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 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 polymorphism is one of the most remarkable inventions in programming languages. The concept was introduced by Strachey in 1967 (see [28]) who classified the phenomena into ad-hoc and parametric polymorphism. The theory of the latter started with the polymorphic λ-calculus of Girard [13] and Reynolds [26], and led to Milner's striking application to functional programming [22]. Since then, the theory of polymorphism has deepened and its applicability spread broadly. Nowadays, it is not only supported in functional languages (e.g. Haskell), but is an integral part of a variety of calculi, such as Coq, the π-calculus [24], and XML [18]. It has also been incorporated into object-oriented languages (e.g. C++, Scala) 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 rooted in 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 λ-calculi. Rather, it is semantically driven, capturing varieties of polymorphic systems that include extended λ-calculi as particular examples. The necessary background for our work follows.

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

$$(\nu)\ \frac{1 \leq i \leq n}{1,\ldots,n \vdash \nu_i} \qquad (@)\ \frac{1,\ldots,n \vdash t \quad 1,\ldots,n \vdash s}{1,\ldots,n \vdash t@s}$$

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

Its abstract syntax is generated by three term constructors: the variable ν, the application $@$, and the abstraction λ. Here, ν is a nullary operation parameterised by the context, while $@$ is a binary operation; λ, however, is not merely a unary operator, as it binds a variable (and thereby decreases the context). To model the general phenomenon of variable binding, Fiore, Plotkin and Turi [10] took the presheaf category $\mathbf{Set}^{\mathbb{F}}$ as universe of discourse. Here, $\mathbb{F}$ is the category with objects $\{1,\ldots,n\}$ $(n \in \mathbb{N})$, for which we henceforth abuse notation and simply write $n \in \mathbb{F}$, and all functions between them. 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 operators) is characterised as the initial algebra of a prescribed signature endofunctor on $\mathbf{Set}^{\mathbb{F}}$. For example, the signature endofunctor Σ_λ on $\mathbf{Set}^{\mathbb{F}}$ for λ-terms is given by $\Sigma_\lambda(X) = \mathrm{V} + X \times X + \delta X$ 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 $\delta X(n) = X(n+1)$.

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 judgments $n \vdash t$ are modelled as elements

of $\coprod_{n\in\mathbb{F}} \Lambda(n)$, while the presheaf action models the renaming of free variables.

Fiore, Plotkin and Turi [10] also considered the modelling of single-variable and simultaneous capture-avoiding substitution. The relevant mathematical structure for the latter was elucidated as that of a *Σ-monoid* (A, α, ν, μ), consisting of a compatible pair of a Σ-algebra (A, α) and a monoid (A, ν, μ) with respect to the substitution monoidal structure $(\mathrm{V}, \bullet)$ on $\mathbf{Set}^{\mathbb{F}}$. The unit ν models variables; the multiplication μ substitution.

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. Indeed, because of α-equivalence, the operation of substituting a term for an object variable is not a simple textual substitution, e.g $(\lambda x.\, y\, M)[y := x\, y] = \lambda z.\, x\, y\, M[x := z][y := x\, y]$ for fresh z; while, on the contrary, the substitution of a term for a metavariable essentially is, e.g.

$$(\lambda x.\, y\, M)[M \mapsto x\, y] = \lambda x.\, y\, (x\, y) \ . \tag{1}$$

Free Σ-monoids. A mathematical theory for both object variables and metavariables should thus take the above distinctions into account. This problem was explored in Hamana [16] and in Fiore [5, Part I]. The key is to use *free Σ-monoids*.

The idea is to regard presheaves $X \in \mathbf{Set}^{\mathbb{F}}$ as metavariable declarations, with elements of $X(k)$ corresponding to metavariables of arity k, and consider the Σ-monoid $\mathcal{M}X$ freely generated by X. This can be inductively constructed as an initial $\big(\mathrm{V} + \Sigma(-) + X \bullet (-)\big)$-algebra, with structure maps $\mathrm{V} \to \mathcal{M}X$, $\Sigma(\mathcal{M}X) \to \mathcal{M}X$, and $X \bullet \mathcal{M}X \to X$ respectively modelling variables, operators, and parameterised metavariables. The latter under the rule

$$\frac{1, \ldots, n \vdash t_i \ \ (1 \le i \le k)}{1, \ldots, n \vdash \mathrm{M}[t_1, \ldots, t_k]} \ \mathrm{M} \in X(k)$$

The arity of a metavariable denotes the number of term parameters that it is to take for subsequent substitution in terms instantiating the metavariable.

The universal property of $\mathcal{M}X$ internalises [5] as a map

$$\mathcal{M}X \times [X \Rightarrow A] \to A : (t, \theta) \mapsto t\theta \quad \text{in } \mathbf{Set}^{\mathbb{F}}$$

for every Σ-monoid A. It is here that the notion of *metavariable substitution* appears. Syntactically, one understands θ as an assignment and $t\theta$ as the induced meta-substitution operation on t. For instance, example (1) above is formally recast as

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

where M has arity 1 and θ corresponds to the assignment $\big[\mathrm{M}[\ ] \mapsto [\ ] @ y\big]$.

Second-order algebra and equational logic. The syntactic theory of abstract syntax with variable binding and metavariables was introduced by Aczel [1]. This formal language allowed him to consider equational presentations and rewriting rules for calculi with variable binding. For instance, in this setting, the β and η axioms of the λ-calculus are each specified as a single equation:

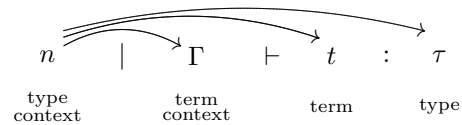
$$\begin{array}{ll} (\beta) & \mathrm{M} : 1, \mathrm{N} : 0 \vdash \lambda(x.\, \mathrm{M}[x]) @ \mathrm{N} = \mathrm{M}[\mathrm{N}] \ , \\ (\eta) & \mathrm{M} : 0 \vdash \lambda(x. \mathrm{M} @ x) = \mathrm{M} \ . \end{array} \tag{2}$$

In view of the above developments, Fiore et al. [8,9] considered algebraic theories for such *second-order* equational presentations. As in (first-order) universal algebra, one may work with these model theoretically as abstract algebraic structures (viz. Σ-monoids satisfying the axioms) [9] or logically by means of a logical framework referred to as second-order equational logic [8].

An important point here is that the generality of the approach makes it also applicable to the formalisation of type structure with binding. For instance, signatures Σ with a binding operator μ, say intended to model recursive types, give rise to free Σ-monoids modelling abstract syntax with μ-types, type-variable binding, and type metavariables. Equational presentations on these can also be considered; e.g. for modelling equirecursive types: $\mu(x.\mathrm{T}[x]) = \mathrm{T}\big[\mu(x.\mathrm{T}[x])\big]$.

Polymorphic abstract syntax. The above theory is limited to sorted (e.g. untyped and simply-typed) languages. The passage from the mono-sorted universe of discourse $\mathbf{Set}^{\mathbb{F}}$ to an S-sorted one consists in considering the presheaf category $\mathbf{Set}^{(\mathbb{F}\downarrow S)\times S}$ for $(\mathbb{F}\downarrow S) \times S$ the category of S-sorted contexts paired with sorts [21,4].

Hamana [17] tackled the algebraic modelling of polymorphically-typed abstract syntax. A crucial departure from the multi-sorted case 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:



(3)

The arrows here indicate dependency, as type variables may appear in any of the other parts of the judgment. To model such dependent context and type structure with respect to a universe of types in context $U \in \mathbf{Set}^{\mathbb{F}}$, Hamana [17] identified the Grothendieck construction [15] (for which we use the notation $\oint$) as key, introducing the category

$$\mathsf{G}U = \oint^{n\in\mathbb{F}} (\mathbb{F}\downarrow Un) \times Un$$

with objects given by structures of the form:

$$\big(\ n \in \mathbb{F}\ ,\ \big\langle\ |\Gamma| \in \mathbb{F}\ ,\ \Gamma \in (Un)^{|\Gamma|}\ \big\rangle\ ,\ \tau \in Un\ \big) \ .$$

The universe of discourse $\mathbf{Set}^{GU}$ was then shown to be appropriate to model polymorphically-typed object-level syntax [17]. Fiore [6] further isolated the notion of generalised polynomial functor as a suitable mathematical theory for this purpose.

This paper. We develop polymorphic algebraic theories founded on these earlier works. We incorporate the notions of metavariables and Σ-monoids into the algebraic polymorphic setting. 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 vernacular rule for abstraction in the typed λ-calculus:

$$\frac{[x:\sigma]\, M:\tau}{\lambda x^{\sigma}.\, M:\sigma\Rightarrow\tau} \tag{4}$$

Is this a *single* operator? One usally thinks so. But this is only the case when regarded as a *schema*, with σ and τ being type metavariables that, by instantiation, yield a family of (object-level) operators indexed by all pairs of (object) types. Analogously, consider the typed versions of the β and η axioms (2):

$$\begin{array}{ll}(\beta) & \mathrm{M}:[\sigma]\tau,\mathrm{N}:\sigma\vdash\lambda_{\sigma,\tau}(x.\,\mathrm{M}[x])@\mathrm{N}=\mathrm{M}[\mathrm{N}]:\tau\ ,\\ (\eta) & \mathrm{M}:\sigma\Rightarrow\tau\vdash\lambda_{\sigma,\tau}(x.\,\mathrm{M}@x)=\mathrm{M}:\sigma\Rightarrow\tau\ .\end{array} \tag{5}$$

Are these a pair of *single* axioms? Again, this is only the case when they are regarded as *schema* with σ and τ being type metavariables yielding a family of (object-level) axioms by instantiation.

Although the distinction between meta and object levels usually receives little attention when developing a theory in the mathematical vernacular, this viewpoint is seriously needed when developing a formal meta-theory or a mechanised formalisation of a theory. In such case, one must formalise all ingredients, including the meta-level aspects.

We precisely formulate the distinction between meta and object levels. For instance, in our polymorphic algebraic theory, the λ-abstraction rule (4) is formalised as a single operator specified as

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

where S and T are metavariables for types, and where the symbol $\triangleright$ separates the type metavariable context from the source and target arity information (resp. $(\mathrm{S})\mathrm{T}$ and $\mathrm{S}\Rightarrow\mathrm{T}$) of the operator (here termed abs) as specified by meta-types. In a similar vein, our formalism permits single specifications of the β and η axioms (5).

Conceptually, our analysis leads to the viewpoint that the universe of types in context $U\in\mathbf{Set}^{\mathbb{F}}$ as discussed above should be a Σ-monoid for a type signature Σ. For instance, the initial Σ-monoid $\mathcal{M}0$ corresponds to the universe of object types; the free Σ-monoid $\mathcal{M}X$ on a presheaf of type metavariables X corresponds to the universe of meta-types with metavariables in X; and the free Σ-monoid $\mathcal{M}^{E}X$ satisfying a second-order equational presentation E on a presheaf of type metavariables X corresponds to the universe of meta-types with metavariables in X modulo the axiomatisation E. 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*.

The notion of algebraic structure for polymorphism that arises consists thus of a Σ^{Ty}-monoid $U\in\mathbf{Set}^{\mathbb{F}}$ for a type signature Σ^{Ty} together with a Σ^{Tm}-monoid $A\in\mathbf{Set}^{GU}$ for a term signature Σ^{Tm} with U acting on A by means of a *type-in-term substitution* structure. Free algebras provide polymorphically-typed abstract syntax, from which equational presentations arise as sets of equations. As customary, algebraic models for equational presentations are then defined as algebras that satisfy the equations. The logical counterpart of this model theory is provided by *Polymorphic Equational Logic (PEL)*, a sound and complete logical framework for equational reasoning about polymorphic algebraic theories.

Further work. Various directions for further work are possible. A promising one is to apply the mathematical theory of the paper to mechanised formalisation. In the light of the correspondence between dependent polynomial functors and inductively defined dependent types [23,12], our algebraic theory provides a methodology for formalising the syntax and semantics of polymorphic systems in proof assistants supporting dependent and inductive types. In fact, the strongly-typed representation of System F syntax of Benton et al. [2] in Coq can be thus obtained.

The type structure of the present theory will be extended to encompass polymorphic kinds. Preliminary results [17] indicate that this direction will be a natural extension of the current framework.

2. 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 2.1 (**System F [13,26]**) The polymorphic signature $\Sigma_{\mathrm{F}}=(\Sigma_{\mathrm{F}}^{\mathsf{Ty}},\Sigma_{\mathrm{F}}^{\mathsf{Tm}})$ for System F consists of: a type signature $\Sigma_{\mathrm{F}}^{\mathsf{Ty}}=\{\ \mathsf{b}:*,\quad\Rightarrow:*,*\to*,\quad\forall:(*)*\to*\ \}$ together with a term signature $\Sigma_{\mathrm{F}}^{\mathsf{Tm}}$ given as follows

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

The type signature specifies three type constructors. Their associated arities respectively express that b is a type, that $\Rightarrow$ is a constructor building a type from two given types, and that $\forall$ is a constructor building a type as the result of binding a type variable in a type.

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 S and T, both of arity $*$ and thereby to be understood as representing types. The the source arity $(\mathrm{S})\mathrm{T}$ of abs represents the premise of the term-abstraction rule, expressing that it consists of a term of type T in a context extended with a fresh term variable of type S. Note that the informal metavariables σ and τ of the vernacular rule are respectively formalised by means of the formal metavariables S and T. The target arity $\mathrm{S} \Rightarrow \mathrm{T}$ of abs represents the type of the term in the conclusion of the rule.

The arity of the tabs term constructor is parameterised by one metavariable T of arity $[*]*$ representing an open type. The source arity $\alpha.\mathrm{T}[\alpha]$ 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.

The formal treatment follows.

2.1. Type signatures and universes

Definition 2.2 A *type signature* Σ^{Ty} is a set of type constructors with arities specified as $c : (*^{n_1})*, \cdots, (*^{n_l})* \to *$ 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.

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

Notation 2.3 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 functor $|-| : \mathbf{Set}^{\mathbb{C}} \to \mathbf{Set}^{|\mathbb{C}|}$ is monadic and its left adjoint is denoted $\underline{(-)}$. It is explicitly given by $\underline{S}(c) = \coprod_{z \in \mathbb{C}} S(z) \times \mathbb{C}(z, c)$.

Definition 2.4 A *type metavariable* S of arity $n \in \mathbb{N}$ is declared as $\mathrm{S} : [*^n]*$, with the prefix omitted when $n = 0$. Sets of type metavariable declarations S correspond to $\mathbb{N}$-indexed sets $\{S(k)\}_{k\in\mathbb{N}}$ where the $S(k)$ are the sets of metavariables in S of arity k, and can be freely considered as a presheaf $\underline{S} \in \mathbf{Set}^{\mathbb{F}}$. A pair $(\mathrm{S}, \rho) \in \underline{S}(n)$ with $\mathrm{S} \in S(k)$ and $\rho : k \to n$ in $\mathbb{F}$ is displayed as $\mathrm{S}[\rho_1, \ldots, \rho_k]$, and simply as S when $k = 0$.

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. It is thus natural to use the notation $S \triangleright n \vdash \tau$ for $\tau \in \mathcal{M}\underline{S}(n)$. 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 2.1, the close type $\forall(\alpha.\mathrm{T}[\alpha])$, where T is a type metavariable of arity 1, stands for $\forall(1.\mathrm{T}[1])$.

Definition 2.5 A *type universe* for a type signature Σ^{Ty} is defined to be Σ^{Ty}-monoid.

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 7.3).

We consider next the denotation of meta-types by means of a meta-substitution operation [16,5].

Definition 2.6 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

$$S \triangleright k \vdash \tau \ , \ \theta : S \rightsquigarrow_{n+k} U \ \mapsto \ \tau\theta \in U(n+k) \ .$$

2.2. Polymorphic signatures

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

Definition 2.7 A *polymorphic signature* $\Sigma = (\Sigma^{\mathsf{Ty}}, \Sigma^{\mathsf{Tm}})$ consists of a type signature Σ^{Ty} together with a term signature Σ^{Tm} given by a set of operators with arities of the form

$$S \triangleright f : k_1.(\overrightarrow{\sigma_1})\tau_1, \ldots, k_\ell.(\overrightarrow{\sigma_l})\tau_\ell \to \tau \quad (\ell \in \mathbb{N}) \qquad (6)$$

where 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 $1 \leq i \leq \ell$, 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.

An important point to note above is that the source and target arities of operators are written in the language of syntactic type universes rather than in an informal meta-language. In particular, operator declarations are meant to be instantiated by means of meta-substitutions (see Fig. 1).

Example 2.8 (**Polymorphic FPC [20]**) The signature Σ^{Ty} for types is $\Sigma_{\mathrm{F}}^{\mathsf{Ty}}$ extended with $+, \times : *, * \to *$ and $\mu : (*)* \to *$. An excerpt of the term signature follows:

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

Example 2.9 (**Existential λ-calculus [11]**) The type signature is given by $\bot : *$, $\neg : * \to *$, $\wedge : *, * \to *$, and

$\exists : (*)* \to *$. As for the terms, we only consider the following two key rules:

$$\frac{\Xi \mid \Gamma \vdash s : \sigma\{\alpha := \tau\}}{\Xi \mid \Gamma \vdash \langle \tau, s\rangle : \exists(\alpha.\sigma)}$$

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

The rule for unpack requires the side condition that in its second premise the variable α is fresh for all the types in the context Γ and for τ. The term signature corresponding to these two rules is thus:

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

Note that the second argument of the operator 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 2.10 (**Global state**) The signature for a basic algebraic theory of global state [25] 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 has two operators:

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

providing operations that are parameterised by types. This is unlike the original treatment [25], 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.

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 operators (parameterised by type metavariables) between them.

3. Polymorphic Signature Endofunctors

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 operators (§ 3.3). As customary, then, models arise as endofunctor algebras.

Two preliminary sections are respectively devoted to introducing the appropriate universe of discourse (§ 3.1) and the general theory (§ 3.2) for the interpretation.

3.1. Contexts for polymorphism

Following [17], 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 $\oint \mathcal{F}$ or $\oint^{I \in \mathscr{F}} \mathcal{F}(I)$ 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 give sample uses of the Grothendieck construction for manufacturing categories of contexts.

Example 3.1 For a set T (of sorts), the category $\mathbb{F}\!\downarrow\! T$ (of T-sorted contexts) is the opposite of $\oint^{n \in \mathbb{F}^{\mathrm{op}}} T^n$. 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 $\Gamma = \Gamma' \circ \rho : |\Gamma| \to T$.

Example 3.2 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 $\mathsf{F}T = \oint^{n \in \mathbb{F}} \mathbb{F}\!\downarrow\!(Tn)$ (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 $X \in \mathbf{Set}^{\mathbb{F}}$, we define $\mathsf{H}(T, X) = \oint^{n \in \mathbb{F}} \mathbb{F} \downarrow (Tn) \times Xn$. This category has objects $(n \mid \Gamma \vdash x)$ with $(n \,|\, \Gamma) \in \mathsf{F}T$ and $x \in Xn$, and morphisms $(n \mid \Gamma \vdash x) \to (n' \mid \Gamma' \vdash x')$ given by maps $(\rho, \pi) : (n|\Gamma) \to (n'|\Gamma')$ in $\mathsf{F}T$ such that $X\rho x = x'$. An ubiquitous particular case is the category $\mathsf{G}T = \mathsf{H}(T, T)$.

3.2. 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 [23,12]. This section reviews the basics, deferring details to [6].

We need 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 [15,19]. For instance, the adjunction $\underline{(-)} \dashv |-| : \mathbf{Set}^{\mathbb{C}} \to \mathbf{Set}^{|\mathbb{C}|}$ introduced in Notation 2.3 amounts to $i_! \dashv i^*$ for i the inclusion $|\mathbb{C}| \to \mathbb{C}$.

A $\mathbf{Cat}$-*polynomial* P is a diagram $\mathbb{A} \xleftarrow{s} \mathbb{I} \xrightarrow{a} \mathbb{J} \xrightarrow{t} \mathbb{B}$ in $\mathbf{Cat}$. Its induced *generalised polynomial functor* F_P is the composite $t_!\ a_*\ s^* : \mathbf{Set}^{\mathbb{A}} \to \mathbf{Set}^{\mathbb{B}}$. 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 $[\mathrm{Id}_{\mathbb{C}}]_{\ell \in L}$.

Discrete generalised polynomial endofunctors are useful because by means of their algebras one can specify structure on presheaves with complex source and target arities. To see this, note that an F_P-algebra for the generalised polynomial endofunctor induced by a discrete polynomial P as

$$\mathbb{C} \xleftarrow{[s_i]_{i \in I}} \coprod_{i \in I} L_i \cdot \mathbb{C}_i \xrightarrow{\coprod_{i \in I} \nabla_{L_i}} \coprod_{i \in I} \mathbb{C}_i \xrightarrow{[t_i]_{i \in I}} \mathbb{C}$$

consists of a presheaf $A \in \mathbf{Set}^{\mathbb{C}}$ together with an I-indexed family of structure maps

$$\textstyle\prod_{l\in L_i} A\big(s_i(l\cdot -)\big) \to A\big(t_i(-)\big) \text{ in } \mathbf{Set}^{\mathbb{C}_i} \quad . \tag{7}$$

Note that the first and last components of a polynomial respectively specify source and target re-indexing. As we will see, this provides a very expressive and flexible formalism. Indeed, since furthermore discrete generalised polynomial functors admit inductive constructions of free algebras [6, Prop. 5.1], in the following two sections we will use them to model polymorphic signatures (§ 3.3) and type-in-term substitution (§ 4).

3.3. Polymorphic signature endofunctors

We define the signature endofunctor corresponding to a polymorphic signature Σ. This is done parametrically on a type universe U for Σ^{Ty} on the universe of discourse given by the presheaf category $\mathbf{Set}^{\mathsf{G}U}$ (recall Example 3.2).

To explain the approach, we start with analysing concrete cases in System F (Example 2.1). For the term abstraction operator abs, the corresponding operation of an algebra $A \in \mathbf{Set}^{\mathsf{G}U}$ should be given by a natural map

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

Operators may also extend contexts for type variables in operations, as in the case of the type abstraction operator tabs for which the corresponding operation of an algebra $A \in \mathbf{Set}^{\mathsf{G}U}$ should be given by a natural map

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

To obtain these kind of algebraic operations from the arity specification of operators, 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 to use of the Grothendieck construction $\mathsf{H}\big(U, [\underline{S} \Rightarrow U]\big)$ of Example 3.2, for S a set of metavariable declarations.

Definition 3.3 Let Σ be a polymorphic signature and U a type universe for Σ^{Ty}. To every operator f as in (6) 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 $s_f\big(\, i \cdot (\, n \mid \Gamma \vdash \theta\,)\,\big) = (n + k_i \mid \big(\Gamma_i + \langle \overrightarrow{\sigma_i} \theta_i \rangle\big) \vdash \tau_i\theta_i)$ 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 endofunctor* 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

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

where the coproduct ranges over operators f in Σ^{Tm} as in (6) and assignments $\theta : S \rightsquigarrow_n U$ such that $\tau\theta = u \in Un$.

4. Type-in-term Substitution

The monoid multiplication of a type universe provides an operation that models simultaneous capture-avoiding type-in-type substitution. Also 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 $\tau \in U(n+l)$ and $\sigma_i \in U(n)$ with $1 \le i \le l$, we let $\tau\{\sigma_1, \ldots, \sigma_l\} \in U(n)$ be given by the multi-variable capture-avoiding substitution $\mu_n(\tau; \nu_n(1), \ldots, \nu_n(n), \sigma_1, \ldots, \sigma_l)$; pointwise extending the notation to $\Gamma\{\sigma_1, \ldots, \sigma_l\} \in \mathbb{F}{\downarrow}(Un)$ for $\Gamma \in \mathbb{F}{\downarrow}\big(U(n+l)\big)$. Our aim is to define algebraic structure on a presheaf $A \in \mathbf{Set}^{\mathsf{G}U}$ amounting to structure maps

$$\begin{array}{rccc} \varsigma^{\sigma}_{n} : & A(n+1 \mid \Gamma \vdash \tau) & \longrightarrow & A\big(n \mid \Gamma\{\sigma\} \vdash \tau\{\sigma\}\big) \\ & a & \mapsto & (a)\{\sigma\} \end{array} \tag{8}$$

with $(n \mid \Gamma \vdash \tau) \in \mathsf{G}U$ and $\sigma \in Un$, 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

$$\mathsf{G}U \xleftarrow{s} \mathsf{H}(\delta U, \delta U \times U) \xrightarrow{t} \mathsf{G}U \tag{9}$$

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 endofunctor on $\mathbf{Set}^{\mathsf{G}U}$ induced by (9), noting that according to (7) an algebra-structure map $\uparrow A \to A$ exactly specialises to (8).

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, swapping, and contraction maps $\mathsf{up} : 0 \to 1$, $\mathsf{sw} : 2 \to 2$, and $\mathsf{con} : 2 \to 1$.

Definition 4.1 Let (U, ν) be a V-pointed object in $\mathbf{Set}^{\mathbb{F}}$. A *type-in-term substitution* for $A \in \mathbf{Set}^{\mathsf{G}U}$ is an algebra $\uparrow A \to A$ in $\mathbf{Set}^{\mathsf{G}U}$ subject to the following axioms:

- $a \in A(n \mid \Gamma \vdash \tau),\ \sigma \in U(n) \vdash (\mathsf{up}^{A}_{n}\, a)\{\sigma\} = a$
- $a \in A(n+2 \mid \Gamma \vdash \tau),\ \sigma \in U(n+1),\ \sigma' \in U(n)$ $\vdash \big((a)\{\sigma\}\big)\{\sigma'\} = \big((\mathsf{sw}^{A}_{n}\, a)\{\mathsf{up}^{U}_{n}\, \sigma'\}\big)\{\sigma\{\sigma'\}\}$
- $a \in A(n+2 \mid \Gamma \vdash \tau) \vdash (a)\{\mathsf{new}_n\} = \mathsf{con}^{A}_{n}\, a$

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.

Example 4.2 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 ς^{V} :

$\uparrow \mathrm{V} \to \mathrm{V}$ given by maps that reinterpret a variable of type τ in context Γ as one of type $\tau\{\sigma\}$ in context $\Gamma\{\sigma\}$.

5. 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 [21,4,17], for $A, B \in \mathbf{Set}^{\mathsf{G}U}$, we define $(A \bullet B) \in \mathbf{Set}^{\mathsf{G}U}$ by the coend

$$(A \bullet B)(n \mid \Gamma \vdash \tau) = \int^{\Delta \in \mathbb{F}\downarrow Un} A(n \mid \Delta \vdash \tau) \times \textstyle\prod_{1 \le i \le |\Delta|} B\big(n \mid \Gamma \vdash \Delta(i)\big) \quad (10)$$

remarking that $(\mathrm{V}, \bullet)$ provides a monoidal structure on $\mathbf{Set}^{\mathsf{G}U}$ suitable for considering monoids with polymorphic algebraic structure.

Definition 5.1 A *polymorphic structure* for a polymorphic signature Σ and a type universe U for Σ^{Ty} consists of a Σ-monoid (A, φ, ν, μ) in $\mathbf{Set}^{\mathsf{G}U}$ and a type-in-term substitution $\varsigma : \uparrow A \to A$ that are compatible in the sense that the following diagrams commute

$$\begin{array}{ccc} \uparrow \Sigma A & \longrightarrow & \Sigma \uparrow A \xrightarrow{\Sigma\varsigma} \Sigma A \\ {\scriptstyle \uparrow\varphi}\downarrow & & \downarrow{\scriptstyle \varphi} \\ \uparrow A & \xrightarrow[\varsigma]{} & A \end{array} \qquad \begin{array}{ccc} \uparrow \mathrm{V} & \xrightarrow{\varsigma^{\mathrm{V}}} & \mathrm{V} \\ {\scriptstyle \uparrow\nu}\downarrow & & \downarrow{\scriptstyle \nu} \\ \uparrow A & \xrightarrow[\varsigma]{} & A \end{array} \qquad \begin{array}{ccc} \uparrow (A \bullet A) & \longrightarrow \uparrow A \bullet \uparrow A \xrightarrow{\varsigma\bullet\varsigma} & A \bullet A \\ {\scriptstyle \uparrow\mu}\downarrow & & \downarrow{\scriptstyle \mu} \\ \uparrow A & \xrightarrow[\varsigma]{} & A \end{array}$$

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

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

The category of polymorphic structures and translations for a polymorphic signature Σ and a type universe U for Σ^{Ty} is denoted $\mathbf{Poly}(\Sigma, U)$.

5.1. Free polymorphic structures

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

Theorem 5.2 *The forgetful functor* $\mathbf{Poly}(U, \Sigma) \to \mathbf{Set}^{\mathsf{G}U}$ *is monadic.*

We outline a construction of free 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

$$\mathrm{z}\{\overrightarrow{\sigma}\}[t_{\pi 1}, \ldots, t_{\pi|\Delta|}] \;=\; \mathrm{z}'\{\overrightarrow{\sigma}\}\big[t_1, \ldots, t_{|\Delta'|}\big] \qquad (11)$$

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 (10). The presheaf and polymorphic structures of $\mathcal{N}_U X$ are essentially given syntactically.

For a polymorphic structure $(A, \varphi, \nu, \mu, \varsigma)$, the universal extension of a morphism $\vartheta : X \to A$ in $\mathbf{Set}^{\mathsf{G}U}$ is the polymorphic translation $\vartheta^\sharp : \mathcal{N}_U X \to A$ given as follows:

- $\vartheta^\sharp(x) = \nu(x)$
- $\vartheta^\sharp\big(f_\theta(\ldots, \overrightarrow{\alpha_i}.\overrightarrow{x_i}.t_i, \ldots)\big) = \varphi^f(\ldots, \vartheta^\sharp(t_i), \ldots)$
- $\vartheta^\sharp\big(\mathrm{z}\{\sigma_1, \ldots, \sigma_l\}[t_1, \ldots, t_m]\big)$
  $= \mu\big((\vartheta\,\mathrm{z})\{\sigma_1, \ldots, \sigma_l\};\ \vartheta^\sharp(t_1), \ldots, \vartheta^\sharp(t_m)\big)$

where, extending the notation introduced in (8), for $a \in A(n + l \mid \Gamma \vdash \tau)$ and $\sigma_i \in U(n)$ with $1 \le i \le l$, we set $(a)\{\sigma_1, \ldots, \sigma_l\} = \big(\cdots\big((a)\{\imath_1(\sigma_1)\}\big)\cdots\big)\{\imath_l(\sigma_l)\}$ for $\imath_i$ the coproduct injection $n \hookrightarrow n + l - i$.

5.2. Polymorphic abstract syntax with metavariables

We now consider free polymorphic structures that correspond to *polymorphic abstract syntax with metavariables*. For this purpose, the notion of metavariable arity is directly suggested by the mathematical model as that of a context for polymorphism. This leads to the consideration of indexed sets $Z \in \mathbf{Set}^{|GU|}$ as metavariable declarations, where one regards $\mathrm{z} \in Z(n \mid \overrightarrow{\sigma} \vdash \tau)$ as the declaration $\mathrm{z} : \{n\}[\overrightarrow{\sigma}]\tau$, for the metavariable z of arity $\{n\}[\overrightarrow{\sigma}]\tau$, in Z. Such metavariables are to be instantiated by terms involving n fresh type variables and $|\overrightarrow{\sigma}|$ fresh variables.

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 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

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

We henceforth refer to elements of $\mathcal{N}_U \underline{Z}(n \mid \Gamma \vdash \tau)$ as *meta-terms*. For these we introduce an operation of meta-substitution (i.e. substitution for metavariables in meta-terms) as follows.

Definition 5.3 For $(k|\Delta) \in \mathsf{F}U$, 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

$$d(k|\Delta)(n \mid \Gamma \vdash \tau) = (\ k + n \ \mid\ \Delta, \Gamma \vdash \jmath(\tau)\ )$$

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

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

(Var)
$$\frac{(x : \tau) \in \Gamma}{x \in N_U Z(n \mid \Gamma \vdash \tau)}$$

(MVar)
$$\frac{\mathrm{z} \in Z(n + l \mid x_1 \colon \tau_1, \ldots, x_m \colon \tau_m \vdash \tau) \qquad \sigma_i \in U(n) \quad (1 \le i \le l) \qquad t_i \in N_U Z\big(n \mid \Gamma \vdash \tau_i\{\sigma_1, \ldots, \sigma_l\}\big) \quad (1 \le i \le m)}{\mathrm{z}\{\sigma_1, \ldots, \sigma_l\}[t_1, \ldots, t_m] \in N_U Z\big(n \mid \Gamma \vdash \tau\{\sigma_1, \ldots, \sigma_l\}\big)}$$

NB: 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 operators bind both type and term variables and we implicitly assume the technique of de Bruijn levels [3] for their representation.

Fig. 1. Rules for $N_U Z$

For every polymorphic structure A, the presheaf $\delta^{(k|\Delta)}A$ canonically inherits a polymorphic structure. Thus, every assignment $\vartheta : Z \rightsquigarrow_{(k|\Delta)} A$ yields an *interpretation function*

$$\begin{array}{rccl} \vartheta^\sharp : & \mathcal{N}_U \underline{Z}(n \,|\, \Gamma \vdash \tau) & \longrightarrow & A(\, k + n \mid \Delta, \Gamma \vdash \jmath(\tau)\,) \\ & t & \mapsto & t\vartheta \end{array}$$

that gives meaning to meta-terms according to metavariable interpretations.

5.3. The multiverse of polymorphic structures

We combine all polymorphic structures, varying over all type universes, for a polymorphic signature Σ into a single category $\mathbf{Poly}(\Sigma)$. This category abstractly arises as the (contravariant) Grothendieck construction applied to a functor $\mathbf{Poly}(\Sigma, -) : \Sigma^{\mathsf{Ty}}\text{-}\mathbf{Mon}^{\mathrm{op}} \to \mathbf{CAT} : U \mapsto \mathbf{Poly}(\Sigma, U)$. An explicit definition follows. We note that it relies on the fact that for every Σ^{Ty}-monoid homomorphism $\phi : U \to U'$, the functor $(\mathsf{G}\phi)^* : \mathbf{Set}^{\mathsf{G}U'} \to \mathbf{Set}^{\mathsf{G}U}$ maps polymorphic structures for (Σ, U') to polymorphic structures for (Σ, U).

Definition 5.5 The category $\mathbf{Poly}(\Sigma)$ for a polymorphic signature Σ has objects (U, A) consisting of a type universe U for Σ^{Ty} and a polymorphic structure A for (Σ, U). A morphism $(\varphi, \vartheta) : (U, A) \to (U', A')$ consists of a morphism $\phi : U \to U'$ of type universes together with a polymorphic translation $\vartheta : A \to (\mathsf{G}\phi)^* A'$.

Universe shift. The category $\mathbf{Poly}(\Sigma)$ supports a notion of translation between polymorphic structures on possibly different universes, with maps that provide two-levelled translations for types and for terms over types. As an application, we indicate how one can transport a meta-term on a universe to another one.

Let $(\!|-|\!) : W \to U$ be a morphism of type universes, translating types $\tau \in W(n)$ to types $(\!|\tau|\!) \in U(n)$. The functor $\mathsf{F}(\!|-|\!) : \mathsf{F}W \to \mathsf{F}U$ translates thus contexts $\Gamma \in \mathbb{F}{\downarrow}(Wn)$ to contexts $(\!|\Gamma|\!) = \big((\mathbb{F}{\downarrow}(\!|-|\!)_n) \circ \Gamma\big) \in \mathbb{F}{\downarrow}(Un)$. Furthermore, the functor $\big|\mathsf{G}(\!|-|\!)\big|_! : \mathbf{Set}^{|\mathsf{G}W|} \to \mathbf{Set}^{|\mathsf{G}U|}$ translates a set of metavariable declarations Z on the universe W to the set of metavariable declarations $(\!|Z|\!)$ on the universe U given by

$$\mathrm{z} \in (\!|Z|\!)\big(\, n \ \mid \ (\!|\Gamma|\!) \vdash (\!|\tau|\!)\,\big) \iff \mathrm{z} \in Z(\, n \ \mid \ \Gamma \vdash \tau\,) \ .$$

Finally, we have the assignment $Z \to \big|\big(\mathsf{G}(\!|-|\!)\big)^* \mathcal{N}_U \underline{(\!|Z|\!)}\big|$, mapping the metavariable $\mathrm{z} \in Z(n \mid \overrightarrow{x} : \overrightarrow{\sigma} \vdash \tau)$ to the meta-term $\mathrm{z}[\overrightarrow{x}] \in \mathcal{N}_U \underline{(\!|Z|\!)}\big(n \mid \overrightarrow{x} : \overrightarrow{(\!|\sigma|\!)} \vdash (\!|\tau|\!)\big)$, that induces the polymorphic translation

$$(\!|-|\!) : \mathcal{N}_W \underline{Z}(\, n \mid \Gamma \vdash \tau) \to \mathcal{N}_U \underline{(\!|Z|\!)}\big(\, n \mid (\!|\Gamma|\!) \vdash (\!|\tau|\!)\big)$$

whose effect is to translate types within meta-terms.

6. 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

$$Z \ \triangleright \ n \ \mid \ \Gamma \vdash_W s = t : \tau \tag{12}$$

where W is a type universe, $Z \in \mathbf{Set}^{|\mathsf{G}W|}$ is a set of metavariable declarations, $(n \mid \Gamma \vdash \tau) \in \mathsf{G}W$ is a context, 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

$$S \ \mid \ X \ \triangleright \ n \ \mid \ \Gamma \vdash s = t : \tau \ .$$

An *equational presentation* consists of a set of equational judgments, typically referred to as axioms.

Remark 6.1 In type theory, the type universe is given syntactically by means of a second-order equational presentation (see [8, § 5]) for the type signature. In PEL, this corresponds to working with universes of meta-types modulo equations. This can be done model theoretically by means of free second-order algebraic models for the equational presentation (see [8, § 8]) or logically by enriching PEL with type-equality judgments manipulated by second-order equational logic (see [8, Figs. 2 & 3]). We will however not dwell further on this here.

(Ref) $\dfrac{}{Z \triangleright n \mid \Gamma \vdash_U t = t : \tau}$

(Sym) $\dfrac{Z \triangleright n \mid \Gamma \vdash_U s = t : \tau}{Z \triangleright n \mid \Gamma \vdash_U t = s : \tau}$

(Tra) $\dfrac{Z \triangleright n \mid \Gamma \vdash_U s = t : \tau \quad Z \triangleright n \mid \Gamma \vdash_U t = u : \tau}{Z \triangleright n \mid \Gamma \vdash_U s = u : \tau}$

(Ax) $\dfrac{(Z \triangleright n \mid \Gamma \vdash_W s = t : \tau) \in E}{(\!|Z|\!) \triangleright n \mid (\!|\Gamma|\!) \vdash_U (\!|s|\!) = (\!|t|\!) : (\!|\tau|\!)}\ (\!|-|\!) : W \to U$

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

(Op) $\dfrac{S \triangleright f : k_1.(\overrightarrow{\sigma_1})\tau_1, \ldots, k_\ell.(\overrightarrow{\sigma_\ell})\tau_\ell \to \tau \in \Sigma^{\mathsf{Tm}} \quad |\overrightarrow{\alpha}_i| = k_i\ (1 \le i \le \ell) \qquad \theta : S \rightsquigarrow_n U \quad \theta_i = \iota_i^{[S \Rightarrow U]}(\theta) \quad (\iota_i : n \hookrightarrow n + k_i\ ,\ 1 \le i \le \ell) \qquad Z \triangleright n + k_i \mid \Gamma\theta_i,\ \overrightarrow{x_i : \sigma_i\theta_i} \vdash_U s_i = t_i : \tau_i\theta_i \quad (1 \le i \le \ell)}{Z \triangleright n \mid \Gamma \vdash_U f_\theta(\ldots, \overrightarrow{\alpha_i}.\overrightarrow{x_i}.s_i, \ldots) = f_\theta(\ldots, \overrightarrow{\alpha_i}.\overrightarrow{x_i}.t_i, \ldots) : \tau\theta}$

(MSub) $\dfrac{\vartheta : Z_0 \rightsquigarrow_{(k|\Delta)} \mathcal{N}_U\underline{Z} \quad \jmath : n \hookrightarrow k + n \qquad Z_0 \triangleright n \mid \Gamma \vdash_U s = t : \tau}{Z \triangleright k + n \mid \Delta, \Gamma \vdash_U s\vartheta = t\vartheta : \jmath(\tau)}$

(MVar) $\dfrac{\big(\mathrm{z} : \{n + k\}[\tau_1, \ldots, \tau_l]\tau\big) \in Z \qquad \sigma_i \in U(n) \quad (1 \le i \le k) \qquad Z \triangleright n \mid \Gamma \vdash_U s_i = t_i : \tau_i\{\sigma_1, \ldots, \sigma_n\} \quad (1 \le i \le l)}{Z \triangleright n \mid \Gamma \vdash_U \mathrm{z}\{\sigma_1, \ldots, \sigma_k\}[s_1, \ldots, s_l] = \mathrm{z}\{\sigma_1, \ldots, \sigma_k\}[t_1, \ldots, t_l] : \tau\{\sigma_1, \ldots, \sigma_k\}}$

(Act) $\dfrac{\rho : m \to n \qquad \pi : \rho(\Gamma) \to \Delta \qquad Z \triangleright m \mid \Gamma \vdash_U s = t : \tau}{Z \triangleright n \mid \Delta \vdash_U \pi\rho(s) = \pi\rho(t) : \rho(\tau)}$

(TSub) $\dfrac{Z \triangleright n + 1 \mid \Gamma \vdash_U s = t : \tau \qquad \sigma \in U(n)}{Z \triangleright n \mid \Gamma\{\sigma\} \vdash_U s\{\sigma\} = t\{\sigma\} : \tau\{\sigma\}}$

Fig. 2. Polymorphic Equational Logic (PEL)

Vernacular notation

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

Formal notation in PEL

$$\begin{array}{lllllll} (\beta) & \mathrm{S}, \mathrm{T} : * \mid \mathrm{M} \colon [\mathrm{S}]\mathrm{T},\ \mathrm{N} \colon \mathrm{S} \triangleright \vdash \mathsf{app}(\,\mathsf{abs}(x.\,\mathrm{M}[x]), \mathrm{N}\,) & = & \mathrm{M}[\mathrm{N}] & : & \mathrm{T} \\ (\text{type } \beta) & \mathrm{S} \colon *,\ \mathrm{T} \colon [*]* \mid \mathrm{M} \colon \{\alpha\}\mathrm{T}[\alpha] \quad \triangleright \vdash \mathsf{tapp}\big(\mathsf{tabs}(\alpha.\,\mathrm{M}\{\alpha\})\big) & = & \mathrm{M}\{\mathrm{S}\} & : & \mathrm{T}[\mathrm{S}] \end{array}$$

Fig. 3. Sample axioms for System F

Vernacular notation

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

Formal notation in PEL

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

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

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

Fig. 5. Sample axioms for global state

A polymorphic structure A on a type universe U *satisfies* an equation as in (12) above whenever for every morphism of type universes $(\!|-|\!) : W \to U$ and all assignments $\vartheta : Z \rightsquigarrow_{(k|\Delta)} A\big(\mathsf{G}(\!|-|\!)\big)$, we have $s\vartheta = t\vartheta$ in $A\big(k+n \mid (\!|\Gamma, \Delta|\!) \vdash (\!|\jmath(\tau)|\!)\big)$. An *algebraic model* of an equational presentation is a polymorphic structure that satisfies all the axioms.

6.1. PEL

The deduction system of PEL is given by the inference rules of Fig. 2. These have been synthesised from the mathematical model of polymorphic structures on type universes. Besides the equivalence-relation and axiom rules common to all equational deductive systems, it consists of congruence rules that reflect the various algebraic aspects of polymorphism.

Specifically, the rules (Ref), (Sym), (Tra) are those of equivalence relations; the rule (Ax) stipulates that the translation of axioms in a type universe become theorems in another type universe. Finally, the rules (Sub), (Op), (MSub), (MVar), (Act), and (TSub) are respectively congruence rules for substitution, operators, meta-substitution, metavariables, presheaf action, and type-in-term substitution.

We also have the following derived and admissible rules.

Structural rules. Weakening, contraction, and permutation on metavariables are obtained as instances of (MSub). Weakening, contraction, and permutation on type and term contexts are obtained as instances of (Act).

Universe shift. In PEL, the universe of an axiom can be changed when instantiating it in the (Ax) rule. This property extends to the whole deductive system as the following admissible rule for universe shifting:

$$(\mathrm{USh})\frac{Z \ \triangleright\ n \ \mid\ \Gamma \vdash_W s = t : \tau}{(\!|Z|\!) \ \triangleright\ n \ \mid\ (\!|\Gamma|\!) \vdash_U (\!|s|\!) = (\!|t|\!) : (\!|\tau|\!)} \quad (\!|-|\!) : W \to U$$

6.2. Soundness and completeness

Let E be an equational presentation. For a type universe U and a set of metavariable declarations $Z \in \mathbf{Set}^{|\mathsf{G}U|}$, consider the quotient $\mathcal{N}_U^E \underline{Z}$ of $\mathcal{N}_U \underline{Z}$ in $\mathbf{Set}^{\mathsf{G}U}$ determined by the equivalence relation that identifies $s, t \in \mathcal{N}_U \underline{Z}(n \mid \Gamma \vdash \tau)$ whenever the equation $(Z \triangleright n \mid \Gamma \vdash_U s = t : \tau)$ is derivable from E in PEL. This construction yields the free algebraic model of E on $\underline{Z}$ for U, and is instrumental in establishing the following result.

Theorem 6.2 *Polymorphic Equational Logic is sound and complete (i.e. for all equational presentations E, an equation is derivable from E in PEL iff it is satisfied by all algebraic models of E).*

7. Examples

Example 7.1 Continuing with the examples of § 2, 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 7.2 (**Polymorphic 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:

$$\Gamma \vdash_{\mathrm{F}}\ s = t : \tau \implies \neg\neg\Gamma^\bullet \vdash_{\lambda\exists}\ [\![s]\!] = [\![t]\!] : \neg\neg\tau^\bullet\ . \quad (13)$$

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 $\Lambda_{\mathrm{F}} = \mathcal{N}^{\mathrm{F}}_{\mathbb{T}_{\mathrm{F}}} 0$ (resp. $\Lambda_\exists = \mathcal{N}^{\lambda_\exists}_{\mathbb{T}_\exists} 0$) 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 polymorphic structure $(\mathbb{T}_\exists, \Lambda_\exists^{\neg\neg})$ which, by (13), is an algebraic model of System F, and we have a polymorphic translation $\big((-)^\bullet, [\![-]\!]\big) : (\mathbb{T}_{\mathrm{F}}, \Lambda_{\mathrm{F}}) \to (\mathbb{T}_\exists, \Lambda_\exists^{\neg\neg})$.

Example 7.3 (**Categorical model of System F [27]**) 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 **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

$$\begin{aligned}
&\mathrm{U} = \big|\mathbb{C}(\Omega^{(-)}, \Omega)\big|\ , \\
&\mathrm{PL}(\,n \mid \Gamma \vdash \tau\,) = \mathbb{C}(\Omega^n, \Omega)\big(\textstyle\prod_{1 \le i \le |\Gamma|} \Gamma(i), \tau\big)\ .
\end{aligned}$$

Seely's categorical interpretation of the types and terms of System F determines algebra structures on U and PL. For example, for $(\forall : \langle * \rangle * \to *) \in \Sigma_{\mathrm{F}}^{\mathsf{Ty}}$, 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.

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. Polymorphic abstract syntax via Grothendieck construction. In *FoSSaCS'11*, pages 381–395, 2011.

[18] H. Hosoya, A. Frisch, and G. Castagna. Parametric polymorphism for XML. *ACM Trans. Program. Lang. Syst.*, 32(1), 2009.

[19] F. W. Lawvere. Adjointness in foundations. *Dialectica*, 23:281–296, 1969.

[20] R. E. Møgelberg. From parametric polymorphism to models of polymorphic FPC. *Math. Struct. in Compt. Sci.*, 19(4):639–686, 2009.

[21] M. Miculan and I. Scagnetto. A framework for typed HOAS and semantics. In *PPDP'03*, pages 184–194. ACM Press, 2003.

[22] R. Milner. A theory of type polymorphism in programming. *J. Comput. Syst. Sci.*, 17(3):348–375, 1978.

[23] I. Moerdijk and E. Palmgren. Wellfounded trees in categories. *Ann. Pure Appl. Logic*, 104:189–218, 2000.

[24] B. Pierce and D. Sangiorgi. Behavioral equivalence in the polymorphic pi-calculus. *J. ACM*, 47(3):531–584, 2000.

[25] G. Plotkin and J. Power. Notions of computation determine monads. In *FoSSaCS'02*, pages 342–356, 2002.

[26] J. Reynolds. Types, abstraction and parametric polymorphism. *Inform. Process.*, 83:513–523, 1983.

[27] R. A. G. Seely. Categorical semantics for higher order polymorphic lambda calculus. *J. Symb. Log.*, 52(4):969–989, 1987.

[28] C. Strachey. Fundamental concepts in programming languages. *Higher-Order and Symbolic Computation*, 13:11–49, 2000.