv\big)$.

4.2. Signature functor

We define the signature functor corresponding to a polymorphic signature Σ using a discrete polynomial. We start with analysing concrete cases. Consider the function symbol for the abstraction of System F (Example 3.1):

$$\mathsf{S} : *, \mathsf{T} : * \triangleright \mathsf{abs} : (\mathsf{S})\mathsf{T} \to \mathsf{S} \Rightarrow \mathsf{T}$$

The corresponding operation on an algebra A should be

$$\mathsf{abs}^A : A(n \mid \Gamma, \sigma \vdash \tau) \to A(n \mid \Gamma \vdash \sigma \Rightarrow \tau)$$

where $\sigma, \tau \in U(n)$. A type context n may also be changed as in the case of type abstraction:

$$\mathsf{tabs}^A : A(n+1 \mid \Gamma \vdash \tau) \to A(n \mid \Gamma \vdash \forall(\alpha.\tau))$$

To obtain these types of algebra operations from the specification of the arities of function symbols, we need
- to formulate the instantiation of meta-types in arities by types in U by meta-substitutions as $\theta = \{\mathsf{S} \mapsto \sigma, \mathsf{T} \mapsto \tau\}$,
- to give the indices (e.g. $(n+1 \mid \Gamma \vdash \tau)$) for A from arities.

This requires to use maps from the Grothendieck construction $\int \big(\mathbb{F}\downarrow U \times [\underline{S} \Rightarrow U]\big)$, which consists of contexts and meta-substitutions under arbitrary type context n. Here, we regard an exponential presheaf $[\underline{S} \Rightarrow U] \in \mathbf{Set}^{\mathbb{F}}$ as an indexed category in $\mathbf{Cat}^{\mathbb{F}}$ giving a discrete category $[\underline{S} \Rightarrow U](n) = \mathbf{Set}^{\mathbb{F}}(\underline{S}, \delta^n U)$, and $\theta \in [\underline{S} \Rightarrow U](n)$ gives a meta-substitution.

First we define the denotation of a meta-type. An element $\tau \in \mathcal{M}\underline{S}(n)$ bijective corresponds to a map $\hat{\tau} : \mathrm{V}^n \to \mathcal{M}\underline{S}$ [5]. Let $\varepsilon = \mathrm{id}^\sharp_A : \mathcal{M}U \to U$ be a map defined by freeness of $\mathcal{M}U$, and st a strength. Define

$$[\![\tau]\!]' : [\underline{S} \Rightarrow U] \times \mathrm{V}^n \xrightarrow{\mathrm{id}\times\hat{\tau}} [\underline{S} \Rightarrow U] \times \mathcal{M}\underline{S} \xrightarrow{\mathcal{M}\mathsf{evost}} \mathcal{M}U \xrightarrow{\varepsilon} U$$

We define $[\![S \triangleright k \vdash \tau]\!] : U^{\underline{S}} \to \delta^k U$ by the transpose of $[\![\tau]\!]'$.

Here we use the notion of a map of indexed categories [**?**] from $\mathcal{C} : \mathbb{C} \to \mathbf{Cat}$ to $\mathcal{D} : \mathbb{D} \to \mathbf{Cat}$, which is a pair (F, φ) consisting of a functor $F : \mathbb{C} \to \mathbb{D}$ and a natural transformation $\varphi : \mathcal{C} \to \mathcal{D}F$. A map of indexed categories yields a functor at the level of the Grothendieck constructions $\int(F, \varphi) : \int \mathcal{C} \to \int \mathcal{D}$ given by $(c \in \mathbb{C}, z \in \mathcal{C}c) \longmapsto (Fc \in \mathbb{D}, \varphi_c(z) \in \mathcal{D}Fc)$.

Definition 4.2 Suppose meta-types $S \triangleright k \vdash \overrightarrow{\sigma}$, $S \triangleright k \vdash \tau$. A *context extension*

$$\mathsf{ext}(k \mid \overrightarrow{\sigma} \vdash \tau) : \mathbb{F}\downarrow U \times [\underline{S} \Rightarrow U] \longrightarrow \mathbb{F}\downarrow U \times U$$

is a map of indexed category in $\mathbf{Cat}^{\mathbb{F}}$ defined by

$$\begin{array}{l} \mathbb{F}\downarrow U \times [\underline{S} \Rightarrow U] \\ \quad \downarrow \iota^k \times \langle \overrightarrow{[\![S \triangleright k \vdash \sigma]\!]}, [\![S \triangleright k \vdash \tau]\!] \rangle \\ \delta^k(\mathbb{F}\downarrow U \times U^{|\overrightarrow{\sigma}|} \times U) \xrightarrow{\delta^k(\oplus \times \mathrm{id}_U)} \delta^k(\mathbb{F}\downarrow U \times U) \xrightarrow{(+k, \mathrm{id})} \mathbb{F}\downarrow U \times U \end{array}$$

where $\oplus(n)(\Gamma,\ (\sigma_1,\ldots,\sigma_m)) = \Gamma + \langle\sigma_1\rangle + \cdots + \langle\sigma_m\rangle$. This yields a functor

$$\textstyle\int \mathsf{ext}(k \mid \overrightarrow{\sigma} \vdash \tau) :\ \int(\mathbb{F}{\downarrow}U \ \times\ [\underline{S}\Rightarrow U]) \longrightarrow \int \mathbb{F}{\downarrow}U \times U$$
$$(n,\Gamma,\theta) \longmapsto (n+k \mid \iota^k(\Gamma), \overrightarrow{\theta^\sharp_k(\sigma)} \vdash \theta^\sharp_k(\tau))$$

This is certainly a context extension, that is, it extends a context $(n|\Gamma)$ with k for type context, and instantiated types $\overrightarrow{\theta^\sharp_k(\sigma)} = \overrightarrow{\sigma\,\theta} \in U(n+k)$ for term context using an instantiation map $\theta : S \to \delta^n U$, and moreover the instantiated result type is set to $\tau\theta$.

Definition 4.3 Given a specification of a function symbol $S \triangleright f : \langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1, \ldots, \langle k_\ell\rangle(\overrightarrow{\sigma_\ell})\tau_\ell \to \tau) \in \Sigma^{\mathsf{Tm}}$, we associate the following discrete polynomial diagram P_f

$$\begin{array}{ccc} \ell \cdot \int(\mathbb{F}{\downarrow}U \times [\underline{S}\Rightarrow U]) & \xrightarrow{\nabla_\ell} & \int(\mathbb{F}{\downarrow}U \times [\underline{S}\Rightarrow U]) \\ \Big\downarrow{\scriptstyle [\int \mathsf{ext}(k_i \mid \overrightarrow{\sigma}_i \vdash \tau_i)]_{i\in[\ell]}} & & \Big\downarrow{\scriptstyle \int(\mathrm{id}\times[\![S \triangleright 0 \vdash \tau]\!])} \\ \int(\mathbb{F}{\downarrow}U \times U) & & \int(\mathbb{F}{\downarrow}U \times U) \end{array}$$

Given a polymorphic signature $\Sigma = (\Sigma^{\mathsf{Ty}}, \Sigma^{\mathsf{Tm}})$, we define the corresponding *signature functor* $\Sigma : \mathbf{Set}^{\mathsf{G}U} \longrightarrow \mathbf{Set}^{\mathsf{G}U}$ by the polynomial functor construction

$$\Sigma \stackrel{def}{=} \coprod_{f\in\Sigma^{\mathsf{Tm}}} F_{P_f} =$$

$$\coprod_{S \triangleright f:\langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1,\ldots,\langle k_\ell\rangle(\overrightarrow{\sigma_\ell})\tau_\ell\to\tau' \in \Sigma^{\mathsf{Tm}}} (\textstyle\int(\mathrm{id}\times[\![S \triangleright 0 \vdash \tau']\!]))_! \circ (\nabla_\ell)_* \circ ([\int \mathsf{ext}(k_i \mid \overrightarrow{\sigma}_i \vdash \tau_i)]_{i\in[\ell]})^*$$

Since polynomial functors are closed under sum, this is a discrete polynomial functor. Simplifying it, we finally obtain

$$\begin{array}{l} \Sigma A(n \mid \Gamma \vdash \tau) \\ \quad = \coprod_{\substack{S \triangleright f:\langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1,\ldots,\langle k_l\rangle(\overrightarrow{\sigma_l})\tau_l\to\tau' \in \Sigma^{\mathsf{Tm}} \\ \theta\in[\underline{S}\Rightarrow U](n)}} (\tau \equiv \tau'\theta) \times \prod_{1\le i\le l} A(n+k_i \mid \iota^{k_i}(\Gamma), \overrightarrow{\sigma_i\theta} \vdash \tau_i\theta) \end{array}$$

This more clearly exhibits the functor as a "polynomial", i.e. a sum of products functor. We can construct an initial Σ-algebra *inductively* (cf. §2), and the resulting algebra is an object-level polymorphic abstract syntax.

5. Type in term substitution

A type variable in a type is instantiated by a (concrete) type by a Σ^{Ty}-monoid multiplication. A type variable in a *term* must also be instantiated with a (concrete) type. In this section, we axiomatise type-in-term substitution.

Let $U = (U, \nu^U, \mu^U)$ be a Σ^{Ty}-monoid, $\tau \in U(n+k)$, $\overrightarrow{\sigma} \in U(n), |\overrightarrow{\sigma}| = k$. Hereafter we use the abbreviation $\tau\{\overrightarrow{\sigma}\} \stackrel{def}{=} \mu^U(\tau; \nu^U(1), \ldots, \nu^U(n), \overrightarrow{\sigma})$. $\Gamma\{\overrightarrow{\sigma}\}$ is defined similarly. For a presheaf $A \in \mathbf{Set}^{\mathsf{G}U}$, our aim is to define a function of type

$$\varsigma^A_n(-,\sigma) : A(n+1 \mid \Gamma \vdash \tau) \to A(n \mid \Gamma\{\sigma\} \vdash \tau\{\sigma\}) \quad (4)$$

which replaces a type variable $n+1$ in an element of A with a type σ. We invoke again the polynomial formulation. To specify the source and target indices of (4), we use the following discrete polynomial P (since this is unary, the part $a = \mathrm{id}$ is omitted)

$$\mathsf{G}U \xleftarrow{\ s\ } \textstyle\int\big(\mathbb{F}{\downarrow}\delta U \times \delta U \times U\big) \xrightarrow{\ t\ } \mathsf{G}U$$

where using a strength $\mathsf{st} : \delta U \times U \to \delta(U \times U)$,

$$s = \textstyle\int\big((+1 : \mathbb{F} \to \mathbb{F},\ \pi_1)\big) :\ (n,\Gamma,\tau,\sigma) \mapsto (n+1,\Gamma,\tau)$$

$$\begin{array}{l} t = \textstyle\int\big((\mathrm{Id},\ \mathbb{F}{\downarrow}(-\{-\}) \times (-\{-\})) \circ (\mathrm{Id}, \mathsf{st})\big) \\ \qquad : (n,\Gamma,\tau,\sigma) \mapsto (n,\Gamma\{\sigma\},\tau\{\sigma\}) \end{array}$$

The polynomial functor $F_P : \mathbf{Set}^{\mathsf{G}U} \to \mathbf{Set}^{\mathsf{G}U}$ generated by P determines the arity of type-in-term substitution. For later use, we name it as $\uparrow \stackrel{def}{=} F_P$. Now an F_P-algebra $\varsigma^A : \uparrow A \to A$ exactly gives the type of (4).

Only specifying the arity is not enough. A type-in-term substitution also must satisfy the properties of substitution, which we axiomatise using the axioms of one variable substitution (analogous axioms were given for substitution algebra [10]).

Definition 5.1 Let (U, ν^U, μ^U) be a Σ^{Ty}-monoid and $A \in \mathbf{Set}^{\mathsf{G}U}$. Let $\mathsf{new}^U : 1 \to \delta U$ (a generic new variable) be the transpose of $\nu^U : \mathrm{V} \to U$. A *type-in-term substitution* is a natural transformation $\varsigma^A : \uparrow A \to A$ subject to the following axioms

(i) $a \in A(n \mid \Gamma \vdash \tau),\ \sigma \in U_n \ \vdash\ \varsigma^A_n(\mathsf{up}^A_n a,\ \sigma) = a$
(ii) $a \in A(n+1 \mid \Gamma \vdash \tau) \qquad \vdash \varsigma^A_{n+1}(a, \mathsf{new}^U_n) = \mathsf{contr}^A_n\, a$
(iii) $a \in A(n+2 \mid \Gamma \vdash \tau),\ \sigma \in U_{n+1},\ \sigma' \in U_n \ \vdash$
$\qquad \varsigma^A_n(\varsigma^A_{n+1}(a,\sigma),\ \sigma') = \varsigma^A_n(\varsigma^A_{n+1}(\mathsf{sw}^A_n\, a, \mathsf{up}^U_n\, \sigma'),\ \sigma\{\sigma'\})$

An arrow $\rho : m \to n$ in $\mathbb{F}$ gives rise to natural transformations $\rho^U \stackrel{def}{=} \delta^\rho : \delta^m \to \delta^n : \mathbf{Set}^{\mathbb{F}} \to \mathbf{Set}^{\mathbb{F}}$ for $U \in \mathbf{Set}^{\mathbb{F}}$, and $\rho^A \stackrel{def}{=} \delta^{(\rho,\mathrm{id})} : \delta^{(m|\cdot)} \to \delta^{(n|\cdot)} : \mathbf{Set}^{\mathsf{G}U} \to \mathbf{Set}^{\mathsf{G}U}$ for $A \in \mathbf{Set}^{\mathsf{G}U}$. Hence swapping $\mathsf{sw} : 2 \to 2$, weakening $\mathsf{up} : 0 \to 1$, and contraction $\mathsf{contr} : 2 \to 1$ maps in $\mathbb{F}$ give rise corresponding maps in $\mathbf{Set}^{\mathbb{F}}$ and $\mathbf{Set}^{\mathsf{G}U}$.

All the free variables appearing before "$\vdash$" are universally quantified. This have the intuitive reading, e.g. the first axiom says that substituting for a type variable that is not in a term does not affect the term, etc.

The presheaf V of variables has the type-in-term substitution $\varsigma^{\mathrm{V}} : \uparrow \mathrm{V} \to \mathrm{V}$ defined by $\varsigma^{\mathrm{V}} : \mathrm{V}(n+1 \mid \Gamma \vdash \tau) \to \mathrm{V}(n \mid \Gamma\{\sigma\} \vdash \tau\{\sigma\});\ x \mapsto x$ (just changing types).

6. Polymorphic Structures

We define polymorphic structures that provide a general notion of syntax and algebraic model of polymorphic algebraic theories.

Following [24], [20], given $A, B \in \mathbf{Set}^{\mathsf{G}U}$, we define the presheaf $(A \bullet B) \in \mathbf{Set}^{\mathsf{G}U}$ by $(A \bullet B)(n \mid \Gamma \vdash \tau) = \int^{\Delta\in\mathbb{F}|U(n)} A(n \mid \Delta \vdash \tau) \ \times\ \prod_{1\le i\le|\Delta|} B(n \mid \Gamma \vdash \Delta(i))$, where $\int$ denotes a coend. The presheaf $\mathrm{V} \in \mathbf{Set}^{\mathsf{G}U}$ of object variables is defined by $\mathrm{V}(n \mid \Gamma \vdash \tau) = \mathbb{F}{\downarrow}U(n)\big(\langle\tau\rangle, \Gamma\big) \cong \{x \mid (x : \tau) \in \Gamma\}$. Then $(\mathbf{Set}^{\mathsf{G}U}, \bullet, \mathrm{V})$ forms a monoidal category.

(Var)
$$\frac{(x:\tau)\in\Gamma}{x\in\widehat{N}^U_\Sigma X(n\mid\Gamma\vdash\tau)}$$

(Fun)
$$\frac{\begin{array}{l}(S\ \triangleright\ f:\langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1,\dots,\langle k_l\rangle(\overrightarrow{\sigma_l})\tau_l\to\tau)\in\Sigma^{\mathsf{Tm}}\\ \theta:S\rightsquigarrow_n U\qquad k_i=|\overrightarrow{\alpha_i}|\\ t_i\in\widehat{N}^U_\Sigma X(n+k_i\mid\Gamma,\ \overrightarrow{x_i:\sigma_i\theta}\vdash\tau_i\theta)\ \ (1\le i\le l)\end{array}}{f_\theta(\overrightarrow{\alpha_1.\,\overrightarrow{x_1}.t_1},\dots,\overrightarrow{\alpha_l.\,\overrightarrow{x_l}.t_l})\in\widehat{N}^U_\Sigma X(n\mid\Gamma\vdash\tau\theta)}$$

(MVar)
$$\frac{\begin{array}{l}\overrightarrow{\sigma}\in U(n)\qquad|\overrightarrow{\sigma}|=l\\ \mathsf{z}\in X(n+l\mid x_1{:}\,\tau_1,\dots,x_m{:}\,\tau_m\vdash\tau)\\ t_i\in\widehat{N}^U_\Sigma X(n\mid\Gamma\vdash\tau_i\{\overrightarrow{\sigma}\})\ \ (1\le i\le m)\end{array}}{\mathsf{z}[\overrightarrow{\sigma};t_1,\dots,t_m]\in\widehat{N}^U_\Sigma X(n\mid\Gamma\vdash\tau\{\overrightarrow{\sigma}\})}$$

Note: $f_\theta(\overrightarrow{\alpha.\,\overrightarrow{x}.t})$ may be denoted by $f_{\theta(s_1),\dots,\theta(s_m)}(\overrightarrow{\alpha.\,\overrightarrow{x}.t})$ for $S=\{s_1,\dots,s_m\}$, or simply $f(\overrightarrow{t})$ when the omissions are inferable from context.

Fig. 1. Construction rules of an indexed set $\widehat{N}^U_\Sigma X$

6.1. (U,Σ)-Polymorphic Structures

Definition 6.1 Let $\Sigma=(\Sigma^{\mathsf{Ty}},\Sigma^{\mathsf{Tm}})$ be a signature, and U a Σ^{Ty}-monoid. A (U,Σ)-*polymorphic structure* $A=(A,\varsigma^A)$ consists of

(i) a Σ-monoid $A=(A,\alpha,\nu,\mu)$ in $\mathbf{Set}^{\mathsf{G}U}$

(ii) a type-in-term substitution $\varsigma^A:\!\uparrow\! A\to A$ such that

$$\begin{array}{ccccc}
\uparrow\Sigma A & \xrightarrow{\iota_1} & \Sigma\uparrow A & \xrightarrow{\Sigma\varsigma^A} & \Sigma A\\
{\scriptstyle\uparrow\alpha}\downarrow & & & & \downarrow{\scriptstyle\alpha}\\
\uparrow A & & \xrightarrow{\varsigma^A} & & A
\end{array}
\qquad
\begin{array}{ccc}
\uparrow\mathrm{V} & \xrightarrow{\varsigma^{\mathrm{V}}} & \mathrm{V}\\
{\scriptstyle\uparrow\nu}\downarrow & & \downarrow{\scriptstyle\nu}\\
\uparrow A & \xrightarrow{\varsigma^A} & A
\end{array}$$

$$\begin{array}{ccccc}
\uparrow(A\bullet A) & \xrightarrow{\iota_2} & \uparrow A\bullet\uparrow A & \xrightarrow{\varsigma^A\bullet\varsigma^A} & A\bullet A\\
{\scriptstyle\uparrow\mu}\downarrow & & & & \downarrow{\scriptstyle\mu}\\
\uparrow A & & \xrightarrow{\varsigma^A} & & A
\end{array}$$

commute, where ι_1,ι_2 are inclusions that are definable by the above data.

A morphism of (U,Σ)-polymorphic structures $\varphi:A\to A'$, called (U,Σ)-*polymorphic translation*, is a Σ-monoid morphism that makes the right diagram commute. This defines the category $\mathbf{Poly}(U,\Sigma)$.

$$\begin{array}{ccc}
\uparrow A & \xrightarrow{\varsigma^A} & A\\
{\scriptstyle\uparrow\varphi}\downarrow & & \downarrow{\scriptstyle\varphi}\\
\uparrow A' & \xrightarrow[\varsigma^{A'}]{} & A'
\end{array}$$

Next we seek a free (U,Σ)-polymorphic structure. Let $X\in\mathbf{Set}^{\mathsf{G}U}$. We define an indexed set $\widehat{N}^U_\Sigma X$ by the rules in Fig. 1. For every context $(n\mid\Gamma\vdash\tau\{\overrightarrow{\sigma}\})$ where $\overrightarrow{\sigma}\in U(n)$ with $|\overrightarrow{\sigma}|=l$, we define an equivalence relation $\doteq$ on $\widehat{N}^U_\Sigma X(n\mid\Gamma\vdash\tau\{\overrightarrow{\sigma}\})$ generated by context closure of

$$\mathsf{z}[\overrightarrow{\sigma};\,t_{\pi 1},\dots,t_{\pi|\Delta|}]\ \doteq\ X(\mathrm{id},\pi)(\mathsf{z})[\overrightarrow{\sigma};\,t_1,\dots,t_{|\Delta'|}]\quad(5)$$

where $\pi:\Delta\to\Delta'$, $\mathsf{z}\in X(n+l\mid\Delta\vdash\tau)$. The presheaf $N^U_\Sigma X\in\mathbf{Set}^{\mathsf{G}U}$ is defined by $N^U_\Sigma X(n\mid\Gamma\vdash\tau)\stackrel{def}{=}\widehat{N}^U_\Sigma X(n\mid\Gamma\vdash\tau)\,/\,\doteq$

Let $\uparrow^*(X)\stackrel{def}{=}\coprod_{i\in\mathbb{N}}\uparrow^i(X)$. We define the algebra structure $[\nu,[f^{N^U_\Sigma X}]_{f\in\Sigma^{\mathsf{Tm}}},\mathsf{mapp}]:\mathrm{V}+\Sigma+(\uparrow^*X\bullet N^U_\Sigma X)\to N^U_\Sigma X$ on $N^U_\Sigma X$ by

$$\nu(x)=x;\ f^{N^U_\Sigma X}(\overrightarrow{t})=f(\overrightarrow{t});\ \mathsf{mapp}(\mathsf{z},\overrightarrow{\sigma};\overrightarrow{t})=\mathsf{z}[\overrightarrow{\sigma};\overrightarrow{t}].$$

Theorem 6.2 *$N^U_\Sigma X$ is an initial $\mathrm{V}+\Sigma+(\uparrow^*X\bullet-)$-algebra.*

$N^U_\Sigma X$ is a monoid in $\mathbf{Set}^{\mathsf{G}U}$ and has a syntactic type-in-term substitution $\varsigma^{N^U_\Sigma X}$. We will write $t\{\sigma\}$ for $\varsigma^{N^U_\Sigma X}(t,\sigma)$.

Definition 6.3 **(Assignment)** Given (U,Σ)-polymorphic structure $(A,\nu^A,\mu^A,\varsigma^A)$, a morphism $\varphi:X\to A$ of $\mathbf{Set}^{\mathsf{G}U}$ is called an *assignment*. The *extension* $\varphi^\sharp:N^U_\Sigma X\to A$ is a (U,Σ)-polymorphic translation defined by

$$\varphi^\sharp(x)=\nu^A(x)$$
$$\varphi^\sharp(f(\overrightarrow{\alpha_1.\overrightarrow{x_1}.t_1},\dots,\overrightarrow{\alpha_l.\overrightarrow{x_l}.t_l}))=f^A(\varphi^\sharp_{(n+|\overrightarrow{\alpha}_1|\mid\Gamma,\overrightarrow{x_1:\sigma_1}\vdash\tau)}(t_1),\dots,\varphi^\sharp_{(n+|\overrightarrow{\alpha}_l|\mid\Gamma,\overrightarrow{x_l:\sigma_l}\vdash\tau)}(t_l))$$
$$\varphi^\sharp(\mathsf{z}[\overrightarrow{\sigma};\overrightarrow{t}])=\mu^A\big(\varsigma^A(\varphi(\mathsf{z}),\overrightarrow{\sigma});\ \overrightarrow{\varphi^\sharp(t)}\big)$$

We will also write $t\,\varphi$ for $\varphi^\sharp(t)$.

As explained in §1-III, $\varphi^\sharp$ is the unique *extension* of φ to a (U,Σ)-polymorphic translation. This $\varphi^\sharp$ gives the meaning of a meta-term in a polymorphic structure A using an assignment φ. The proof of the next theorem essentially uses this extension.

Theorem 6.4 *Let $X\in\boldsymbol{Set}^{\mathsf{G}U}$. $N^U_\Sigma X$ is a free (U,Σ)-polymorphic structure over X. Hence the functor that forgets the (U,Σ)-polymorphic structure has the left adjoint $N^U_\Sigma:\boldsymbol{Set}^{\mathsf{G}U}\to\boldsymbol{Poly}(U,\Sigma)$.*

6.2. Free (U,Σ)-Polymorphic Structure as Polymorphic Abstract Syntax with Metavariables

We want to give a free structure $N^U_\Sigma X$ syntactically to use it as a *polymorphic abstract syntax with metavariables*. Given $X\in\mathbf{Set}^{\mathsf{G}U}$, we can inductively construct the underlying set $\widehat{N}^U_\Sigma X$ using Fig. 1, hence in this sense, it is syntactic. However, there is one issue. To give a *presheaf* X syntactically is hard, because to give a functor $X:\mathsf{G}U\to\mathbf{Set}$ requires (1) the arrow part $X(\rho,\pi)$ for all (ρ,π) in $\mathsf{G}U$, and (2) functoriality.

Instead, we consider a simpler index structure for contexts. Define $\mathsf{D}:\Sigma^{\mathsf{Ty}}\text{-}\mathbf{Mon}\to\mathbf{Cat}$; $\mathsf{D}(X)\stackrel{def}{=}\sum_{n\in\mathbb{N}}\mathbb{N}\downarrow X(n)\times X(n)$ (D for "discrete", due to $\mathbb{N}$). To give $X\in\mathbf{Set}^{\mathsf{D}U}$ is to give just an indexed set $X(n\mid\Gamma\vdash\tau)$ indexed by contexts $(n\mid\Gamma\vdash\tau)\in\mathsf{D}U$. We regard $\mathsf{z}\in X(n\mid\Gamma\vdash\tau)$ as a metavariable, which may also be denoted by $\mathsf{z}:\langle n\rangle(\Gamma)\tau$ in later examples. A metavariable z is intended to be replaced by a term involving free type variables from n, and free variables from Γ.

Given a set of metavariables X, regarded as a presheaf $X\in\mathbf{Set}^{\mathsf{D}U}$, we can construct a presheaf $\underline{X}\in\mathbf{Set}^{\mathsf{G}U}$ by a left Kan extension along the inclusion $\mathsf{D}U\to\mathsf{G}U$. A left Kan extension is the left adjoint to the restriction functor $|-|$ (see below), and together with Theorem 6.4, there is a series of adjunctions

$$\mathbf{Set}^{\mathsf{D}U}\ \underset{|-|}{\overset{(-)}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}}\ \mathbf{Set}^{\mathsf{G}U}\ \underset{\longleftarrow}{\overset{N^U_\Sigma}{\overset{\longrightarrow}{\bot}}}\ \mathbf{Poly}(U,\Sigma)$$

This tells us a way to give a free polymorphic structure syntactically. Starting from a set of metavariables $X \in \mathbf{Set}^{\mathsf{D}U}$, we can freely generate a polymorphic structure $N^U_\Sigma \underline{X}$, for which we require only a syntactic data X. We call an element

$$t \ \in \ N^U_\Sigma \underline{X}(n \mid \Gamma \vdash \tau)$$

a *meta-term* because it is a term that may involve metavariables as in (MVar) of Fig. 1. As in §1-III, we define a meta-substitution, i.e. substitution for metavariables in a meta-term. The idea is to employ the notion of assignment in Def. 6.3.

Let X be a set of metavariables. An *assignment* $\vartheta : X \rightsquigarrow_{(k|\Delta)} A$ for meta-substitution is given by an $\mathsf{D}U$-indexed function $\vartheta : X \to |\delta^{(k|\Delta)}A|$ where $(k|\Delta)$ is "possible free type and term variables" in the results, and $|-|$ is the above restriction functor. It maps each metavariable z as

$$X(n \mid \Gamma \vdash \tau) \ \ni \ \mathsf{z} \overset{\vartheta_{(n \mid \Gamma \vdash \tau)}}{\longmapsto} \ a \ \in A(k+n \mid \Delta, \Gamma \vdash \tau).$$

By adjointness, the map ϑ in $\mathbf{Set}^{\mathsf{D}U}$ bijectively corresponds to a map $\hat{\vartheta} : \underline{X} \to A$ in $\mathbf{Set}^{\mathsf{G}U}$. Applying Def. 6.3, we have $\hat{\vartheta}^\sharp : N^U_\Sigma \underline{X} \to A$, which gives a meta-substitution that replaces all metavariables in a meta-term using ϑ.

6.3. Σ-Polymorphic Structures

We combine all (U, Σ)-polymorphic structures varying all possible U together, and then construct a single category $\mathbf{Poly}(\Sigma)$ by the Grothendieck construction:

$$\mathbf{Poly}(\Sigma) \ \overset{def}{=} \ \int^U \mathbf{Poly}(U, \Sigma)$$

where $\mathbf{Poly}(-, \Sigma) : \Sigma^{\mathsf{Ty}}\text{-}\mathbf{Mon}^{\mathsf{op}} \to \mathbf{CAT}$ is defined by

$$U \mapsto \mathbf{Poly}(U, \Sigma)$$
$$f : U \to U' \mapsto - \circ \mathsf{G}f : \mathbf{Poly}(U', \Sigma) \to \mathbf{Poly}(U, \Sigma).$$

Definition 6.5 A *Σ-polymorphic structure* (U, A) consists of a Σ^{Ty}-monoid U and a (U, Σ)-polymorphic structure A.

Let (U', A') be another Σ-polymorphic structure. A *Σ-polymorphic translation* from (U, A) to (U', A') denoted by

$$(\llparenthesis - \rrparenthesis, \varphi) : (U, A) \longrightarrow (U', A')$$

consists of a pair of maps

- a Σ^{Ty}-monoid morphism $\llparenthesis - \rrparenthesis : U \to U'$
- a (U, Σ)-polymorphic translation $\varphi : A \longrightarrow A'\mathsf{G}\llparenthesis - \rrparenthesis$.

Note that if (U', A') is a Σ-polymorphic structure, so is $(U, \ A'\mathsf{G}\llparenthesis - \rrparenthesis)$. The idea is that $\llparenthesis - \rrparenthesis$ translates types in U to U' and φ translates elements from A to A' that also takes into account of the universe shift. An important point is that we derived the definition of translation by the Grothendieck construction. Actually, the category $\mathbf{Poly}(\Sigma)$ consists of Σ-polymorphic structures and Σ-polymorphic translations.

6.4. Universe shift

The category $\mathbf{Poly}(\Sigma)$ offers a right notion of a translation between polymorphic structures on possibly different universes. As an application of it, we transport a meta-term on a universe W to that on another universe U. Let $\llparenthesis - \rrparenthesis : W \to U$ be a Σ^{Ty}-monoid morphism.

- For a type $\tau \in W(n)$, $\llparenthesis - \rrparenthesis$ translates it to a type $\llparenthesis \tau \rrparenthesis \in U(n)$.

We overload the notation $\llparenthesis - \rrparenthesis$ for other syntactic constructs:

- For $\Gamma = x_1 : \tau_1, \ldots, x_l : \tau_l$, $\llparenthesis \Gamma \rrparenthesis \overset{def}{=} x_1 : \llparenthesis \tau_1 \rrparenthesis, \ldots, x_l : \llparenthesis \tau_l \rrparenthesis$.
- For a set X of metavariables on a universe W, define a set $\llparenthesis X \rrparenthesis$ of metavariables on a universe U by $\llparenthesis X \rrparenthesis \overset{def}{=} \mathrm{Lan}_J X$, (a left Kan extension), where $J = \mathsf{D}\llparenthesis - \rrparenthesis : \mathsf{D}W \to \mathsf{D}U$. Hence $\llparenthesis X \rrparenthesis(n \mid \llparenthesis \Theta \rrparenthesis \vdash \llparenthesis \sigma \rrparenthesis) = X(n \mid \Theta \vdash \sigma)$, and other cases are empty.
- For a meta-term $t \in N^W_\Sigma \underline{X}(n \mid \Gamma \vdash \tau)$, a meta-term $\llparenthesis t \rrparenthesis \in N^U_\Sigma \underline{\llparenthesis X \rrparenthesis}(n \mid \llparenthesis \Gamma \rrparenthesis \vdash \llparenthesis \tau \rrparenthesis)$ is defined by replacing every type τ in t with $\llparenthesis \tau \rrparenthesis$. More precisely, $\llparenthesis t \rrparenthesis \overset{def}{=} \mathsf{ushift}^\sharp(t)$ where the assignment $\mathsf{ushift} : X \to N^U_\Sigma \underline{\llparenthesis X \rrparenthesis} \circ \mathsf{G}\llparenthesis - \rrparenthesis$ is defined by $\mathsf{ushift}(n \mid \overrightarrow{x : \tau} \vdash \tau)(\mathsf{z}) = \mathsf{z}[; \overrightarrow{x}]$.

7. Polymorphic Equational Logic

In this section, we give a Polymorphic Equational Logic PEL that constitutes a sound and complete logical framework for equational reasoning about polymorphic algebraic theories.

First we define the judgment for equation, then define the notions of satisfiability and models. Let W be a universe. A *W-equation* is of the form

$$X \ \triangleright \ n \mid \Gamma \vdash_W \ s = t : \tau$$

for meta-terms $s, t \in N^W_\Sigma \underline{X}(n \mid \Gamma \vdash \tau)$. A set E of W-equations is called *W-axioms*. When W is presented as the universe $W = \mathcal{M}\underline{S}$ of meta-types, we write an $\mathcal{M}\underline{S}$-axiom as $S \mid X \ \triangleright \ n \mid \Gamma \vdash s = t : \tau$ (e.g. Fig. 3, 4, 5).

Definition 7.1 A Σ-polymorphic structure (U, A) *satisfies* a W-equation $X \ \triangleright \ n \mid \Gamma \vdash_W s = t : \tau$ if

- for all Σ^{Ty}-monoid morphisms $\llparenthesis - \rrparenthesis : W \to U$,
- for all contexts $(k|\Delta)$, assignments $\varphi : X \rightsquigarrow_{(k|\Delta)} A\mathsf{G}\llparenthesis - \rrparenthesis$,

$$\varphi^\sharp_{(n \mid \Gamma \vdash \tau)}(s) \ = \ \varphi^\sharp_{(n \mid \Gamma \vdash \tau)}(t)$$

holds in $A(n + k \mid \llparenthesis \Gamma, \Delta \rrparenthesis \vdash \llparenthesis \tau \rrparenthesis)$. If (U, A) satisfies all equations in W-axioms E, (U, A) is called a *(polymorphic) model* of E.

The idea is to apply $(\llparenthesis - \rrparenthesis, \varphi^\sharp) : (W, N^U_\Sigma \underline{X}) \longrightarrow (U, \delta^{(k|\Delta)}A)$ to the W-equation, which is the unique Σ-polymorphic translation that extends φ. Note that if A is a (U, Σ)-polymorphic structure, so is $\delta^{(k|\Delta)}A$.

$$\text{(Ax)}\frac{(X \rhd n \mid \Gamma \vdash_W s = t : \tau) \in E}{(\!|X|\!) \rhd n \mid (\!|\Gamma|\!) \vdash_U (\!|s|\!) = (\!|t|\!) : (\!|\tau|\!)}\ (\!|-|\!) : W \to U$$

$$\text{(Ref)}\ \frac{}{X \rhd n \mid \Gamma \vdash_U t = t : \tau} \qquad \text{(Sym)}\ \frac{X \rhd n \mid \Gamma \vdash_U s = t : \tau}{X \rhd n \mid \Gamma \vdash_U t = s : \tau} \qquad \text{(Tra)}\ \frac{X \rhd n \mid \Gamma \vdash_U s = t : \tau \quad X \rhd n \mid \Gamma \vdash_U t = u : \tau}{X \rhd n \mid \Gamma \vdash_U s = u : \tau}$$

$$\text{(Sub)}\ \frac{X \rhd n \mid \Gamma \vdash_U s_i : \sigma_i \qquad (1 \le i \le m) \quad X \rhd n \mid x_1 : \sigma_1, \ldots, x_m : \sigma_m \vdash_U t = t' : \tau}{X \rhd n \mid \Gamma \vdash_U t[\overrightarrow{x_i := s_i}] = t'[\overrightarrow{x_i := s_i}] : \tau}$$

$$\text{(Fun)}\ \frac{S \rhd f : \langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1, \ldots, \langle k_l\rangle(\overrightarrow{\sigma_l})\tau_l \to \tau \in \Sigma^{\mathsf{Tm}} \qquad k_i = |\overrightarrow{\alpha}_i| \quad \theta : S \rightsquigarrow_n U \qquad X \rhd n + k_i \mid \Gamma,\ \overrightarrow{x_i : \sigma_i\theta} \vdash_U s_i = t_i : \tau_i\theta \quad (1 \le i \le l)}{X \rhd n \mid \Gamma \vdash_U f_\theta(\overrightarrow{\alpha_1}.\overrightarrow{x_1}.s_1, \ldots, \overrightarrow{\alpha_l}.\overrightarrow{x_l}.s_l) = f_\theta(\overrightarrow{\alpha_l}.\overrightarrow{x_1}.t_1, \ldots, \overrightarrow{\alpha_l}.\overrightarrow{x_l}.t_l) : \tau\theta}$$

$$\text{(MSub)}\ \frac{\vartheta : X \rightsquigarrow_{(k|\Delta)} N^U_\Sigma X' \qquad X \rhd n \mid \Gamma \vdash_U s = t : \tau}{X' \rhd k + n \mid \Delta, \Gamma \vdash_U s\vartheta = t\vartheta : \tau}$$

$$\text{(MEx)}\ \frac{\big(\mathrm{z} : \langle n + k\rangle(\tau_1, \ldots, \tau_l)\tau\big) \in X \qquad |\overrightarrow{\sigma}| = k \quad X \rhd n \mid \Gamma \vdash_U \overrightarrow{\sigma} \qquad X \rhd n \mid \Gamma \vdash_U s_i = t_i : \tau_i\{\overrightarrow{\sigma}\} \qquad (1 \le i \le l)}{X \rhd n \mid \Gamma\{\overrightarrow{\sigma}\} \vdash_U \mathrm{z}[\overrightarrow{\sigma}; s_1, \ldots, s_l] = \mathrm{z}[\overrightarrow{\sigma}; t_1, \ldots, t_l] : \tau\{\overrightarrow{\sigma}\}}$$

$$\text{(Gr)}\ \frac{\rho : m \to n \qquad \pi : \rho(\Gamma) \to \Delta \quad X \rhd m \mid \Gamma \vdash_U s = t : \tau}{X \rhd n \mid \Delta \vdash_U \pi\rho(s) = \pi\rho(t) : \rho(\tau)} \qquad \text{(TSub)}\ \frac{X \rhd n + 1 \mid \Gamma \vdash_U s = t : \tau \qquad n \vdash \sigma}{X \rhd n \mid \Gamma\{\sigma\} \vdash_U s\{\sigma\} = t\{\sigma\} : \tau\{\sigma\}}$$

Fig. 2. Polymorphic Equational Logic PEL

Vernacular notation

$$\begin{array}{lllll} (\beta) & \Gamma \vdash (\lambda x.\, M)\, N & = & M[x := N] & : \tau \\ (\text{type } \beta) & \Gamma \vdash (\Lambda\alpha.M)\, \sigma & = & M[\alpha := \sigma]) & : \tau\{\alpha := \sigma\} \end{array}$$

Notation in polymorphic equational logic

$$\begin{array}{lllll} (\beta) & \mathrm{S}, \mathrm{T} : * \mid \mathrm{M} : (\mathrm{S})\mathrm{T},\ \mathrm{N} : \mathrm{S} \rhd \vdash \mathsf{app}(\,\mathsf{abs}(x.\,\mathrm{M}[\,;x\,]), \mathrm{N}\,) & = & \mathrm{M}[\,;\mathrm{N}\,] & : \mathrm{T} \\ (\text{type } \beta) & \mathrm{S} : *,\ \mathrm{T} : \langle *\rangle * \mid \mathrm{M} : \langle\alpha\rangle\mathrm{T}[\alpha] \quad \rhd \vdash \mathsf{tapp}(\,\mathsf{tabs}(\alpha.\,\mathrm{M}[\,\alpha;\,])) & = & \mathrm{M}[\,\mathrm{S};\,]\,) & : \mathrm{T}\,[\mathrm{S}] \end{array}$$

Fig. 3. Example: axioms for System F

Vernacular notation

$$\begin{array}{lllll} (\mathsf{let}_\wedge) & \Gamma \vdash \mathsf{let}\ \langle x_1, x_2\rangle = \langle M_1, M_2\rangle\ \mathsf{in}\ M & = & M[x_1 := M_1, x_2 := M_2] & : \tau \\ (\mathsf{let}\eta) & \Gamma \vdash \mathsf{let}\ \langle x_1, x_2\rangle = N\ \mathsf{in}\ M[z := \langle x_1, x_2\rangle] & = & M[z := N] & : \tau \\ (\exists\beta) & \Gamma \vdash \mathsf{unpack}\ \langle \iota, N\rangle\ \mathsf{as}\ \langle \alpha, x\rangle\ \mathsf{in}\ M & = & M[\alpha := \iota,\ x := N] & : \tau\{\alpha := \iota\} \\ (\exists\eta) & \Gamma \vdash \mathsf{unpack}\ N\ \mathsf{as}\ \langle \alpha, x\rangle\ \mathsf{in}\ M[z := \langle \alpha, x\rangle] & = & M[z := N] & : \tau \end{array}$$

Notation in polymorphic equational logic

$$\begin{array}{llllll} (\text{let}\wedge) & \mathrm{S}_1, \mathrm{S}_2, \mathrm{T} : * & \mid \mathrm{M} : (\mathrm{S}_1, \mathrm{S}_2)\mathrm{T},\ \mathrm{M}_1 : \mathrm{S}_1,\ \mathrm{M}_2 : \mathrm{S}_2 \rhd \vdash \mathsf{let}\big(\mathsf{pair}(\mathrm{M}_1, \mathrm{M}_2),\ x_1.x_2.\mathrm{M}[\,;x_1, x_2\,]\big) & = & \mathrm{M}[\,;\mathrm{M}_1, \mathrm{M}_2\,] & : \mathrm{T} \\ (\text{let}\,\eta) & \mathrm{S}_1, \mathrm{S}_2, \mathrm{T} : * & \mid \mathrm{M} : (\mathrm{S}_1 \wedge \mathrm{S}_2)\mathrm{T},\ \ \mathrm{N} : \mathrm{S}_1 \wedge \mathrm{S}_2 \quad \rhd \vdash \mathsf{let}\big(\mathrm{N},\ x_1.x_2.\mathrm{M}[\,;\mathsf{pair}(x_1, x_2)\,]\big) & = & \mathrm{M}[\,;\mathrm{N}\,] & : \mathrm{T} \\ (\exists\beta) & \mathrm{U} : *,\ \mathrm{S}, \mathrm{T} : \langle *\rangle * & \mid \mathrm{M} : \langle\alpha\rangle(\mathrm{S})\mathrm{T}[\alpha],\ \mathrm{N} : \mathrm{S}\,[\mathrm{U}] \quad \rhd \vdash \mathsf{unpack}_{\mathrm{S},\mathrm{T}[\mathrm{U}]}\big(\mathsf{pack}_{\mathrm{S},\mathrm{U}}(\mathrm{N}),\ \alpha.x.\mathrm{M}[\,\alpha;x\,]\big) & = & \mathrm{M}[\,\mathrm{U}\,;\,\mathrm{N}] & : \mathrm{T}\,[\mathrm{U}] \\ (\exists\eta) & \mathrm{S} : \langle *\rangle *,\ \mathrm{T} : * & \mid \mathrm{M} : (\exists(\alpha.\mathrm{S}))\,\mathrm{T},\ \ \mathrm{N} : \exists(\alpha.\mathrm{S}) \quad \rhd \vdash \mathsf{unpack}_{\mathrm{S},\mathrm{T}}\big(\mathrm{N},\ \alpha.x.\mathrm{M}[\mathsf{pack}_{\mathrm{S},\alpha}(x)]\big) & = & \mathrm{M}[\,;\mathrm{N}\,] & : \mathrm{T} \end{array}$$

Fig. 4. Example: axioms for the existential λ-calculus $\lambda^\exists$

$$\begin{array}{llllll} \mathrm{V} : * \mid \mathrm{X} : \mathsf{E} & \rhd\ \ell : \mathsf{L} & \vdash\ \mathsf{lookup}(\ell, v.\mathsf{update}(\ell, v, \mathrm{X})) & = & \mathrm{X} & : \mathsf{E} \\ \mathrm{V} : * \mid \mathrm{X} : (\mathrm{V}, \mathrm{V})\mathsf{E} & \rhd\ \ell : \mathsf{L} & \vdash\ \mathsf{lookup}(\ell, w.\mathsf{lookup}(\ell, v.\mathrm{X}[v, w])) & = & \mathsf{lookup}(\ell, v.\mathrm{X}[v, v]) & : \mathsf{E} \\ \mathrm{V} : * \mid \mathrm{X} : \mathsf{E} & \rhd\ \ell : \mathsf{L}, v, w : \mathrm{V} & \vdash\ \mathsf{update}(\ell, v, \mathsf{update}(\ell, w, \mathrm{X})) & = & \mathsf{update}(\ell, w, \mathrm{X}) & : \mathsf{E} \end{array}$$

Fig. 5. Example: axioms for global state (excerpt)

7.1. Equational logic

The Polymorphic Equational Logic PEL is given by the rules in Fig. 2. The notation $[x := t]$ means to replace a object variable x with t. The notation $\rho(s)$ (resp. $\pi(s)$) means to replace type variables (resp. variables) in s by ρ (resp. π).

To design a sound and complete inference system for polymorphic equational reasoning, our approach is to *distill* syntactic rules from various aspects of the polymorphic structure $N^U_\Sigma \underline{X}$ and required properties for equational reasoning as follows.

(Ref),(Sym),(Tra): The rules for equivalence relation.

(Ax): An axiom becomes a theorem. Also types of the axiom must be instantiated by a Σ^{Ty}-monoid morphism $(\!|-|\!)$ (e.g. from meta-types to object types).

(Gr): $N^U_\Sigma \underline{X}$ is a presheaf in $\mathbf{Set}^{\mathsf{G}U}$. This rule is for a presheaf action (cf. [20] §3.2 Meaning of arrows).

(Fun): Σ-algebra structure of $N^U_\Sigma \underline{X}$.

(MEx): $(X \bullet -)$-algebra structure of $N^U_\Sigma \underline{X}$.

(TSub): Type-in-term substitution on $N^U_\Sigma \underline{X}$.

(Sub): Multiplication of the Σ-monoid $N^U_\Sigma \underline{X}$.

(MSub): Polymorphic translation $\vartheta^\sharp : N^U_\Sigma \underline{X} \to N^U_\Sigma \underline{X}'$.

We list several important derived and admissible rules.

Structural rules. Weakening, contraction, and permutation (= injective renaming) on metavariables X are obtained as instances of (MSub). Weakening, contraction, and permutation on type contexts n and term contexts Γ are obtained as instances of (Gr).

Universe shift. In PEL, a universe can be changed when an axiom is instantiated in (Ax). We have also an admissible rule of the following "universe shift" rule using $(\!|-|\!) : W \to U$:

$$\text{(UnivSh)}\frac{X \triangleright n \mid \Gamma \vdash_W s = t : \tau}{(\!|X|\!) \triangleright n \mid (\!|\Gamma|\!) \vdash_U (\!|s|\!) = (\!|t|\!) : (\!|\tau|\!)}$$

This is obtained by the fact that $((\!|-|\!), \mathsf{ushift}^\sharp)$ (cf. §6.4) is a Σ-polymorphic translation.

7.2. Soundness and completeness

Let U be an arbitrary universe. Define X to be the disjoint union of metavariable contexts in all equations in W-axioms E then applying a Σ^{Ty}-monoid morphism $(\!|-|\!) : W \to U$, hence $X \in \mathbf{Set}^{\mathsf{D}U}$. Let U be a Σ^{Ty}-monoid. Take the term polymorphic structure $(U, N^U_\Sigma \underline{X})$. It is clear that derivable equations from E defines an $(n \mid \Gamma \vdash \tau)$-indexed equivalence relation $=_E$ on $N^U_\Sigma \underline{X}(n \mid \Gamma \vdash \tau)$. We define $N^U_E \underline{X}(n \mid \Gamma \vdash \tau) \stackrel{def}{=} N^U_\Sigma \underline{X}(n \mid \Gamma \vdash \tau)/=_E$.

Lemma 7.2 *Then $(U, N^U_E \underline{X})$ is a polymorphic model of E.*

Theorem 7.3 *Let E be W-axioms. The polymorphic equational logic is sound and complete, i.e. the following are equivalent for all universes U:*

(i) $(X \triangleright n \mid \Gamma \vdash_U s = t : \tau)$ is derivable from E in PEL.

(ii) For all polymorphic models (V, A) of E, (V, A) satisfies $(X \triangleright n \mid \Gamma \vdash_U s = t : \tau)$.

8. Examples

Example 8.1 Continued from the examples in §3. Axioms of System F, the existential λ-calculus, global state [28] can be written as PEL's axioms in Fig. 3, 4, 5 respectively.

Example 8.2 **(CPS Translation from F to $\lambda_\exists$ [11])** Fujita gives a CPS translation from System F to the existential λ-calculus $\lambda_\exists$. The translation consists of the type translation $(-)^\bullet$ from System F types to $\lambda_\exists$-types, and the CPS translation $[\![-]\!]$ from System F terms to $\lambda_\exists$-terms. (excerpts: $(\forall(\alpha.\tau))^\bullet = \neg\exists\alpha.\neg\tau^\bullet$; $[\![\Lambda\alpha.M]\!] = \lambda a.\, a\,(\lambda k.\mathsf{unpack}\ k\ \mathsf{as}\ \langle\alpha, c\rangle\ \mathsf{in}\ [\![M]\!]c)$) The CPS translation is sound:

$$\Gamma \vdash_{\mathrm{Sys\,F}} s = t : \tau \quad \Rightarrow \quad \neg\neg\Gamma^\bullet \vdash_{\lambda\exists} [\![s]\!] = [\![t]\!] : \neg\neg\tau^\bullet. \quad (6)$$

Interestingly, this pair of translations is an example of our notion of polymorphic translation, i.e. a Σ_{F}-polymorphic translation $((-)^\bullet, [\![-]\!]) : (\mathbb{T}_{\mathrm{F}}, \Lambda_{\mathrm{F}}) \longrightarrow (\mathbb{T}_\exists, \Lambda^{\neg\neg}_\exists)$ and moreover $(\mathbb{T}_\exists, \Lambda^{\neg\neg}_\exists)$ is a model. We explain what this means. Let $\mathbb{T}_\exists$ be the initial $\Sigma^{\mathsf{Ty}}_\exists$-monoid, $\Lambda_\exists$ the free $\Sigma_\exists$-polymorphic structure over 0, $\mathbb{T}_{\mathrm{F}}$ the initial $\Sigma^{\mathsf{Ty}}_{\mathrm{F}}$-monoid, Λ_{F} the free Σ_{F}-polymorphic structure over 0. Define $\Lambda^{\neg\neg}_\exists(n \mid \Gamma \vdash \tau) \stackrel{def}{=} \Lambda_\exists(n \mid \neg\neg\Gamma \vdash \neg\neg\tau)$. The definition of $(-)^\bullet$ determines an $\Sigma^{\mathsf{Ty}}_{\mathrm{F}}$-algebra structure on $\mathbb{T}_\exists$. The definition of $[\![-]\!]$ determines a Σ_{F}-algebra structure on $\Lambda^{\neg\neg}_\exists$. It amounts to a Σ_{F}-polymorphic structure $(\mathbb{T}_\exists, \Lambda^{\neg\neg}_\exists)$, which is a model of System F by (6).

Example 8.3 **(Categorical model of F [30])** Let $\mathbb{C}$ be a category with finite products, which consists of a distinguished object Ω which generates all other objects using the finite products. A *PL-category* $\mathcal{C}$ is a functor $\mathcal{C} : \mathbb{C}^{\mathsf{op}} \to \mathbf{CCC}$ (the category of all cartesian closed categories) defined by $\mathcal{C} \stackrel{def}{=} \mathbb{C}(-, \Omega)$. From a PL-category, we can define a Σ_{F}-polymorphic structure $(\mathrm{U},\ \mathrm{PL} \in \mathbf{Set}^{\mathsf{G}\mathrm{U}})$ by

$$\mathrm{U} \stackrel{def}{=} |\mathbb{C}(\Omega^{(-)}, \Omega)| \in \mathbf{Set}^{\mathbb{F}}$$

$$\mathrm{PL}(n \mid \Gamma \vdash \tau) \stackrel{def}{=} \mathbb{C}(\Omega^n, \Omega)(\langle\Gamma\rangle, \tau)$$

where the notation $|-|$ denotes the underlying set, and $\langle\Gamma\rangle = \prod_{1 \le i \le |\Gamma|} \Gamma(i)$. Seely's interpretation of System F's types and terms using a PL-category [30] determines algebra structures on U and PL. For example, for $\forall : \langle * \rangle * \to * \in \Sigma^{\mathsf{Ty}}_{\mathrm{F}}$, the corresponding algebra operation $\forall^{\mathrm{U}} : \delta\mathrm{U} \to \mathrm{U}$ is defined by the right adjoint to π^* where $\pi : \Omega^{n+1} \to \Omega^n$, etc. The multiplications of both monoids U and PL are given by compositions. The type-in-term substitution of $\sigma \in \mathrm{U}(n)$ is arisen as $\langle \mathrm{id}^n_\Omega, \sigma \rangle^*_{\mathsf{arr}} : \mathrm{PL}(n+1 \mid \Gamma \vdash \tau) \to \mathrm{PL}(n \mid \Gamma\{\sigma\} \vdash \tau\{\sigma\})$. The Σ_{F}-polymorphic structure $(\mathrm{U}, \mathrm{PL})$ amounts to a model of System F.

References

[1] N. Benton, C.-K. Hur, A. Kennedy, and C. McBride. Strongly typed term representations in Coq. *Journal of Automated Reasoning*, 49(2):141–159, 2012.

[2] N. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the church-rosser theorem. *Indagationes Mathematicae*, 34:381–391, 1972.

[3] M. Fiore. Second-order and dependently-sorted abstract syntax. In *Proc. of LICS'08*, pages 57–68, 2008.

[4] M. Fiore. Discrete generalised polynomial functors. In *ICALP'12*, pages 214–226, 2012.
[5] M. Fiore and C.-K. Hur. Second-order equational logic. In *Proc. of CSL'10*, LNCS 6247, pages 320–335, 2010.
[6] M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proc. of LICS'99*, pages 193–202, 1999.
[7] K. Fujita. Galois embedding from polymorphic types into existential types. In *Proc. of TLCA'05*, pages 194–208, 2005.
[8] N. Gambino and M. Hyland. Wellfounded trees and dependent polynomial functors. In *TYPES'03*, pages 210–225, 2003.
[9] J.-Y. Girard. The system F of variable types, fifteen years later. *Theor. Comput. Sci.*, 45(2):159–192, 1986.
[10] A. Grothendieck. Catégories fibrées et descente (exposé VI). In *Revêtement Etales et Groupe Fondamental (SGA1)*, Lecture Notes in Mathematics 224, pages 145–194. Springer, 1970.
[11] M. Hamana. Free Σ-monoids: A higher-order syntax with metavariables. In *Proc. of APLAS'04*, LNCS 3302, pages 348–363, 2004.
[12] M. Hamana. Polymorphic abstract syntax via Grothendieck construction. In *Proc. of FoSSaCS'11*, pages 381–395, 2011.
[13] M. Hamana and M. Fiore. A foundation for GADTs and inductive families: Dependent polynomial functor approach. In *Proc. of WGP'11*, pages 59–70, 2011.
[14] M. Hofmann. Semantical analysis of higher-order abstract syntax. In *Proc. of LICS'99*, pages 204–213, 1999.
[15] H. Hosoya, A. Frisch, and G. Castagna. Parametric polymorphism for XML. *ACM Trans. Program. Lang. Syst.*, 32(1), 2009.
[16] F.W. Lawvere. Adjointness in foundations. *Dialectica*, pages 281–296, 1969.
[17] T. Leinster. *Higher Operads, Higher Categories*. London Mathematical Society Lecture Note Series 298. Cambridge University Press, 2004.
[18] R. E. Møgelberg. From parametric polymorphism to models of polymorphic FPC. *Math. Struct. in Compt. Sci.*, 19(4):639–686, 2009.
[19] M. Miculan and I. Scagnetto. A framework for typed HOAS and semantics. In *Proc. of PPDP'03*, pages 184–194. ACM Press, 2003.
[20] R. Milner. A theory of type polymorphism in programming. *J. Comput. Syst. Sci.*, 17(3):348–375, 1978.
[21] R. Paŕe and D. Schumacher. Abstract families and the adjoint functor theorems. In *Indexed Categories and their Applications*, Lect. Notes Math. 661, pages 1–125. Springer, 1978.
[22] B. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *J. ACM*, 47(3):531–584, 2000.
[23] G. Plotkin and J. Power. Notions of computation determine monads. In *FoSSaCS'02*, pages 342–356, 2002.
[24] R. A. G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *J. Symb. Log.*, 52(4):969–989, 1987.
[25] I. Stark. Free-algebra models for the π-calculus. *Theor. Comp. Sci.*, 390(2-3):248–270, 2008.
[26] S. Staton. Two cotensors in one: Presentations of algebraic theories for local state and fresh names. *Proc. of MFPS 25*, *ENTCS*, 249:471–490, 2009.