Multiversal Polymorphic Algebraic Theories
— Syntax, Semantics, Translations, and Equational Logic —

Marcelo Fiore
Computer Laboratory, University of Cambridge

Makoto Hamana
Dept. of Computer Science, Gunma University

***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 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 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 different type universes;**
- ***polymorphic structures* that provide a general notion of algebraic model (including the PL-category semantics of System F);**
- **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.

***Index Terms*—polymorphism, equational logic, presheaves, categorical semantics, the Grothendieck construction**

I. Introduction

The notion of polymorphism introduced by Strachey [32] is one of the most remarkable inventions in programming-language theory. The theory of polymorphism started with the polymorphic λ-calculus of Girard [13] and Reynolds [29], and led to Milner's striking application to functional programming [25]. Since then, the theory has deepened and its applicability spread broadly. Numerous systems have been extended to support polymorphism. Nowadays, it is not only supported in functional languages and proof assistants (e.g. ML, Haskell, Coq, Agda), but also other systems have been extended to cope with polymorphism, such as the π-calculus [27]. It has further been incorporated into object-oriented languages (e.g. Java, C++) where it is regarded as a key feature of generic programming. The range of applicability of polymorphism illustrates that, despite its origins, it is not necessarily based on λ-calculi.

But, *what are polymorphic calculi?*

In tackling this question, the aim of the paper is to establish an algebraic framework for analysing and reasoning about polymorphic systems generally. Indeed, we formalise and study the notion of *polymorphic algebraic theory*, as a formal theory presented by equations between polymorphically-typed terms. In doing so, we develop: signatures for polymorphic term constructors built on top of signatures for polymorphic types; algebraic theories giving rise to the syntax and semantics of polymorphic types and terms, and thereby to equational presentations and their models; and a sound and complete logical framework for equational reasoning about polymorphic algebraic theories.

Our approach is not based on a specific polymorphic λ-calculus. It is more general, and captures varieties of polymorphic systems that include extended λ-calculi as particular examples. The necessary background for our work follows.

1. Abstract syntax and variable binding. Our starting point is the algebraic model of abstract syntax with variable binding in presheaf categories [10]. The prototypical example is the syntax of untyped λ-terms:

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

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

Its abstract syntax is generated by three term constructors: the variable former, the application @, and the abstraction λ. The variable former is a nullary operation parameterised by the context, while @ is a binary function symbol; λ, however, is not merely a unary function symbol, as it binds a variable (and thereby decreases the context). To model the general phenomenon of variable binding (not only for λ-terms), Fiore, Plotkin and Turi [10] took the presheaf category $\mathbf{Set}^{\mathbb{F}}$ as universe of discourse. The category $\mathbb{F}$ has finite cardinals $\{1,\ldots,n\}$ ($n \in \mathbb{N}$) as objects, for which we henceforth abuse notation and simply write $n \in \mathbb{F}$, and all functions between them as morphisms. Intuitively, this is the category of contexts of nameless object variables (in the sense of de Bruijn [3]) and their renamings. An important result of [10] is that the abstract syntax with variable binding (up to α-equivalence) of any binding signature (viz. one with variable-binding function symbols) is characterised as the initial algebra of a prescribed endofunctor modelling a signature (e.g. for λ-terms).

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

A *Σ-algebra* for an endofunctor Σ is a pair (A, α) consisting of a carrier A and an algebra-structure map $\alpha : \Sigma A \to A$. The initial Σ_λ-algebra can be constructed inductively as the presheaf Λ of all λ-terms modulo α-equivalence. This explains 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 the renaming of free variables in a λ-term according to $\rho : n \to n'$ in $\mathbb{F}$ is modelled by the presheaf action $\Lambda(\rho) : \Lambda(n) \to \Lambda(n')$.

2. Object variables and metavariables. The above development was limited to the modelling of *object-level* abstract syntax. There is however also a need for considering a *meta-level*. We explain this with an example.

When developing a theory of λ-calculus, one uses both object and meta variables. For instance, in the mathematical vernacular (e.g. in the context of head normal forms), one may consider the λ-term

$$\lambda x.\, y\, M \ .$$

Here "x" and "y" are object-level variables, as the λ-calculus is the object language; while, at the level of text, "M" is a meta-level variable, standing for some λ-term. From the viewpoint of substitution, there is a crucial difference between object variables and metavariables. Because of α-equivalence, the operation of substituting a term for an object variable is not a simple textual substitution, e.g $(\lambda x.y\, M)[y := xx] = (\lambda x'.y\, M)[y := xx] = \lambda x'.(xx)\, M$ for fresh x'; while the substitution of a term for a metavariable is:

$$(\lambda x.\, y\, M)\{M \mapsto xx\} = \lambda x.\, y\,(xx) \ . \qquad (1)$$

Note that the object variable x is *captured* by the binder, something intended at the meta-level. Viewing these phenomena from the object and meta level viewpoints, the two classes of variables are classified by their respective substitution operations: capture-avoiding vs. possibly capturing.

3. Free Σ-monoids. Besides object-level abstract syntax, how can metavariables and the distinction between substitutions for object and meta variables be incorporated within the algebraic model of syntax with binding? This problem was explored in Hamana [16] and in Fiore [5, Part I].

A *Σ-monoid* (A, α, ν, μ) introduced by Fiore, Plotkin and Turi [10] consists of a Σ-algebra (A, α) and a monoid $(\nu : \mathrm{V} \to A,\ \mu : A \bullet A \to A)$ with respect to the substitution monoidal structure $(\mathrm{V}, \bullet)$ on $\mathbf{Set}^{\mathbb{F}}$ that is compatible with the algebra structure (i.e. $\mu \circ (\alpha \bullet \mathrm{id}) = \alpha \circ (\Sigma\mu) \circ \mathsf{strength}$). The unit ν models the variable former; while the multiplication μ models object-variable substitution (here the tensor product $\bullet$ gives the arity of substitution).

The key is to use *free Σ-monoids* on presheaves of *metavariables*. Importantly, the free Σ-monoid over a presheaf $X \in \mathbf{Set}^{\mathbb{F}}$, denoted $\mathcal{M}X$, is constructed inductively as an initial $\big(\mathrm{V} + \Sigma(-) + X \bullet (-)\big)$-algebra, and gives a presentation of syntax involving binding and *metavariables*. In this characterisation, the endofunctor $\mathrm{V} + \Sigma(-)$ models syntax with binding (as in § I-1) and the endofunctor $X \bullet (-)$ models the syntactic construct of metavariables:

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

where $\mathrm{M} \in X(k)$ is a *metavariable* and where the index k, referred to as the *arity* of M, designates k free variables in terms to be substituted for M. The terms $t_1, \ldots, t_k$ are replacements for these k free variables after instantiating M.

The freeness of $\mathcal{M}X$ states that every map $\theta : X \to A$, for A a Σ-monoid, uniquely extends to a Σ-monoid homomorphism $\theta^\sharp : \mathcal{M}X \to A$. It is here that the notion of *metavariable substitution* appears. Syntactically, one understands θ as an *assignment for meta-substitution* and $\theta^\sharp$ as the corresponding meta-substitution operation on terms involving metavariables from X. For instance, in the language of free Σ-monoids, example (1) above is formally recast as

$$\theta^\sharp\big(\, \lambda(x.\, y@\ \mathrm{M}[x])\,\big) = \lambda\big(x.\, y@\ (x@x)\,\big)$$

for $\theta = \{\mathrm{M} \mapsto 1@1\}$ where 1 denotes a free variable.

The syntactic theory of abstract syntax with variable binding and metavariables was introduced by Aczel [1]. This formal language allowed him to consider a general framework of rewrite rules for calculi with variable binding. This influenced Klop's Combinatory Reduction Systems [21]. Hamana clarified its algebraic semantics using Σ-monoids [17], extended it to simply-typed case [19], and considered equational logic and rewriting systems with variable binding [18]. Fiore et al. [8,9] considered algebraic theories for such *second-order* equational presentations. The model theory and equational logic of these was only developed for sorted (e.g. untyped and simply-typed) languages.

4. Polymorphic abstract syntax. The work [20] tackled the algebraic modelling of polymorphically-typed abstract syntax. A crucial departure from the many-sorted case [24,4] is the need for a dependent indexing structure on contexts and types to capture the algebraic structure of polymorphic terms. In a polymorphic system, such as System F, well-typed terms are formulated using judgments of the following form:

$$\begin{tikzcd}[column sep=small] \underset{\text{type context}}{n} \arrow[r, bend left] \arrow[rrr, bend left] \arrow[rrrrr, bend left] & \mid & \underset{\text{term context}}{\Gamma} & \vdash & \underset{\text{term}}{t} & : & \underset{\text{result type}}{\tau} \end{tikzcd} \qquad (2)$$

The arrows here indicate dependency, as type variables in $n \in \mathbb{F}$ (which refers to the set $\{1, \ldots, n\}$ of nameless type variables) may appear in any of the other parts of the judgment. This dependence is more complex than in the untyped case $n \vdash t$ (where only $n \in \mathbb{F}$ is a required index and therefore $\mathbf{Set}^{\mathbb{F}}$ suffices). To model such dependent context and type structures with respect to a universe of types $U \in \mathbf{Set}^{\mathbb{F}}$, [20] introduced the category $\mathsf{G}U$ of contexts and result types defined as

$$\begin{tikzcd} \boxed{\int^{n \in \mathbb{F}} \mathbb{F}{\downarrow}U(n) \times U(n)} \arrow[d, equal, "def"] \\ (\ n \ \mid \quad \Gamma \quad \vdash \ \tau\ ) \in \mathsf{G}U \end{tikzcd}$$

and identified the Grothendieck construction [15] (for which we use the notation $\int$) as the key to capture the dependence (2).

The universe of discourse $\mathbf{Set}^{\mathsf{G}U}$ was then shown to be appropriate for modelling polymorphically-typed object-level syntax [20], and Fiore [6] further isolated the notion of discrete generalised polynomial functor as a suitable mathematical structure for this purpose. Here U is the "type universe" (with $U(n)$ the set of types under the type context n) and $\mathbb{F}{\downarrow}(Un)$ is the category of term contexts with types in $U(n)$.

5. This paper. We develop polymorphic algebraic theories founded on these earlier works. We incorporate the notions of metavariables and Σ-monoids reviewed in §I-2 and I-3 into the algebraic polymorphic setting reviewed in §I-4. Syntactically, the framework encompasses meta-types (viz. types with type metavariables) and meta-terms (viz. terms with both meta-types and term metavariables). We explain why this is necessary next.

Consider the β-axiom of typed λ-calculus:

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

Is this a *single* axiom? One usually thinks so, but this is in fact a *schema* of axioms, as M and N are metavariables, and σ and τ are *metavariables for types*. Therefore, it 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 signatures. Following this line, then, λ-abstraction should also be regarded as *a family* of function symbols $\lambda_{\sigma,\tau} : (\sigma)\tau \to \sigma \Rightarrow \tau$ indexed by object types σ, τ. However, one usually keeps σ and τ as meta-level types (henceforth referred to as *meta-types*) and considers a single $\lambda_{\sigma,\tau}$ to develop a theory.

Here, we will precisely formulate the meta and object variable distinction. In polymorphic algebraic theory, 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 $\mathtt{S}$ and $\mathtt{T}$ are type metavariables, and where $\triangleright$ separates a metavariable context and the arity information of a function symbol using *meta-types* (as e.g. $\mathtt{S} \Rightarrow \mathtt{T}$). This leads to an important clarification. Up to this point, we have encountered *two* type universes for formalising a theory:

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

Clearly, the universes of object types and of meta-types must have similar mathematical structure. What is this precisely? Moreover, the semantics of the type universe must also have such a structure. This is the first step to answer the question posed at the beginning of the Introduction: "what are polymorphic calculi?". Namely, we need to clarify at first *what is a universe of polymorphic types*. Our answer is

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

for Σ a signature of type constructors. Note that $\mathcal{M}0$ is the initial Σ-monoid, while $\mathcal{M}\underline{S}$ is a free Σ-monoid. The mathematical theory naturally requires one to deal with such multiple universes, and it is in this sense that we refer to it as being *multiversal*.

6. Organisation. The paper is organised as follows. We introduce the notion of polymorphic signature in Section II and their corresponding endofunctors in Section III. We axiomatise type-in-term substitution in Section IV. We further define polymorphic structures as a general algebraic model in Section V. In Section VI, we present Polymorphic Equational Logic. Finally, in Section VII, we exemplify how polymorphic algebraic theories specify concrete calculi.

7. Future work. Various directions for further work are possible. A promising one is to apply the mathematical theory of the paper to mechanised formalisation (such as in Coq [2]).

Polymorphic algebraic theories of effects, which are here only exemplified, should be worked out in detail.

An algebraic theory for the π-calculus has been given [31]. Along this line, an algebraic theory for the polymorphic π-calculus [27] in the setting of polymorphic algebraic theories seems a challenging problem.

II. Type Universes and Polymorphic Signatures

We start motivating the development with the prototypical example of System F, illustrating and explaining how our notion of polymorphic signature specifies type and term structures.

Example II.1 (**System F [13,29]**) The polymorphic signature $\Sigma_{\mathrm{F}} = (\Sigma_{\mathrm{F}}^{\mathsf{Ty}}, \Sigma_{\mathrm{F}}^{\mathsf{Tm}})$ for System F consists of: a type signature $\Sigma_{\mathrm{F}}^{\mathsf{Ty}} = \{\ \mathsf{b} : *, \quad \Rightarrow : *, * \to *, \quad \forall : (*)* \to *\ \}$ which specifies type constructors. The signature $\Sigma_{\mathrm{F}}^{\mathsf{Tm}}$ for terms is

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

Let us now see how the term signature faithfully encodes the vernacular typing rules, considering the term and type abstraction 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 the abs term constructor is parameterised by two type metavariables $\mathtt{S}$ and $\mathtt{T}$, both of arity $*$ and thereby to be understood as representing types. The source arity $(\mathtt{S})\mathtt{T}$ of abs represents the premise of the term-abstraction rule, expressing that it consists of a term of type $\mathtt{T}$ in a context extended with a fresh term variable of type $\mathtt{S}$. Note that the informal metavariables σ and τ are respectively formalised by means of the formal metavariables $\mathtt{S}$ and $\mathtt{T}$. The target arity $\mathtt{S} \Rightarrow \mathtt{T}$ of abs represents the type of the term in the conclusion of the rule.

The arity of the tabs term constructor is parameterised by a metavariable $\mathtt{T}$ of arity $(*)*$ representing an open type with one type variable. The source arity $\langle\alpha\rangle\mathtt{T}[\alpha]$ of tabs represents the premise of the type-abstraction rule, expressing that it consists of a term of open type $\mathrm{T}[\alpha]$ in a context extended with a fresh type variable α. Function symbol declarations are meant to be instantiated to concrete cases. For instance, $\mathsf{abs} : (\mathrm{S})\mathrm{T} \to \mathrm{S} \Rightarrow \mathrm{T}$ is instantiated to $\mathsf{abs}_\theta : (\mathsf{nat})\mathsf{bool} \to \mathsf{nat} \Rightarrow \mathsf{bool}$ under $\theta = \{\mathrm{S} \mapsto \mathsf{nat}, \mathrm{T} \mapsto \mathsf{bool}\}$. The formal treatment follows.

Definition II.2 A *type signature* Σ^{Ty} is a set of type constructors with arities specified as $c : (*^{n_1})*, \cdots, (*^{n_\ell})* \to *$ with $n_i \in \mathbb{N}$. The intended meaning here is that of c taking ℓ arguments with the i-th argument binding n_i type variables.

Definition II.3 A *type metavariable* S of arity $n \in \mathbb{N}$ is declared as $\mathrm{S} : (*^n)*$, with the prefix omitted when $n = 0$. A set of type metavariables S corresponds to an $\mathbb{N}$-indexed set $\{S(k)\}_{k\in\mathbb{N}}$ where the $S(k)$ are the sets of metavariables in S of arity k. It can be considered as the presheaf $\underline{S} \in \mathbf{Set}^{\mathbb{F}}$ given by $\underline{S}(n) = \coprod_{k\in\mathbb{N}} S(k) \times \mathbb{F}(k, n)$.

A type signature is a *binding signature* in the sense of [10] and, as such, induces a signature functor Σ^{Ty} on $\mathbf{Set}^{\mathbb{F}}$ (cf. § I-1). We let $\mathcal{M}X$ denote the free Σ^{Ty}-monoid on a presheaf $X \in \mathbf{Set}^{\mathbb{F}}$ [16,5] (cf. § I-3), being particularly interested in this construction when performed on a presheaf of type metavariables.

For a set of type metavariable declarations S, the presheaf $\mathcal{M}\underline{S} \in \mathbf{Set}^{\mathbb{F}}$ consists of *meta-types* in context, with metavariables from S. We use the notation $(S \triangleright n \vdash \tau)$ for $\tau \in \mathcal{M}\underline{S}(n)$. Such a meta-type τ may contain metavariables from S and n type variables. Furthermore, we implicitly use the technique of de Bruijn levels [3] for representing abstract syntax with variable binding. For example, in the context of Example II.1, the meta-type $\forall(\alpha.\mathrm{T}[\alpha])$, where T is a type metavariable of arity 1, stands for $\forall(1.\mathrm{T}[1])$. Also, the source arity $\langle\alpha\rangle(\mathrm{T}[\alpha])$ of tabs stands for $\langle 1\rangle(\mathrm{T}[1])$.

Definition II.4 A *type universe* U for a type signature Σ^{Ty} is a Σ^{Ty}-monoid. A *morphism of type universes* is a Σ^{Ty}-monoid homomorphism.

A typical example of a type universe is $\mathcal{M}0$ (the universe of object types) and, more generally, $\mathcal{M}\underline{S}$ (the universe of meta-types with metavariables from S). A non-syntactic type universe features in the PL-category semantics of System F (Example VII.3).

As a notational convention, we use the vector notation $\overrightarrow{(-)}$ for sequences; the length function for these is denoted $|-|$.

Definition II.5 A *polymorphic signature* $\Sigma = (\Sigma^{\mathsf{Ty}}, \Sigma^{\mathsf{Tm}})$ consists of a type signature Σ^{Ty} with a term signature Σ^{Tm} given by a set of function symbols with arities of the form

$$S \triangleright f : \langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1, \ldots, \langle k_\ell\rangle(\overrightarrow{\sigma_l})\tau_\ell \to \tau \qquad (3)$$

where $\ell \in \mathbb{N}$ and S is a set of type metavariable declarations with respect to which $S \triangleright k_i \vdash \overrightarrow{\sigma_i}$ and $S \triangleright k_i \vdash \tau_i$ for all $1 \le i \le \ell$, and $S \triangleright 0 \vdash \tau$. The intended meaning here is that of f taking ℓ arguments with the i-th argument binding k_i type variables and $|\overrightarrow{\sigma_i}|$ term variables.

Example II.6 (**Polymorphic FPC [23]**) The type signature Σ^{Ty} is $\Sigma^{\mathsf{Ty}}_{\mathrm{F}}$ extended with $+, \times : *, * \to *$ and $\mu : (*)* \to *$. An excerpt of the term signature follows:

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

An important point to note above is that the source and target arities of function symbols are written in the language of free Σ^{Ty}-monoids [16,5] (cf. § I-3), rather than in an informal meta-language.

Example II.7 (**Existential λ-calculus [11]**) The type signature is given by

$$\bot : * \ , \quad \neg : * \to * \ , \quad \wedge : *, * \to * \ , \quad \exists : (*)* \to * \ .$$

As for the terms, we consider the following two key rules:

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

In the unpack rule, α does not appear free in Γ, τ. The term signature corresponding to these two rules is thus:

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

Note that the second argument of the function symbol unpack is specified under a type metavariable context with $\mathrm{T} : *$ thereby enforcing the side condition of the rule that α does not appear free in τ.

Example II.8 (**Global state**) The signature for a basic algebraic theory of global state [28] has type signature $\mathsf{L} : *$ for locations, $\mathsf{E} : *$ for expressions, and $\mathsf{Bool}, \mathsf{Nat} : *$ for Boolean and natural number values. The term signature

$$\begin{array}{l} \mathrm{V} : * \ \triangleright\ \mathsf{lookup}\ : \mathsf{L}, (\mathrm{V})\mathsf{E} \ \to\ \mathsf{E} \\ \mathrm{V} : * \ \triangleright\ \mathsf{update}\ : \mathrm{V}, \mathsf{L}, \mathsf{E} \ \to\ \mathsf{E} \end{array}$$

provides operations that are parameterised by types. This is unlike the original treatment [28], where the parameterisation is only treated informally. An algebraic theory over this signature is first-order with variable binding and polymorphism, without recourse to λ-calculi. This style may be useful for more flexible algebraic characterisation of effects [28].

As shown in the examples, polymorphic signatures are suitable for the specification of a wide variety of polymorphic languages. Indeed, the notion encompasses polymorphic types (with variable binding and type metavariables) in the broad sense of them being *variable types*, as referred to by Girard [14], together with polymorphic function symbols (parameterised by type metavariables) between them.

III. Polymorphic Signature Functors

This section presents the categorical semantics of polymorphic signatures. In the spirit of categorical algebra, this is done by associating signatures with endofunctors that interpret the arity of function symbols (§ III-D). As customary, then, models arise as endofunctor algebras.

A. Meta-substitution

Definition III.1 For a small category $\mathbb{C}$, we use the notation $|\mathbb{C}|$ for the set of its objects; while, for a presheaf $P \in \mathbf{Set}^{\mathbb{C}}$, we write $|P|$ for its underlying indexed set in $\mathbf{Set}^{|\mathbb{C}|}$. The left adjoint to the functor $|-| : \mathbf{Set}^{\mathbb{C}} \to \mathbf{Set}^{|\mathbb{C}|}$ is denoted by $\underline{(-)}$. (The definition of $\underline{S}$ in Def. II.3 is the instance of this construction for $\mathbb{C} = \mathbb{F}$.)

Definition III.2 Let S be a set of type metavariable declarations and let U be a type universe. An *assignment* $\theta : S \rightsquigarrow_n U$ ($n \in \mathbb{N}$) is an $\mathbb{N}$-indexed function $\theta : S \to |\delta^n U|$. Henceforth, since $[\underline{S} \Rightarrow U](n) \cong \mathbf{Set}^{\mathbb{N}}(S, |\delta^n U|)$, the exponential presheaf $[\underline{S} \Rightarrow U] \in \mathbf{Set}^{\mathbb{F}}$ will be regarded as consisting of assignments.

A main result of [5, Part I] established that the free Σ^{Ty}-monoid monad $\mathcal{M}$ is strong with respect to the cartesian closed structure of $\mathbf{Set}^{\mathbb{F}}$. This has two important consequences for us here: for every type universe U, the universal extension map internalises as a morphism $\mathcal{M}(X) \times [X \Rightarrow U] \to U$ in $\mathbf{Set}^{\mathbb{F}}$ and every presheaf $\delta^n U \cong [\,\mathrm{V}^n \Rightarrow U\,] \in \mathbf{Set}^{\mathbb{F}}$ ($n \in \mathbb{N}$) pointwise acquires a canonical type-universe structure. One thus obtains a *meta-substitution operation* (or *interpretation function*) $\mathcal{M}(\underline{S}) \times [\underline{S} \Rightarrow \delta^n U] \to \delta^n U$ of meta-types under assignments, for which we will use the following notation

$$S \rhd k \vdash \tau \;,\; \theta : S \rightsquigarrow_n U \quad\mapsto\quad \tau\theta \in U(n+k) \;\; .$$

B. Contexts for polymorphism

Following [20], the universes of discourse for the interpretation of polymorphic signatures will be categories of presheaves on small categories of contexts that arise from the Grothendieck construction [15].

The (covariant) *Grothendieck construction* on a functor $\mathcal{F} : \mathscr{F} \to \mathscr{C}$, for $\mathscr{C}$ a full subcategory of the large category of locally small categories and functors, is the category denoted $\int \mathcal{F}$ or $\int^{I \in \mathscr{F}} \mathcal{F}(I)$ with objects given by pairs $I \in \mathscr{F}$ and $A \in \mathcal{F}(I)$, and morphisms $(f, \varphi) : (I, A) \to (J, B)$ with $f : I \to J$ in $\mathscr{F}$ and $\varphi : \mathcal{F}(f)(A) \to B$ in $\mathcal{F}(J)$.

We use the Grothendieck construction for manufacturing categories of contexts. For a set T (of sorts), the category $\mathbb{F}{\downarrow}T$ (of T-sorted contexts) is the opposite of $\int^{n \in \mathbb{F}^{\mathrm{op}}} T^n$. Thus, it has objects $\Gamma : |\Gamma| \to T$ with $|\Gamma| \in \mathbb{F}$ and morphisms $\rho : \Gamma \to \Gamma'$ given by maps $\rho : |\Gamma| \to |\Gamma'|$ in $\mathbb{F}$ such that $\Gamma = \Gamma' \circ \rho : |\Gamma| \to T$. Every presheaf $T \in \mathbf{Set}^{\mathbb{F}}$ (of sorts in context) induces a functor $\mathbb{F}{\downarrow}(T-) : \mathbb{F} \to \mathbf{Cat}$, for $\mathbf{Cat}$ the category of small categories and functors. The category

$$\mathsf{F}(T) \overset{def}{=} \int^{n \in \mathbb{F}} \mathbb{F}{\downarrow}(Tn)$$

(of T-sorted contexts) has objects $(n|\Gamma)$ with $n \in \mathbb{F}$ and $\Gamma \in \mathbb{F}{\downarrow}(Tn)$, and morphisms $(\rho, \pi) : (n|\Gamma) \to (n'|\Gamma')$ with $\rho : n \to n'$ in $\mathbb{F}$ and $\pi : \big((T\rho) \circ \Gamma\big) \to \Gamma'$ in $\mathbb{F}{\downarrow}(Tn')$.

We will also need to consider indexed versions of this construction. To this end, for a presheaf $U \in \mathbf{Set}^{\mathbb{F}}$, we define

$$\mathsf{H}(T, U) \overset{def}{=} \int^{n \in \mathbb{F}} \mathbb{F}{\downarrow}(Tn) \times Un \;.$$

This category has objects $(n \mid \Gamma \vdash \tau)$ with $(n \,|\, \Gamma) \in \mathsf{F}T$ and $\tau \in Un$, and morphisms $(n \mid \Gamma \vdash \tau) \to (n' \mid \Gamma' \vdash \tau')$ given by maps $(\rho, \pi) : (n|\Gamma) \to (n'|\Gamma')$ in $\mathsf{F}T$ such that $(U\rho)\tau = \tau'$. The *category* $\mathsf{G}U$ *for contexts and result types* (cf. § I-4) is defined as $\mathsf{H}(U, U)$.

C. Generalised polynomial functors

A central technical tool in our development is the notion of generalised polynomial functor introduced in [6]. This fits a general abstract scheme for defining polynomial constructions that also incorporates the notion of dependent polynomial functor [26,12]. This section reviews the basics, deferring details to [6]. Recall that every $f : \mathbb{X} \to \mathbb{Y}$ in $\mathbf{Cat}$ induces the adjoint situations

$$f_! \dashv f^* \dashv f_* : \mathbf{Set}^{\mathbb{X}} \to \mathbf{Set}^{\mathbb{Y}}$$

where $f^* = (-) \circ f$ and $f_!$ and f_* are respectively given by left and right Kan extending along f [15,22]. For instance, the adjunction $\underline{(-)} \dashv |-| : \mathbf{Set}^{\mathbb{C}} \to \mathbf{Set}^{|\mathbb{C}|}$ introduced in Definition III.1 amounts to $i_! \dashv i^*$ for i the inclusion $|\mathbb{C}| \to \mathbb{C}$.

A *polynomial* P is a diagram $\mathbb{A} \xleftarrow{s} \mathbb{I} \xrightarrow{a} \mathbb{J} \xrightarrow{t} \mathbb{B}$ in $\mathbf{Cat}$. The *generalised polynomial functor* induced by P is

$$F_P = t_!\, a_*\, s^* : \mathbf{Set}^{\mathbb{A}} \to \mathbf{Set}^{\mathbb{B}} \;. \tag{4}$$

A polynomial diagram is *discrete* when its component $a : \mathbb{I} \to \mathbb{J}$ is of the form $\coprod_{i \in I} \nabla_{L_i} : \coprod_{i \in I} L_i \cdot \mathbb{C}_i \to \coprod_{i \in I} \mathbb{C}_i$ for a set I and finite sets L_i, where $L \cdot \mathbb{C} = \coprod_{\ell \in L} \mathbb{C}$ and $\nabla_L : L \cdot \mathbb{C} \to \mathbb{C}$ is the codiagonal $[\mathrm{Id}_{\mathbb{C}}]_{\ell \in L}$. As we will see, discrete generalised polynomial functors provide an expressive and flexible formalism. Indeed, since they admit inductive constructions of free algebras [6, Prop. 5.1], in the following two sections we will use them to model polymorphic signatures (§ III-D) and type-in-term substitution (§ IV).

D. Polymorphic signature functors

We define the signature functor corresponding to a polymorphic signature Σ. This is done on a type universe U for Σ^{Ty} on the universe of discourse given by the presheaf category $\mathbf{Set}^{\mathsf{G}U}$.

To explain the approach, we start by analysing concrete cases in System F (Example II.1). For the term abstraction function symbol

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

the corresponding operation of an algebra $A \in \mathbf{Set}^{\mathsf{G}U}$ should be given by a natural map

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

using an assignment $\theta = \{\mathrm{S} \mapsto \sigma, \mathrm{T} \mapsto \tau\}$, where $\mathrm{S} \Rightarrow \mathrm{T}$ is interpreted as $\sigma \Rightarrow^U \tau$. Some operations may also change the context for type variables, as in the case of the type abstraction:

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

To obtain these kind of algebraic operations from the arity specification of function symbols, one needs to formulate the instantiation of meta-types in arities by types in the universe U by means of meta-substitutions, and incorporate the extension of contexts for type variables as prescribed by source arities. Technically, this requires the use of the Grothendieck construction $\mathsf{H}\big(U,[\underline{S}\Rightarrow U]\big)$ of § III-B, for S a set of metavariable declarations.

Definition III.3 Let Σ be a polymorphic signature and U a type universe for Σ^{Ty}. To every function symbol $f\in\Sigma^{\mathsf{Tm}}$ we associate the following discrete polynomial P_f

$$\mathsf{G}U \xleftarrow{s_f} \ell\cdot\mathsf{H}\big(U,[\underline{S}\Rightarrow U]\big) \xrightarrow{\nabla_\ell} \mathsf{H}\big(U,[\underline{S}\Rightarrow U]\big) \xrightarrow{t_f} \mathsf{G}U$$

where the source and target functors are respectively given by $s_f\big(i,(n\mid\Gamma\vdash\theta)\big) = \big(n+k_i \mid \big(\Gamma_i+\langle\overrightarrow{\sigma_i}\theta_i\rangle\big)\vdash\tau_i\theta_i\big)$ and $t_f(\,n\mid\Gamma\vdash\theta\,) = (\,n\mid\Gamma\vdash\tau\theta\,)$, where Γ_i and θ_i respectively arise from Γ and θ by the presheaf action along the coproduct injections $k_i\hookrightarrow n+k_i$.

The *signature functor* Σ on $\mathbf{Set}^{\mathsf{G}U}$ is defined as the discrete generalised polynomial functor $\coprod_{f\in\Sigma^{\mathsf{Tm}}}F_{P_f}$. This is explicitly given by

$$\Sigma A(\,n\mid\Gamma\vdash u\,) = \coprod_{f,\theta}\prod_{i\in\ell}A(\,n+k_i\mid\Gamma_i,\overrightarrow{\sigma_i}\theta_i\vdash\tau_i\theta_i\,)$$

where the coproduct ranges over function symbols $(S\rhd f:\langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1,\ldots,\langle k_\ell\rangle(\overrightarrow{\sigma_\ell})\tau_\ell\to\tau)$ in Σ^{Tm} and assignments $\theta:S\rightsquigarrow_n U$ such that $\tau\theta=u\in U(n)$.

IV. Type-in-term Substitution

The monoid multiplication of a type universe provides an operation that models simultaneous capture-avoiding type-in-type substitution. Polymorphic terms have type variables and the need for instantiating them leads to a type-in-term substitution operation that should be modelled. We address this issue by giving an algebraic axiomatisation of type-in-term substitution.

Let (U,ν,μ) be a type universe. For $\tau\in U(n+\ell)$ and $\sigma_i\in U(n)$ with $1\le i\le\ell$, we denote by $\tau\{\sigma_1,\ldots,\sigma_\ell\}$ the multi-variable capture-avoiding substitution $\mu_n(\tau;\nu_n(1),\ldots,\nu_n(n),\sigma_1,\ldots,\sigma_\ell)\in U(n)$; pointwise extending the notation to $\Gamma\{\sigma_1,\ldots,\sigma_\ell\}\in\mathbb{F}\downarrow(Un)$ for $\Gamma\in\mathbb{F}\downarrow\big(U(n+\ell)\big)$. Our aim is to define algebraic structure on a presheaf $A\in\mathbf{Set}^{\mathsf{G}U}$ amounting to functions

$$\varsigma_n(-,\sigma):A(n+1\mid\Gamma\vdash\tau)\to A\big(n\mid\Gamma\{\sigma\}\vdash\tau\{\sigma\}\big)\qquad(5)$$

with $(n\mid\Gamma\vdash\tau)\in\mathsf{G}U$ and $\sigma\in U(n)$, that we will later on axiomatise so as to model the substitution of a distinguished type variable by a type. We again invoke the formalism of generalised polynomial functors and consider the discrete polynomial P

$$\mathsf{G}U\xleftarrow{s}\mathsf{H}(\delta U,\delta U\times U)\xrightarrow{t}\mathsf{G}U$$

with $s(n\mid\Gamma\vdash\tau,\sigma)=(n+1\mid\Gamma\vdash\tau)$ and $t(n\mid\Gamma\vdash\tau,\sigma)=(n\mid\Gamma\{\sigma\}\vdash\tau\{\sigma\})$. We write $\uparrow$ for the discrete generalised polynomial functor F_P on $\mathbf{Set}^{\mathsf{G}U}$ induced by P, noting that algebra-structure maps $\uparrow A\to A$ amount to (5).

The axioms for type-in-term substitution are analogous to the ones given for substitution algebras in [10]. To present them, we need introduce the following notation: for $\rho:\ell\to m$ in $\mathbb{F}$, $U\in\mathbf{Set}^{\mathbb{F}}$, and $A\in\mathbf{Set}^{\mathsf{G}U}$, we let $\rho^U_n = U(n+\rho):U(n+\ell)\to U(n+m)$ and $\rho^A_n = A(n+\rho,\mathrm{id}):A(n+\ell\mid\Gamma\vdash\tau)\to A\big(n+m\mid\rho^U_n\circ\Gamma\vdash\rho^U_n(\tau)\big)$; and consider the weakening, contraction, and swapping maps $\mathsf{up}:0\to1$, $\mathsf{con}:2\to1$, and $\mathsf{sw}:2\to2$ in $\mathbb{F}$.

Definition IV.1 Let (U,ν,μ) be a type universe. A *type-in-term substitution* for $A\in\mathbf{Set}^{\mathsf{G}U}$ is an algebra $\varsigma:\uparrow A\to A$ in $\mathbf{Set}^{\mathsf{G}U}$ subject to the following axioms:

$$\begin{array}{l}
a\in A(n\mid\Gamma\vdash\tau),\ \sigma\in U_n\vdash\quad\varsigma_n(\mathsf{up}^A_n a,\ \sigma)=a\\
a\in A(n+2\mid\Gamma\vdash\tau)\qquad\quad\vdash\varsigma_{n+1}(a,\mathsf{new}^U_n)=\mathsf{con}^A_n\,a\\
a\in A(n+2\mid\Gamma\vdash\tau),\ \sigma\in U(n+1),\ \sigma'\in U(n)\\
\ \vdash\varsigma_n(\varsigma_{n+1}(a,\sigma),\ \sigma')=\varsigma_n(\varsigma_{n+1}(\mathsf{sw}^A_n\,a,\mathsf{up}^U_n\,\sigma'),\ \sigma\{\sigma'\})
\end{array}$$

where $\mathsf{new}:1\to\delta U$ (a generic new variable) is the transpose of the point $\nu:\mathrm{V}\to U$.

The axioms have an intuitive reading; e.g. the first one expresses that substituting for a type variable not free in a term does not affect the term.

The presheaf of object variables $\mathrm{V}\in\mathbf{Set}^{\mathsf{G}U}$ is defined as $\mathrm{V}(n\mid\Gamma\vdash\tau)=\big(\mathbb{F}\downarrow Un\big)\big(\langle\tau\rangle,\Gamma\big)\cong\{\,x\in|\Gamma|:\Gamma(x)=\tau\,\}$. It has a type-in-term substitution structure $\varsigma^{\mathrm{V}}:\ \uparrow\mathrm{V}\to\mathrm{V}$ given by maps that interpret a variable of type τ in context Γ as one of type $\tau\{\sigma\}$ in context $\Gamma\{\sigma\}$.

V. Polymorphic Structures

We introduce polymorphic structures. They provide a general basic notion of algebraic model, and thereby of abstract syntax, for polymorphic algebraic theories.

In the vein of [24,4,20], for $A,B\in\mathbf{Set}^{\mathsf{G}U}$, we define $(A\bullet B)\in\mathbf{Set}^{\mathsf{G}U}$ by the coend $(A\bullet B)(n\mid\Gamma\vdash\tau)=\int^{\Delta\in\mathbb{F}\downarrow Un}A(n\,|\,\Delta\vdash\tau)\times\prod_{1\le i\le|\Delta|}B\big(n\,|\,\Gamma\vdash\Delta(i)\big)$ remarking that $(\mathrm{V},\bullet)$ provides a monoidal structure on $\mathbf{Set}^{\mathsf{G}U}$ suitable for considering monoids with polymorphic algebraic structure.

Definition V.1 Let Σ be a signature and U a type universe. A (Σ,U)*-polymorphic structure* consists of

(i) a Σ-monoid (A,α,ν,μ) in $\mathbf{Set}^{\mathsf{G}U}$ for Σ the signature functor on $\mathbf{Set}^{\mathsf{G}U}$ obtained by Def. III.3, and

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

that are compatible in the sense that the following commute

$$\begin{tikzcd}
\uparrow\Sigma A \arrow[r] \arrow[d, "\uparrow\alpha"'] & \Sigma\uparrow A \arrow[r, "\Sigma\varsigma"] & \Sigma A \arrow[d, "\alpha"] \\
\uparrow A \arrow[rr, "\varsigma"'] & & A
\end{tikzcd}$$

$$\begin{tikzcd}
\uparrow\mathrm{V} \arrow[r, "\varsigma^{\mathrm{V}}"] \arrow[d, "\uparrow\nu"'] & \mathrm{V} \arrow[d, "\nu"] \\
\uparrow A \arrow[r, "\varsigma"'] & A
\end{tikzcd}
\qquad
\begin{tikzcd}
\uparrow(A\bullet A) \arrow[r] \arrow[d, "\uparrow\mu"'] & \uparrow A\bullet\uparrow A \arrow[r, "\varsigma\bullet\varsigma"] & A\bullet A \arrow[d, "\mu"] \\
\uparrow A \arrow[rr, "\varsigma"'] & & A
\end{tikzcd}$$

where $\uparrow\Sigma(-)\to\Sigma\uparrow(-)$ and $\uparrow(-\bullet=)\to\uparrow(-)\bullet\uparrow(=)$ are appropriate coercion maps.

$$(\mathrm{Var})\ \frac{(x:\tau)\in\Gamma}{x\in N_U Z(n\mid\Gamma\vdash\tau)} \qquad (\mathrm{Fun})\ \frac{\begin{array}{l}(S\ \triangleright\ f:\langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1,\ldots,\langle k_\ell\rangle(\overrightarrow{\sigma_\ell})\tau_\ell\to\tau)\in\Sigma^{\mathsf{Tm}} \qquad k_i=|\overrightarrow{\alpha_i}| \\ \theta:S\rightsquigarrow_n U \qquad \theta_i=\imath_i^{[\underline{S}\Rightarrow U]}(\theta)\quad(\imath_i:n\hookrightarrow n+k_i) \\ t_i\in N_U Z(n+k_i\mid\Gamma,\ \overrightarrow{x_i}:\overrightarrow{\sigma_i}\theta_i\vdash\tau_i\theta_i)\quad(1\le i\le\ell)\end{array}}{f_\theta(\overrightarrow{\alpha_1}.\ \overrightarrow{x_1}.t_1,\ldots,\overrightarrow{\alpha_l}.\ \overrightarrow{x_l}.t_l)\in N_U Z(n\mid\Gamma\vdash\tau\theta)}$$

$$(\mathrm{MVar})\ \frac{\begin{array}{ll}\mathrm{z}\in Z(n+k\mid x_1:\tau_1,\ldots,x_\ell:\tau_\ell\vdash\tau) & \overrightarrow{\sigma}=\sigma_1,\ldots,\sigma_k\in U(n) \\ t_i\in N_U Z\big(n\mid\Gamma\vdash\tau_i\{\overrightarrow{\sigma}\}\big) & (1\le i\le\ell)\end{array}}{\mathrm{z}\lfloor\overrightarrow{\sigma}\rfloor[t_1,\ldots,t_\ell]\in N_U Z\big(n\mid\Gamma\vdash\tau\{\overrightarrow{\sigma}\}\big)}$$

NB: In (Fun), for $S=\big(\mathrm{s}_i:(*^{n_i})\,*\,\big)_{1\le i\le m}$, the formal concretion f_θ may be denoted $f_{\theta(\mathrm{s}_1),\ldots,\theta(\mathrm{s}_m)}$ or simply f when the omission is inferable from the context. These function symbols bind both type and term variables and we implicitly assume the technique of de Bruijn levels [3] for their representation. Each θ_i is essentially the same substitution as θ, shifted to the index $n+k_i$ to apply it under the binders $\overrightarrow{\alpha_i}$. A meta-term $\mathrm{z}\lfloor\overrightarrow{\sigma}\rfloor[\overrightarrow{t}]$ may be denoted by $\mathrm{z}[\overrightarrow{t}]$ when $|\overrightarrow{\sigma}|=0$, and by $\mathrm{z}\lfloor\overrightarrow{\sigma}\rfloor$ when $|\overrightarrow{t}|=0$.

Fig. 1. Rules for $N_U Z$

Morphisms of (Σ,U)-polymorphic structures, referred to as (Σ,U)-*polymorphic translations*, are maps that are both Σ-monoid morphisms and $\uparrow$-algebra homomorphisms.

The category of (Σ,U)-polymorphic structures and (Σ,U)-polymorphic translations is denoted $\mathbf{Poly}(\Sigma,U)$.

A. Free polymorphic structures

From the theories of equational systems [7] and of discrete generalised polynomial functors [6] we have the following result.

Theorem V.2 *The forgetful functor $\mathbf{Poly}(\Sigma,U)\to\mathbf{Set}^{\mathsf{G}U}$ is monadic.*

We outline a construction of free (Σ,U)-polymorphic structures, obtaining the underlying indexed set of the free polymorphic structure $\mathcal{N}_U X\in\mathbf{Set}^{\mathsf{G}U}$ on a presheaf $X\in\mathbf{Set}^{\mathsf{G}U}$ as a quotient of an indexed set $N_U|X|\in\mathbf{Set}^{|\mathsf{G}U|}$.

For $Z\in\mathbf{Set}^{|\mathsf{G}U|}$, the indexed set $N_U Z\in\mathbf{Set}^{|\mathsf{G}U|}$ has syntactic character and is defined by the rules of Fig. 1. In this context, we let $|\mathcal{N}_U X|$ be given by the quotient of $N_U|X|$ under the congruence generated by the identification

$$\mathrm{z}\lfloor\overrightarrow{\sigma}\rfloor[t_{\pi 1},\ldots,t_{\pi|\Delta|}]\ =\ \mathrm{z}'\lfloor\overrightarrow{\sigma}\rfloor[t_1,\ldots,t_{|\Delta'|}] \qquad (6)$$

for $n\in\mathbb{F}$, $\pi:\Delta\to\Delta'$ in $\mathbb{F}\!\downarrow\!(Un)$, $\mathrm{z}\in X(n{+}|\overrightarrow{\sigma}|\mid\Delta\vdash\tau)$, and $\mathrm{z}'=X(\mathrm{id},\pi)(\mathrm{z})\in X(n{+}|\overrightarrow{\sigma}|\mid\Delta'\vdash\tau)$. This quotient arises from the coend formula of the tensor product $\bullet$. The presheaf and polymorphic structures of $\mathcal{N}_U X$ are essentially given syntactically.

B. Polymorphic abstract syntax with metavariables

We now consider free (Σ,U)-polymorphic structures that correspond to *polymorphic abstract syntax with metavariables*. First, one constructs a syntactic type universe U as a free Σ^{Ty}-monoid. On top of this, one constructs polymorphic abstract syntax as a free (Σ,U)-polymorphic structure over generators given by indexed sets of term metavariables.

An indexed set $Z\in\mathbf{Set}^{|GU|}$ is regarded as consisting of metavariable declarations, where $\mathrm{z}\in Z(n\,|\,\overrightarrow{\sigma}\vdash\tau)$ stands for the arity declaration $\mathrm{z}:\langle n\rangle(\overrightarrow{\sigma})\tau$. Such metavariables are to be instantiated by terms involving n type variables and variables of types $\overrightarrow{\sigma}$.

Every indexed set $Z\in\mathbf{Set}^{|\mathsf{G}U|}$ freely gives rise to the presheaf $\underline{Z}\in\mathbf{Set}^{\mathsf{G}U}$. In view of the following characterisation, the (Σ,U)-polymorphic structure $\mathcal{N}_U(\underline{Z})\in\mathbf{Set}^{\mathsf{G}U}$ is given in purely syntactic terms: $|\mathcal{N}_U\underline{Z}|\ \cong\ N_U\overline{Z}$ for $\overline{Z}(n\mid\Gamma\vdash\tau)$ given by

$$\coprod_{\substack{(k\,|\,\Delta\,\vdash\sigma)\,\in\,\mathsf{G}U\\ \rho\,\in\,\mathbb{F}(k,n)}}\big[U\rho\sigma\equiv\tau\big]\times\big[(U\rho)\circ\Delta\equiv\Gamma\big]\times Z\big(k\mid\Delta\vdash\sigma\big).$$

We refer to elements of $\mathcal{N}_U\underline{Z}(n\mid\Gamma\vdash\tau)$ as *meta-terms*.

We will introduce next the notion of assignment and their induced interpretation functions, yielding operations of meta-substitution (i.e. substitution for metavariables in meta-terms).

For $(k|\Delta)\in\mathsf{F}U$ (cf. § III-B), the endofunctor $\delta^{(k|\Delta)}$ on $\mathbf{Set}^{\mathsf{G}U}$ is defined as $\big(d(k|\Delta)\big)^*$ for $d(k|\Delta)$ the endofunctor on $\mathsf{G}U$ given by $d(k|\Delta)(n\,|\,\Gamma\vdash\tau)=(\ k+n\ \mid\ \Delta,\Gamma\vdash\jmath(\tau)\ )$ where $(\Delta,\Gamma)=(U\imath\circ\Delta)+(U\jmath\circ\Gamma)$ for $\imath$ and $\jmath$ the first and second coproduct injections into $k+n$.

Definition V.3 Let Z be a set of metavariable declarations and A a polymorphic structure. An *assignment* for $\vartheta:Z\rightsquigarrow_{(k|\Delta)}A$, where $(k|\Delta)\in\mathsf{F}U$, is an $|\mathsf{G}U|$-indexed function $\vartheta:Z\to|\delta^{(k|\Delta)}A|$.

For every (Σ,U)-polymorphic structure A, the presheaf $\delta^{(k|\Delta)}A$ inherits a polymorphic structure. As in the case of the extension to a Σ-monoid morphism explained in § I-3, by the universal property of $\mathcal{N}_U\underline{Z}$, an assignment ϑ uniquely extends to a (Σ,U)-polymorphic translation $\vartheta^\sharp$ giving the interpretation function of a meta-term in a polymorphic structure A. This is explicitly defined as follows.

Definition V.4 (**Interpretation function**) For a (Σ,U)-polymorphic structure $(A,\alpha,\nu,\mu,\varsigma)$, the universal extension of a morphism $\vartheta:X\to A$ in $\mathbf{Set}^{\mathsf{G}U}$ is the (Σ,U)-polymorphic translation $\vartheta^\sharp:\mathcal{N}_U\underline{Z}\to A$ given as follows:

$$\vartheta^\sharp:\mathcal{N}_U\underline{Z}(n\ \mid\ \Gamma\vdash\tau)\longrightarrow A(n\ \mid\ \Gamma\vdash\tau)$$

$$\vartheta^\sharp(x) = \nu(x)$$
$$\vartheta^\sharp(f(\overrightarrow{\alpha_1}.\overrightarrow{x_1}.t_1,\dots,\overrightarrow{\alpha_\ell}.\overrightarrow{x_\ell}.t_\ell)) = f^A(\,\vartheta^\sharp(t_1),\dots,\ \vartheta^\sharp(t_\ell)\,)$$
$$\vartheta^\sharp(\mathrm{z}\lfloor\overrightarrow{\sigma}\rfloor[\overrightarrow{t}\,]) = \mu\big(\varsigma(\,\vartheta(\mathrm{z}),\overrightarrow{\sigma}\ );\ \overrightarrow{\vartheta^\sharp(t)}\big)$$

where f^A is the algebra operation for f in (A,α) and where, extending the notation introduced in (5), for $a \in A(n+\ell \mid \Gamma \vdash \tau)$ and $\sigma_i \in U(n)$ with $1 \le i \le \ell$, we set $\varsigma(a,\sigma_1,\dots,\sigma_\ell) = \varsigma\big(\cdots\varsigma(a,\imath_1(\sigma_1))\cdots,\imath_\ell(\sigma_\ell)\big)$ for $\imath_i$ the coproduct injection $n \hookrightarrow n+\ell-i$.

C. *The multiverse of polymorphic structures*

We combine all (Σ,U)-polymorphic structures, varying over all type universes U into a single category **Poly**(Σ). This category abstractly arises as the Grothendieck construction

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

applied to a functor $\textbf{Poly}(\Sigma,-) : \Sigma^{\mathsf{Ty}}\text{-}\textbf{Mon}^{\mathrm{op}} \to \textbf{CAT};\ U \mapsto \textbf{Poly}(\Sigma,U)$. This definition relies on that fact that if (U',A') is a Σ-polymorphic structure then so is $(U,\ A'\mathsf{G}\llparenthesis - \rrparenthesis)$ for every morphism of type universes $\llparenthesis - \rrparenthesis : U \to U'$.

Definition V.5 A *Σ-polymorphic structure* (U,A) consists of a type universe U and a (Σ,U)-polymorphic structure A.

A *Σ-polymorphic translation* between Σ-polymorphic structures

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

consists of a pair of maps:
- a morphism $\llparenthesis - \rrparenthesis : U \to U'$ of type universes, and
- a (Σ,U)-polymorphic translation $\varphi : A \longrightarrow A'\mathsf{G}\llparenthesis - \rrparenthesis$.

The idea behind these maps is that they provide two-levelled translations: $\llparenthesis - \rrparenthesis$ translates types in U to U' and φ translates elements from A to A' taking a shift of universe into account. The category **Poly**(Σ) consists of Σ-polymorphic structures and Σ-polymorphic translations.

Universe shift. The category **Poly**(Σ) supports a notion of translation between polymorphic structures on possibly different universes. As an application, we indicate how one can transport a meta-term on a universe to another one.

Let $\llparenthesis - \rrparenthesis : W \to U$ be a morphism of type universes, translating types $\tau \in W(n)$ to types $\llparenthesis\tau\rrparenthesis \in U(n)$. The functor $\mathsf{F}\llparenthesis - \rrparenthesis : \mathsf{F}W \to \mathsf{F}U$ translates thus contexts $\Gamma \in \mathbb{F}{\downarrow}(Wn)$ to contexts $\llparenthesis\Gamma\rrparenthesis = \big((\mathbb{F}{\downarrow}\llparenthesis - \rrparenthesis_n) \circ \Gamma\big) \in \mathbb{F}{\downarrow}(Un)$. Furthermore, the functor $\big|\mathsf{G}\llparenthesis - \rrparenthesis\big|_! : \textbf{Set}^{|\mathsf{G}W|} \to \textbf{Set}^{|\mathsf{G}U|}$ translates a set of metavariable declarations Z on the universe W to the set of metavariable declarations $\llparenthesis Z\rrparenthesis$ on the universe U given by $\mathrm{z} \in \llparenthesis Z\rrparenthesis\big(\,n \mid \llparenthesis\Gamma\rrparenthesis \vdash \llparenthesis\tau\rrparenthesis\,\big) \iff \mathrm{z} \in Z(\,n \mid \Gamma \vdash \tau\,)$. Finally, we have the assignment $Z \to \big|\mathcal{N}_U\underline{\llparenthesis Z\rrparenthesis}\big(\mathsf{G}\llparenthesis - \rrparenthesis\big)\big|$, mapping the metavariable $\mathrm{z} \in Z(n \mid \overrightarrow{x} : \overrightarrow{\sigma} \vdash \tau)$ to the meta-term $\mathrm{z}[\overrightarrow{x}] \in \mathcal{N}_U\underline{\llparenthesis Z\rrparenthesis}\big(n \mid \overrightarrow{x} : \overrightarrow{\llparenthesis\sigma\rrparenthesis} \vdash \llparenthesis\tau\rrparenthesis\big)$, that induces the polymorphic translation $\llparenthesis - \rrparenthesis : \mathcal{N}_W\underline{Z}(\,n \mid \Gamma \vdash \tau) \to \mathcal{N}_U\underline{\llparenthesis Z\rrparenthesis}\big(\,n \mid \llparenthesis\Gamma\rrparenthesis \vdash \llparenthesis\tau\rrparenthesis\big)$ whose effect is to translate types within meta-terms.

VI. Polymorphic Equational Logic

We introduce Polymorphic Equational Logic (PEL): a sound and complete logical framework for equational reasoning about polymorphic algebraic theories.

The *equational judgments* of PEL are of the form

$$Z \,\triangleright\, n \mid \Gamma \vdash_W s = t : \tau \tag{7}$$

where W is a type universe, Z is a set of metavariable declarations and $s,t \in \mathcal{N}_W\underline{Z}(n \mid \Gamma \vdash \tau)$ are meta-terms. When W is a universe of meta-types $\mathcal{M}\underline{S}$ for a set of type metavariable declarations S (as e.g. in Figs. 3, 4, 5), equational judgments are written as $S \mid Z \,\triangleright\, n \mid \Gamma \vdash s = t : \tau$. Sets of equational judgments are referred to as *axioms*.

Definition VI.1 A Σ-polymorphic structure (U,A) *satisfies* an equational judgment $Z \,\triangleright\, n \mid \Gamma \vdash_W s = t : \tau$ if
- for all morphisms of type universes $\llparenthesis - \rrparenthesis : W \to U$, and
- for all assignments $\vartheta : Z \rightsquigarrow_{(k|\Delta)} A(\mathsf{G}\llparenthesis - \rrparenthesis)$,

$$\vartheta^\sharp(s) \ = \ \vartheta^\sharp(t)$$

holds in $A(k+n \mid \llparenthesis\Gamma,\Delta\rrparenthesis \vdash \llparenthesis\tau\rrparenthesis)$. An *algebraic model* of a set of axioms E is a polymorphic structure that satisfies all the equational judgments in E.

A. *PEL*

The deduction system of PEL is given by the inference rules of Fig. 2. The notation $[x := t]$ used in the (Sub) rule stands for the capture-avoiding substitution of the object variable x by the meta-term t. The application $\rho(s)$ (resp. $\pi(s)$) replaces type variables (resp. variables) in s by ρ (resp. π). PEL has been synthesised from the mathematical model of polymorphic structures on type universes using the following ideas.

(Ref), (Sym), (Tra): The rules of equivalence relations.
(Ax): An axiom becomes a theorem.
(USh): Universe shift with respect to the morphism $\llparenthesis - \rrparenthesis : \mathcal{N}_W\underline{Z}(\,n \mid \Gamma \vdash \tau) \to \mathcal{N}_U\underline{\llparenthesis Z\rrparenthesis}\big(\,n \mid \llparenthesis\Gamma\rrparenthesis \vdash \llparenthesis\tau\rrparenthesis\big)$.
(Sub): Multiplication of the Σ-monoid $\mathcal{N}_U\underline{Z}$.
(Fun): Σ-algebra structure of $\mathcal{N}_U\underline{Z}$.
(MSub): Polymorphic translation $\vartheta^\sharp : \mathcal{N}_U\underline{Z} \to \delta^{(k|\Delta)}\mathcal{N}_U\underline{Z}'$.
(MVar): $(Z \bullet -)$-algebra structure of $\mathcal{N}_U\underline{Z}$.
(Act): Presheaf action of $\mathcal{N}_U\underline{Z} \in \textbf{Set}^{\mathsf{G}U}$ (cf. [20, § 3.2 Meaning of arrows]).
(TSub): Type-in-term substitution on $\mathcal{N}_U\underline{Z}$.

Note that besides the equivalence-relation and axiom rules common to all equational deductive systems, the above consist of congruence rules that reflect the various algebraic aspects of polymorphic structures.

B. *Soundness and completeness*

Let U be a type universe. For a set of axioms E, define Z to be the disjoint union of all the translations of metavariable declarations in the axioms of E into U. Consider then the term polymorphic structure $\mathcal{N}_U\underline{Z}$ on U. Derivable equations from E define an $|\mathsf{G}U|$-indexed equivalence relation $=_E$ on $|\mathcal{N}_U\underline{Z}|$, and we set $\mathcal{N}_U^E\underline{Z}(n \mid \Gamma \vdash \tau) = \mathcal{N}_U\underline{Z}(n \mid \Gamma \vdash \tau)/{=_E}$.

Lemma VI.2 *$(U,\mathcal{N}_U^E\underline{Z})$ is an algebraic model of E.*

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

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

$$\text{(Sub)}\ \frac{\begin{array}{c} Z \rhd n \mid \Gamma \vdash_U s_i : \sigma_i \qquad (1 \le i \le m) \\ Z \rhd n \mid x_1 : \sigma_1, \ldots, x_m : \sigma_m \vdash_U t = t' : \tau \end{array}}{Z \rhd n \mid \Gamma \vdash_U t[\overline{x_i := s_i}] = t'[\overline{x_i := s_i}] : \tau}$$

$$\text{(Fun)}\ \frac{\begin{array}{c} (S \rhd f : \langle k_1\rangle(\overrightarrow{\sigma_1})\tau_1, \ldots, \langle k_\ell\rangle(\overrightarrow{\sigma_\ell})\tau_\ell \to \tau) \in \Sigma^{\mathsf{Tm}} \qquad k_i = |\overrightarrow{\alpha_i}| \\ \theta : S \rightsquigarrow_n U \qquad \theta_i = \imath_i^{[\underline{S} \Rightarrow U]}(\theta) \quad (\imath_i : n \hookrightarrow n + k_i) \\ X \rhd n + k_i \mid \Gamma, \overrightarrow{x_i : \sigma_i\theta_i} \vdash_U s_i = t_i : \tau_i\theta_i \quad (1 \le i \le \ell) \end{array}}{Z \rhd n \mid \Gamma \vdash_U f_\theta(\overrightarrow{\alpha_1}.\overrightarrow{x_1}.s_1, \ldots, \overrightarrow{\alpha_\ell}.\overrightarrow{x_\ell}.\, s_\ell) = f_\theta(\overrightarrow{\alpha_l}.\overrightarrow{x_1}.t_1, \ldots, \overrightarrow{\alpha_\ell}.\overrightarrow{x_\ell}.\, t_\ell) : \tau\theta}$$

$$\text{(MSub)}\ \frac{\begin{array}{c} \vartheta : Z \rightsquigarrow_{(k|\Delta)} \mathcal{N}_U \underline{Z'} \\ Z \rhd n \mid \Gamma \vdash_U s = t : \tau \end{array}}{Z' \rhd k + n \mid \Delta, \Gamma \vdash_U \vartheta^\sharp(s) = \vartheta^\sharp(t) : \tau}$$

$$\text{(MVar)}\ \frac{\begin{array}{c} \big(\mathrm{z} : \langle n + k\rangle(\tau_1, \ldots, \tau_\ell)\tau\,\big) \in Z \qquad \overrightarrow{\sigma} = \sigma_1, \ldots, \sigma_k \in U(n) \\ Z \rhd n \mid \Gamma \vdash_U s_i = t_i : \tau_i\{\overrightarrow{\sigma}\} \qquad (1 \le i \le \ell) \end{array}}{Z \rhd n \mid \Gamma \vdash_U \mathrm{z}\lfloor\overrightarrow{\sigma}\rfloor[s_1, \ldots, s_\ell] = \mathrm{z}\lfloor\overrightarrow{\sigma}\rfloor[t_1, \ldots, t_\ell] : \tau\{\overrightarrow{\sigma}\}}$$

$$\text{(Act)}\frac{\begin{array}{c} \rho : m \to n \qquad \pi : \rho(\Gamma) \to \Delta \\ Z \rhd m \mid \Gamma \vdash_U s = t : \tau \end{array}}{Z \rhd n \mid \Delta \vdash_U \pi\rho(s) = \pi\rho(t) : \rho(\tau)} \qquad \text{(TSub)}\ \frac{Z \rhd n + 1 \mid \Gamma \vdash_U s = t : \tau \qquad \sigma \in U(n)}{Z \rhd n \mid \Gamma\{\sigma\} \vdash_U s\{\sigma\} = t\{\sigma\} : \tau\{\sigma\}}$$

Fig. 2. Polymorphic Equational Logic (PEL)

Vernacular notation

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

Formal notation in PEL

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

Fig. 3. Sample axioms for System F

Vernacular notation

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

Formal notation in PEL

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

Fig. 4. Sample axioms for the existential λ-calculus $\lambda^{\exists}$

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

Fig. 5. Sample axioms for global state

Theorem VI.3 *Polymorphic equational logic is sound and complete; i.e., for all type universes U, an equational judgment $(Z \rhd n \mid \Gamma \vdash_U s = t : \tau)$ is derivable from a set of axioms E in PEL iff it is satisfied by all the algebraic models of E.*

VII. Examples

Example VII.1 Continuing with the examples of § II, sample axioms of System F, the existential λ-calculus, and global state are respectively presented as PEL equations in Figs. 3, 4, and 5.

Example VII.2 (**CPS translation [11]**) Fujita [11] gave a CPS translation from System F to the existential λ-calculus $\lambda_\exists$. The translation consists of a type translation $(-)^\bullet$ from System F types to $\lambda_\exists$-types, together with a CPS translation $[\![-]\!]$ from System F terms to $\lambda_\exists$-terms (e.g. $\big(\forall(\alpha.\tau)\big)^\bullet = \neg\exists(\alpha.\neg\tau^\bullet)$ and $[\![\Lambda\alpha.M]\!] = \lambda a.\, a\,(\lambda k.\mathsf{unpack}\ k\ \mathsf{as}\ \langle\alpha, c\rangle\ \mathsf{in}\ [\![M]\!]c)$). The CPS translation is sound and complete:

$$\Gamma \vdash_{\mathrm{F}} s = t : \tau \iff \neg\neg\Gamma^\bullet \vdash_{\lambda\exists} [\![s]\!] = [\![t]\!] : \neg\neg\tau^\bullet \ . \quad (8)$$

Interestingly, this pair of translations is an example of our notion of polymorphic translation. Indeed, let $\mathbb{T}_{\mathrm{F}}$ (resp. $\mathbb{T}_\exists$) be the initial type universe for $\Sigma_{\mathrm{F}}^{\mathsf{Ty}}$ (resp. $\Sigma_\exists^{\mathsf{Ty}}$) and let $\Lambda_{\mathrm{F}} = \mathcal{N}_{\mathbb{T}_{\mathrm{F}}}^{\mathrm{F}} 0$ (resp. $\Lambda_\exists = \mathcal{N}_{\mathbb{T}_\exists}^{\lambda_\exists} 0$) be the initial algebraic model for System F (resp. $\lambda_\exists$). The definition of $(-)^\bullet$ determines an $\Sigma_{\mathrm{F}}^{\mathsf{Ty}}$-algebra structure on $\mathbb{T}_\exists$, and the definition of $[\![-]\!]$ determines a $\Sigma_{\mathrm{F}}^{\mathsf{Tm}}$-algebra structure on $\Lambda_\exists^{\neg\neg} = \Lambda_\exists\big(\mathsf{G}\neg\neg(-)\big)$ for $\neg\neg$ the double negation endomap on $\mathbb{T}_\exists$. This yields a Σ_{F}-polymorphic structure $(\mathbb{T}_\exists, \Lambda_\exists^{\neg\neg})$ which, by (8), is an algebraic model of System F, and we have a Σ_{F}-polymorphic translation $\big((-)^\bullet, [\![-]\!]\big) : (\mathbb{T}_{\mathrm{F}}, \Lambda_{\mathrm{F}}) \to (\mathbb{T}_\exists, \Lambda_\exists^{\neg\neg})$.

Example VII.3 (**Categorical model of System F [30]**) Recall that a *PL-category* consists of a cartesian category $\mathbb{C}$ with objects given by finite powers of a distinguished object Ω, a $\mathbb{C}$-indexed cartesian closed category $\mathbb{C}(-,\Omega) : \mathbb{C}^{\mathrm{op}} \to \mathbf{CCC}$, for $\mathbf{CCC}$ the large category of cartesian closed categories, and right adjoints to $\mathbb{C}(\pi_1, \Omega) : \mathbb{C}(-,\Omega) \to \mathbb{C}(-\times\Omega, \Omega)$ satisfying the Beck-Chevalley condition.

From a PL-category, one defines a polymorphic structure $(\mathrm{U}, \mathrm{PL})$ for System F by setting

$$\mathrm{U} = \big|\mathbb{C}(\Omega^{(-)}, \Omega)\big| \in \mathbf{Set}^{\mathbb{F}} \ ,$$
$$\mathrm{PL}(\, n \mid \Gamma \vdash \tau\,) = \mathbb{C}(\Omega^n, \Omega)\big(\textstyle\prod_{1\le i\le|\Gamma|}\Gamma(i), \tau\big) \ .$$

Seely's categorical interpretation of the types and terms of System F determines $\Sigma_{\mathrm{F}}^{\mathsf{Ty}}$ and $\Sigma_{\mathrm{F}}^{\mathsf{Tm}}$ algebra structures on U and PL. For example, for $(\forall : (*)* \to *) \in \Sigma_{\mathrm{F}}^{\mathsf{Ty}}$, the corresponding algebraic operation $\forall^{\mathrm{U}} : \delta\mathrm{U} \to \mathrm{U}$ is defined by the right adjoint to $\mathbb{C}(\pi_1, \Omega) : \mathbb{C}(\Omega^{n+1}, \Omega) \to \mathbb{C}(\Omega^n, \Omega)$. The multiplication structures of both monoids U and PL are given by composition. The type-in-term substitution for $\sigma \in \mathrm{U}(n)$ arises as $\mathbb{C}\big(\langle \mathrm{id}_{\Omega^n}, \sigma\rangle, \Omega\big) : \mathrm{PL}(n+1 \mid \Gamma \vdash \tau) \to \mathrm{PL}(n \mid \Gamma\{\sigma\} \vdash \tau\{\sigma\})$. Altogether, this yields an algebraic model of System F.

Acknowledgments

Makoto Hamana is supported in part by a grant from Gunma University Foundation for Science and Technology and the JSPS Grant-in-Aid for Scientific Research (25540002) "A Semantics of Core Haskell".

References

[1] P. Aczel. A general Church-Rosser theorem. Technical report, University of Manchester, 1978.
[2] N. Benton, C.-K. Hur, A. Kennedy, and C. McBride. Strongly typed term representations in Coq. *J. Autom. Reasoning*, 49(2):141–159, 2012.
[3] N. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagat. Math.*, 34:381–391, 1972.
[4] M. Fiore. Mathematical models of computational and combinatorial structures. In *FoSSaCS'05*, LNCS 3441, pages 25–46, 2005.
[5] M. Fiore. Second-order and dependently-sorted abstract syntax. In *LICS'08*, pages 57–68, 2008.
[6] M. Fiore. Discrete generalised polynomial functors. In *ICALP'12*, pages 214–226, 2012.
[7] M. Fiore and C.-K. Hur. On the construction of free algebras for equational systems. *Theor. Comput. Sci.*, 410:1704–1729, 2008.
[8] M. Fiore and C.-K. Hur. Second-order equational logic. In *CSL'10*, LNCS 6247, pages 320–335, 2010.
[9] M. Fiore and O. Mahmoud. Second-order algebraic theories. In *MFCS'10*, LNCS 6281, pages 368–380, 2010.
[10] M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *LICS'99*, pages 193–202, 1999.
[11] K. Fujita. Galois embedding from polymorphic types into existential types. In *TLCA'05*, pages 194–208, 2005.
[12] N. Gambino and M. Hyland. Wellfounded trees and dependent polynomial functors. In *TYPES'03*, pages 210–225, 2003.
[13] J.-Y. Girard. *Interprétation Fonctionnelle et Élimination des Coupures de l'Arithmétique d'Ordre Supérieur*. Thèse de doctorat d'état, 1972.
[14] J.-Y. Girard. The System F of variable types, fifteen years later. *Theor. Comput. Sci.*, 45(2):159–192, 1986.
[15] A. Grothendieck. Catégories fibrées et descente (exposé VI). In *SGA1*, LNM 244, pages 145–194. Springer-Verlag, 1970.
[16] M. Hamana. Free Σ-monoids: A higher-order syntax with metavariables. In *APLAS'04*, LNCS 3301, pages 348–363, 2004.
[17] M. Hamana. Universal algebra for termination of higher-order rewriting. In *RTA'05*, LNCS 3467, pages 135–149, 2005.
[18] M. Hamana. An initial algebra approach to term rewriting systems with variable binders. *Higher-Order and Symb. Comput.*, 19:231–262, 2006.
[19] M. Hamana. Higher-order semantic labelling for inductive datatype systems. In *PPDP'07*, pages 97–108, 2007.
[20] M. Hamana. Polymorphic abstract syntax via Grothendieck construction. In *FoSSaCS'11*, LNCS3467, pages 381–395, 2011.
[21] J.W. Klop. *Combinatory Reduction Systems*. PhD thesis, CWI, Amsterdam, 1980. volume 127 of Mathematical Centre Tracts.
[22] F. W. Lawvere. Adjointness in foundations. *Dialectica*, 23:281–296, 1969.
[23] R. E. Møgelberg. From parametric polymorphism to models of polymorphic FPC. *Math. Struct. in Compt. Sci.*, 19(4):639–686, 2009.
[24] M. Miculan and I. Scagnetto. A framework for typed HOAS and semantics. In *PPDP'03*, pages 184–194. ACM Press, 2003.
[25] R. Milner. A theory of type polymorphism in programming. *J. Comput. Syst. Sci.*, 17(3):348–375, 1978.
[26] I. Moerdijk and E. Palmgren. Wellfounded trees in categories. *Ann. Pure Appl. Logic*, 104:189–218, 2000.
[27] B. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *J. ACM*, 47(3):531–584, 2000.
[28] G. Plotkin and J. Power. Notions of computation determine monads. In *FoSSaCS'02*, pages 342–356, 2002.
[29] J. Reynolds. Types, abstraction and parametric polymorphism. *Inform. Process.*, 83:513–523, 1983.
[30] R. A. G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *J. Symb. Log.*, 52(4):969–989, 1987.
[31] I. Stark. Free-algebra models for the π-calculus. *Theor. Comput. Sci.*, 390(2–3):248–270, 2008.
[32] C. Strachey. Fundamental concepts in programming languages. *Higher-Order and Symbolic Computation*, 13:11–49, 2000.