Modules over monads and operational semantics

André Hirschowitz, Tom Hirschowitz, Ambroise Lafont

Modules over monads and operational semantics

André Hirschowitz[1], Tom Hirschowitz[2], and Ambroise Lafont[3]

[1] Université Côte d'Azur, CNRS, France, ah@unice.fr
[2] Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, 73000 Chambéry, France, tom.hirschowitz@univ-smb.fr
[3] IMT Atlantique, Inria, LS2N CNRS, 44307, Nantes, France, ambroise.lafont@inria.fr

Abstract. This paper is a contribution to the search for efficient and high-level mathematical tools to specify and reason about (abstract) programming languages or calculi. Generalising the *reduction monads* of Ahrens et al., we introduce *operational monads*, thus covering new applications such as the π-calculus, Positive GSOS specifications, and the big-step, simply-typed, call-by-value λ-calculus. Finally, we design a notion of signature for operational monads that covers all our examples.

1 Introduction

The search for a mathematical notion of programming language goes back at least to Turi and Plotkin [33], who coined the name "Mathematical Operational Semantics", and explained how known classes of well-behaved rules for structural operational semantics, such as GSOS [10], can be categorically understood and specified via distributive laws and bialgebras. Their initial framework did not cover variable binding, and several authors have proposed variants which do [16, 15, 31], treating examples like the π-calculus. However, none of these approaches covers higher-order languages like the λ-calculus.

In recent work, following previous work on modules over monads for syntax with binding [20, 4], Ahrens et al. [5] introduce *reduction* monads, and show how they cover several standard variants of the λ-calculus. Furthermore, as expected in similar contexts, they propose a mechanism for specifying reduction monads by suitable signatures.

Our starting point is the fact that already the call-by-value λ-calculus does not form a reduction monad. Indeed, in this calculus, variables are placeholders for values but not for λ-terms; in other words, reduction concerns general terms but is stable under substitution by values only.

In the present work, we generalise reduction monads to what we call *operational* monads[4]. Let us explain the basic intuitions. Monads have long been recognised as a suitable notion for modelling syntax with substitution, especially in the presence of binding, so a monad T is thought of as modelling terms of the considered language. Reductions, however, are of a different nature, as their

[4] Our operational monads are markedly different from Turi and Plotkin's [33, §5]

variables are placeholders for terms (as opposed to reductions): they form what is called a T-*module* R. Such considerations led Ahrens et al. to define a reduction monad to consist of a monad T, equipped with a span $T \overset{s}{\leftarrow} R \overset{t}{\rightarrow} T$ of T-module morphisms, where s and t give the source and target of any reduction.

Our generalisation is quite mild: an *operational* monad is a span $\overline{S_1} \leftarrow R \rightarrow \overline{S_2}$ of T-modules, where $\overline{S_1}$ and $\overline{S_2}$ should be *free*, in the sense of being of the form $S_i \circ T$, for some functors S_i, $i = 1, 2$, called the *configuration* functors. Reduction monads are obtained by taking $\overline{S_1}$ and $\overline{S_2}$ to be just T. In the case of the call-by-value λ-calculus, T is the monad of values, and $\overline{S_1}$ and $\overline{S_2}$ both are the module of λ-terms (in big-step style, $\overline{S_2}$ is the module of values, see §7). As a first result, we prove that, just like reduction monads, operational monads can be seen as relative monads (Theorem 2).

We then investigate the issue of specification, or presentation, i.e., the generation of operational semantics from more basic data. However, specifying an operational monad is quite intricate, as it involves specifying the underlying monad T, both configurations functors S_1 and S_2, and finally the span $\overline{S_1} \leftarrow R \rightarrow \overline{S_2}$. For this reason, we introduce a general notion of *signature* $\mathscr{S}$ over any category $\mathscr{C}$, which induces a category $\mathscr{C}^{\mathscr{S}}$ of *models* and a forgetful functor $\mathscr{U}^{\mathscr{S}} \colon \mathscr{C}^{\mathscr{S}} \rightarrow \mathscr{C}$. As in *initial semantics* [25, 9], the initial object of $\mathscr{C}^{\mathscr{S}}$, when it exists, is thought of as presented by $\mathscr{S}$ and inherits a kind of abstract induction principle. In this case, $\mathscr{S}$ is deemed *effective*.

A crucial point for modelling the dependency of transition rules on the syntax is that signatures compose: denoting by $\mathscr{S} \colon \mathscr{C} \nrightarrow \mathscr{E}$ the fact that $\mathscr{E} = \mathscr{C}^{\mathscr{S}}$, this means that given any $\mathscr{C}_1 \xrightarrow{\mathscr{S}_1} \mathscr{C}_2 \xrightarrow{\mathscr{S}_2} \mathscr{C}_3$, there is a compound signature $\mathscr{C}_1 \xrightarrow{\mathscr{S}_2 \circ \mathscr{S}_1} \mathscr{C}_3$. We exploit this in §4.3 through a general contruction, as follows. We assume given an arbitrary functor $p \colon \mathscr{X} \rightarrow \mathscr{B}$, in our case the forgetful functor $\mathbf{p}$ mapping any operational monad to the underlying monad and configuration functors, and any effective signature $\mathscr{S}$ over $\mathscr{B}$. We then show that $\mathscr{S}$ lifts to a signature, say $\mathscr{S}'$, on $\mathscr{X}$, and introduce $(p, \mathscr{S})$-*vertical endofunctors* Σ on $\mathscr{X}^{\mathscr{S}'}$. We further show that any such pair $(\mathscr{S}, \Sigma)$ induces a signature for $\mathscr{X}$, and call the class of signatures so obtained *vertical*. We finally prove that, under suitable hypotheses which hold in the case of operational monads with $\mathbf{p}$, finitary vertical signatures are effective (Theorem 4).

Returning to the announced examples, we now want to specify them by vertical signatures, so we start in §5 by designing classes of effective signatures $\mathscr{S}$ for monads and configuration functors, exploiting the abundant literature on this topic. We should then construct a suitable $(\mathbf{p}, \mathscr{S})$-vertical endofunctor for each example, but such endofunctors are rather abstract, so we introduce in §6 a notion of *transition rule*, close in spirit to standard operational semantics. We further show that any family of transition rules generates a finitary $(\mathbf{p}, \mathscr{S})$-vertical endofunctor. Thus, calling *operational specification* any pair of an effective signature $\mathscr{S}$ and a family of such transition rules, we obtain (Corollary 1) that the signature induced by any operational specification is effective. Finally, we provide operational specifications for the $\overline{\lambda}\mu$-calculus [19], pure λ-calculus,

the π-calculus [30] given as a *reduction* (= unlabelled transition) relation, and even any Positive GSOS specification [10]. We cover the more advanced case of call-by-value, simply-typed λ-calculus in big-step style in §7.

Summary of contributions Our main contributions are thus

1. the notion of operational monad for modelling operational semantics;
2. the characterisation of operational monads as relative monads (Theorem 2);
3. the general notion of signature (Definition 10) and the effectiveness result for vertical signatures (Theorem 4);
4. the notion of operational specification and the associated effectiveness result (Corollary 1);
5. the detailed treatment of significant examples, notably the call-by-value, simply-typed λ-calculus in big-step style.

Related work Beyond the already evoked related work [5, 33], we partly build on the extensive literature on specifying syntax with variable binding [13, 4, 11, 12]. There is also a solid body of work on categorical approaches to rewriting with variable binding [18, 22, 2], which only covers transition relations that are stable under arbitrary contexts. Furthermore, Hirschowitz [24] proposes an alternative categorical approach to operational semantics, which is however only equipped with an insufficiently expressive specification technique [23], and has not yet been shown to apply to higher-order languages.

Regarding signatures, some authors [14, 8, 17] use notions of signatures involving some form of type dependency, which may be amenable to describing the dependency of transitions on terms and configurations. However, to our knowledge, these notions have never been applied to general operational semantics.

Finally, most of the material presented here is extracted from the third author's forthcoming PhD thesis [28].

Plan We first introduce operational monads in §2, and establish the link with relative monads in §3. In §4, we introduce our notion of signature and define the class of vertical signatures, and establish the effectiveness result. Building on the literature, we then devote §5 to constructing effective signatures for monads and functors. We continue in §6 by introducing operational specifications, and showing that they give rise to effective signatures. We exploit this in §7 to cover our main example, call-by-value, simply-typed λ-calculus. Finally, we conclude and give some perspectives in §8.

2 Operational monads

In this section, we introduce the main new mathematical notion of the paper, operational monads, which was already motivated by the case of call-by-value, simply-typed λ-calculus in §1. Let us start with a further motivating example.

Example 1. Consider the $\overline{\lambda}\mu$-calculus of [19]. Its grammar is given by

$$\begin{array}{lll} c ::= \langle e|\pi\rangle & & \text{Processes} \\ e ::= x \mid \mu\alpha.c \mid \lambda x.e & & \text{Programs} \\ \pi ::= \alpha \mid e \cdot \pi & & \text{Stacks,} \end{array}$$

where α and x range over two disjoint sets of variables, called *stack* and *program* variables respectively. Both constructions μ and λ bind their variable in the body. There are two reduction rules:

$$\langle \mu\alpha.c|\pi\rangle \to c[\alpha \mapsto \pi] \qquad\qquad \langle \lambda x.e|e' \cdot \pi\rangle \to \langle e[x \mapsto e']|\pi\rangle.$$

This may be organised in three layers: first, one defines the *syntax*, consisting of programs and stacks; then, one defines the set of *configurations* for the reduction relation, here processes (= pairs of a program and a stack); finally, one defines the reduction relation, as a graph over configurations.

The syntax may be modelled as a monad T on the category $\mathbf{Set}^2$: T maps any pair $X = (X_\mathbf{t}, X_\mathbf{s})$ to the pair $(T_\mathbf{t}(X), T_\mathbf{s}(X))$, where $T_\mathbf{t}(X)$ denotes the set of programs with free program variables in $X_\mathbf{t}$ and free stack variables in $X_\mathbf{s}$, and similarly $T_\mathbf{s}(X)$ denotes the set of stacks with free variables in X. Processes are then modelled by the functor $S \circ T$, where $S\colon \mathbf{Set}^2 \to \mathbf{Set}$ maps any $(X_\mathbf{t}, X_\mathbf{s})$ to $X_\mathbf{t} \times X_\mathbf{s}$. Reductions of processes with free variables in X then form a graph with vertices in $S(T(X))$, i.e., a set $R(X)$ of edges, equipped with a set map $R(X) \to S(T(X))^2$. As already mentioned, both $(S \circ T)^2$ and R are T*-modules* [32], in the sense that they support substitution by programs and stacks. The former is even *free* in the sense of Lemma 1 below.

Let us recall the definition of modules, and then introduce two constructions on them that finally lead to the definition of operational monads.

We fix categories $\mathscr{C}$ and $\mathscr{E}$ for the rest of this section.

Definition 1. *Let T be a monad over $\mathscr{C}$. A finitary, $\mathscr{E}$-valued T-*module *is a finitary functor $M\colon \mathscr{C} \to \mathscr{E}$ equipped with a T-*action*, i.e., a natural transformation $\rho\colon M \circ T \to M$ making both of the following diagrams commute.*

$$\begin{array}{ccc} M \circ T \circ T & \xrightarrow{M\circ\mu} & M \circ T \\ {\scriptstyle \rho\circ T}\downarrow & & \downarrow{\scriptstyle \rho} \\ M \circ T & \xrightarrow[\rho]{} & M \end{array} \qquad\qquad \begin{array}{ccc} M \circ id_\mathscr{C} & \xrightarrow{M\circ\eta} & M \circ T \\ & \searrow \quad \swarrow_{\rho} & \\ & M & \end{array}$$

Finitary, $\mathscr{E}$-valued T-modules form a category T-$\mathbf{Mod}_\phi(\mathscr{E})$ (or just T-$\mathbf{Mod}_\phi$ when $\mathscr{E}$ is clear), whose morphisms are natural transformations commuting with the action in the obvious sense.

Lemma 1. *The forgetful functor T-$\mathbf{Mod}_\phi(\mathscr{C}, \mathscr{E}) \to [\mathscr{C}, \mathscr{E}]_\phi$ to the (finitary) functor category has a left adjoint, which maps any functor F to $F \circ T$ with action given by monad multiplication. The obtained module is denoted by $\overline{F}$. Any module isomorphic to some module of this form is deemed* free*. Similarly, any T-module morphism of the form $\alpha \circ T\colon \overline{M_1} \to \overline{M_2}$ is deemed* free *and denoted by $\overline{\alpha}$.*

Lemma 2. *For any T-modules $M, N \colon \mathscr{C} \to \mathscr{E}$, the product $M \times N$ is a T-module, with action $(M \times N) \circ T = M \circ T \times N \circ T \to M \times N$.*

Definition 2. *An $\mathscr{E}$-valued* operational monad *on $\mathscr{C}$ consists of a monad T on $\mathscr{C}$, called the* term monad*, equipped with a span $\overline{S_1} \leftarrow R \to \overline{S_2}$ of T-modules, with $\overline{S_1}$ and $\overline{S_2}$ free.*

Remark 1. The restriction to *free* modules makes the technical development easier, and is used for establishing the equivalence with relative monads.

Terminology 1. *R, S_1, and S_2 are respectively called the* transition*,* source*, and* target *modules, and the functors $S_1, S_2 \colon \mathscr{C} \to \mathscr{E}$ underlying the free modules $\overline{S_1}$ and $\overline{S_2}$ are collectively called* configuration functors*. For compactness, we often pretend that $\mathscr{E}$ has products, and use pairings $R \to \overline{S_1} \times \overline{S_2}$ instead of spans.*

Remark 1 *In most cases that we consider, both configuration functors are equal: $S_1 = S_2$. Then, we speak about the configuration functor.*

Let us now organise operational monads into a category. Observing that the assignment $T \mapsto T\text{-}\mathbf{Mod}_\phi(\mathscr{E})$ extends to a functor $(-)\text{-}\mathbf{Mod}_\phi(\mathscr{E}) \colon \mathbf{Mnd}_\phi(\mathscr{C})^{op} \to \mathbf{CAT}$ from finitary monads on $\mathscr{C}$ to (locally small) categories, we put:

Definition 3. *Let $p_{\mathscr{C},\mathscr{E}} \colon \mathbf{Mod}_\phi(\mathscr{C},\mathscr{E}) \to \mathbf{Mnd}_\phi(\mathscr{C})$ denote the fibration corresponding to the functor $(-)\text{-}\mathbf{Mod}_\phi(\mathscr{E})$ under the Grothendieck construction [26].*

This Grothendieck construction is detailed in Appendix A (see Lemma 9). Concretely, $\mathbf{Mod}_\phi(\mathscr{C},\mathscr{E})$ has as objects pairs (T, M), where T is a finitary monad on $\mathscr{C}$ and M is a finitary, $\mathscr{E}$-valued T-module, and a morphism $(T, M) \to (T', M')$ is a monad morphism $\beta \colon T \to T'$, together with a natural transformation $\alpha \colon M \to M'$ making the square below left commute.

$$\begin{array}{ccc} M \circ T & \xrightarrow{\alpha \circ \beta} & M' \circ T' \\ {\scriptstyle\rho}\downarrow & & \downarrow{\scriptstyle\rho'} \\ M & \xrightarrow[\beta]{} & M' \end{array} \qquad \begin{array}{ccc} R & \xrightarrow{\gamma} & R' \\ {\scriptstyle\partial}\downarrow & & \downarrow{\scriptstyle\partial'} \\ \overline{S_1} \times \overline{S_2} & \xrightarrow[\overline{\alpha_1} \times \overline{\alpha_2}]{} & \overline{S_1'} \times \overline{S_2'} \end{array}$$

Definition 4. *Let $\mathbf{OMnd}(\mathscr{C},\mathscr{E})$ denote the category of $\mathscr{E}$-valued operational monads on $\mathscr{C}$, with as morphisms all monad morphisms $\beta \colon T \to T'$, equipped with module morphisms $\overline{\alpha_1}, \overline{\alpha_2}, \gamma$ making the square above right commute, where $\overline{\alpha_1}$ and $\overline{\alpha_2}$ are free.*

Example 2. The call-by-value λ- and $\overline{\lambda}\mu$-calculi form operational monads, as sketched above.

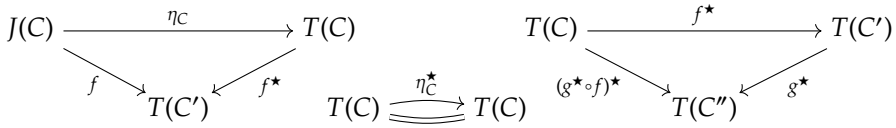
3 Operational monads as relative monads

In this section, we characterise operational monads as a category of relative monads [7]. This fact is not used in later sections. Let us recall the standard definition:

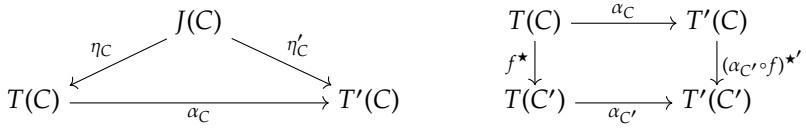
Definition 5. *A* monad T relative to a given functor $J\colon \mathscr{C} \to \mathscr{D}$ *consists of a map* $T\colon \mathbf{ob}(\mathscr{C}) \to \mathbf{ob}(\mathscr{D})$*, equipped with unit- and bind-like operations, mediated by* J*:*

- *for any* $C \in \mathscr{C}$*, a map* $\eta_C \in \mathscr{D}(J(C), T(C))$*,*
- *for any* $C, C' \in \mathscr{C}$ *and* $f \in \mathscr{D}(J(C), T(C'))$*, a map* $f^\star \in \mathscr{D}(T(C), T(C'))$*,*

such that the following diagrams commute for all $J(C) \xrightarrow{f} T(C')$ *and* $J(C') \xrightarrow{g} T(C'')$*.*



A morphism $(T, \eta, (-)^\star) \to (T', \eta', (-)^{\star'})$ consists of a family $\alpha_C\colon T(C) \to T'(C)$ of maps indexed by objects of $\mathscr{C}$, such that the following diagrams commute for all C, C', and $f\colon J(C) \to T(C')$.



Relative monads in particular induce functors $\mathscr{C} \to \mathscr{D}$, and are deemed *finitary* when the latter are.

Proposition 1. *Finitary monads relative to* J *and morphisms between them form a category* $\mathbf{RMnd}(J)$*.*

Now, assume $\mathscr{E}$ has an initial object 0 and consider any functors $S_1, S_2\colon \mathscr{C} \to \mathscr{E}$. Furthermore, letting $S = S_1 \times S_2$, we consider the functor $J_S\colon \mathscr{C} \to \mathscr{E}/S$ maping any C to the unique map $0 \to S(X)$ viewed as an object of the comma category $\mathscr{E}/S$. A monad R relative to J_S maps any $C \in \mathscr{C}$ to some $\partial^R_C\colon E^R(C) \to S(T^R(C))$. The unit map on C boils down to just a map $C \to T^R(C)$, and the bind map turns any map $f\colon C \to T^R(C')$ into maps $f^\sharp$ and $f^\star$ making the following square commute.

$$\begin{array}{ccc} E^R(C) & \xrightarrow{f^\sharp} & E^R(C') \\ {\scriptstyle \partial^R_C}\downarrow & & \downarrow{\scriptstyle \partial^R_{C'}} \\ S(T^R(C)) & \xrightarrow[S(f^\star)]{} & S(T^R(C')) \end{array}$$

This is precisely the structure needed to make T^R into a monad, and E^R into a module over it, which the axioms ensure is indeed the case. We thus have:

Lemma 3. *This defines an isomorphism* $\mathbf{RMnd}(J_S) \cong \mathbf{OMnd}(S)$*, where the latter denotes the restriction of* $\mathbf{OMnd}(\mathscr{C}, \mathscr{E})$ *to the given functor* S*.*

Proof. See Appendix C.

In fact, the assignments $S \mapsto \mathbf{OMnd}(S)$ and $S \mapsto \mathbf{RMnd}(J_S)$ extend to functors $[\mathscr{C},\mathscr{E}]_\phi^{op} \to \mathbf{CAT}$. The total category (i.e., Grothendieck construction) of the former is $\mathbf{OMnd}(\mathscr{C},\mathscr{E})$. Letting $\mathbf{RMnd}(\mathscr{C},\mathscr{E}/-)$ denote the total category of the latter, we obtain:

Theorem 2. *Lemma 3 extends to an isomorphism*

$$\begin{array}{ccc} \mathbf{OMnd}(\mathscr{C},\mathscr{E}) & \xrightarrow{\quad\cong\quad} & \mathbf{RMnd}(\mathscr{C},\mathscr{E}/-) \\ & \searrow \quad \swarrow & \\ & [\mathscr{C},\mathscr{E}]_\phi & \end{array}$$

of Grothendieck fibrations over $[\mathscr{C},\mathscr{E}]_\phi$.

4 Signatures for operational monads

Now that we have characterised operational monads as relative monads, we turn to specifying them, i.e., characterising them as initial objects in some suitable category. The main difficulty is that given the successive layers of terms, equations, configurations, and transitions, the task may quickly look messy. We thus start by introducing general signatures in §4.1 and present a few useful constructions on them in §4.2. This provides us with a uniform framework for designing in §4.3 our class of vertical signatures for operational monads, which we finally prove are effective.

4.1 Signatures over a category

Definition 6. *An* arity *for a category* $\mathscr{C}$ *consists of a category* $\mathscr{D}$*, together with functors* $\mathscr{C} \underset{v}{\overset{u}{\rightleftarrows}} \mathscr{D}$ *with* p *where* u *and* v *are sections of* p*, i.e.,* $p \circ u = p \circ v = id_{\mathscr{C}}$*. An* action *of an arity* $A = (\mathscr{D}, p, u, v)$ *on an object* C *of* $\mathscr{C}$ *is a morphism* $h\colon u(C) \to v(C)$ *such that* $p(h) = id_C$*.*

The intuition here is that objects of $\mathscr{D}$ are like objects of $\mathscr{C}$ with bits of derived additional data, p being the forgetful functor, and that $u(C)$ and $v(C)$ specify potentially different additional data for any given object C.

Example 3. For a standard algebraic operation of arity n on sets (i.e., $\mathscr{C} = \mathbf{Set}$), we would take $\mathscr{D} = \mathbf{Set} \times \mathbf{Set}$, with p the first projection, and for u and v the functors $\mathbf{Set} \underset{\langle id, id\rangle}{\overset{\langle id,\Delta^n\rangle}{\rightrightarrows}} \mathbf{Set} \times \mathbf{Set}$, where $\Delta^n(X) = X^n$ by definition, so that $u(X) = (X, X^n)$ and $v(X) = (X, X)$. Because of the condition that $p(h) = id_C$, an action of this arity on any set X boils down to a mere map $X^n \to X$.

Example 4. More generally, any endofunctor $F\colon \mathscr{C} \to \mathscr{C}$ corresponds to the arity $(\mathscr{C}^2, p_1, \langle id, F\rangle, \langle id, id\rangle)$, where p_1 denotes the first projection, in the sense that actions are exactly F-algebras.

Of course, we can organise actions into a category:

Definition 7. *Let E be a family of arities over $\mathscr{C}$. The* category $\mathscr{C}^E$ of models of E *has as objects all $C \in \mathscr{C}$ equipped with an action of each arity in E, and as morphisms $(C_1, h_1) \to (C_2, h_2)$ those morphisms $f\colon C_1 \to C_2$ that commute with actions in the obvious sense. Let $\mathscr{U}^E\colon \mathscr{C}^E \to \mathscr{C}$ denote the forgetful functor.*

Remark 2. The category $\mathscr{C}^A$ of models of an arity $A = (\mathscr{D}, p, u, v)$ is the *inserter* [27] of (u, v) in the 2-category of functors to $\mathscr{C}$, commutative triangles, and transformations that become the identity upon post-composition with p.

Example 5. A morphism between two actions of the arity of Example 3 is just a map preserving the operation. A morphism between two actions of the arity of Example 4 is an algebra morphism.

An important point about arities is that, although they seem designed for specifying operations, they are expressive enough to specify equations. In order to explain this, let us recall:

Definition 8. *The* fibre $p^{-1}(c)$ *of a functor $p\colon \mathscr{D} \to \mathscr{C}$ over any $C \in \mathscr{C}$ is the subcategory of $\mathscr{D}$ spanning objects mapped to C and morphisms mapped to id_C by p.*

Definition 9. *An arity $(\mathscr{D}, p, u, v)$ over $\mathscr{C}$ is said* equational *if p is* discrete*, i.e., its fibres are discrete categories.*

Remark 2 *An action of an equational arity $(\mathscr{D}, p, u, v)$ on an object $C \in \mathscr{C}$ is always an identity morphism. Thus, C is equipped with an action if and only if $u(C) = v(C)$. Any morphism is then a morphism between actions.*

The crucial point of our signatures is their simple mechanism for specifying sort dependency. The idea is to define signatures as sequences of families of arities, each family over the category of models of the previous one.

Notation 3. *Let $E\colon \mathscr{C} \nrightarrow \mathscr{D}$ mean that $\mathscr{D} = \mathscr{C}^E$.*

Definition 10. *A* signature S over a category $\mathscr{C}$ *is a finite sequence*

$$\mathscr{C} = \mathscr{C}_0 \xrightarrow{E_0} \mathscr{C}_1 \ldots \mathscr{C}_i \xrightarrow{E_i} \mathscr{C}_{i+1} \ldots \mathscr{C}_n,$$

which we denote by $S\colon \mathscr{C} \nrightarrow \mathscr{C}_n$. The empty sequence is called the empty signature*.*

The category of models $\mathscr{C}^S$ *is $\mathscr{C}$ if S is empty, and $\mathscr{C}_n^{E_n}$ otherwise, with* forgetful functor $\mathscr{U}^S$ *given by the composite*

$$\mathscr{C}_n^{E_n} \xrightarrow{\mathscr{U}^{E_n}} \mathscr{C}_{n-1}^{E_{n-1}} \xrightarrow{\mathscr{U}^{E_{n-1}}} \ldots \xrightarrow{\mathscr{U}^{E_1}} \mathscr{C}_0^{E_0} \xrightarrow{\mathscr{U}^{E_0}} \mathscr{C}_0 = \mathscr{C}.$$

Definition 11. *A signature S is* effective *if its category of models has an initial object $S^{c\star}$.*

4.2 Combining signatures

Signatures may be combined in several useful ways. The first, easiest way is *composition*:

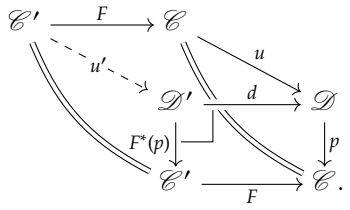
Definition 12. *For any signatures* $\mathscr{C} \overset{S}{\nrightarrow} \mathscr{D} \overset{T}{\nrightarrow} \mathscr{E}$ *with* $S = (E_0, \ldots, E_m)$ *and* $T = (F_0, \ldots, F_n)$*, let* $T \circ S$ *denote the composite signature* $(E_0, \ldots, E_m, F_0, \ldots, F_n)$

We may also *reindex* signatures along functors:

Lemma 4. *Given any functor* $F \colon \mathscr{C}' \to \mathscr{C}$ *and signature* S *over* $\mathscr{C}$*, there is a signature* $F^*(S)$ *whose category of models is the pullback*

$$\begin{array}{ccc} (\mathscr{C}')^{F^*(S)} & \xrightarrow{F^S} & \mathscr{C}^S \\ {\scriptstyle \mathscr{U}^{F^*(S)}}\downarrow & & \downarrow{\scriptstyle \mathscr{U}^S} \\ \mathscr{C}' & \xrightarrow[F]{} & \mathscr{C}. \end{array} \tag{1}$$

Proof. For any arity $(\mathscr{D}, p, u, v)$ over $\mathscr{C}$, we take the pullback $F^*(p)$ is of F and p, as suggested by the notation, and the relevant sections u' and v' follow by universal property, as in the diagram below. The rest follows inductively.



We may finally combine both operations to obtain the following effect.

Proposition 2. *Consider signatures* S_1 *and* S_2 *on* $\mathscr{C}_1$ *and* $\mathscr{C}_2$*, respectively. There is a signature* $S_1 \otimes S_2$ *on* $\mathscr{C}_1 \times \mathscr{C}_2$ *such that* $(\mathscr{C}_1 \times \mathscr{C}_2)^{S_1 \otimes S_2} = \mathscr{C}_1^{S_1} \times \mathscr{C}_2^{S_2}$*, which is effective if* S_1 *and* S_2 *are.*

Proof. Using Lemma 4, we first obtain a signature $p_1^* S_1 \colon \mathscr{C}_1 \times \mathscr{C}_2 \nrightarrow \mathscr{C}_1^{S_1} \times \mathscr{C}_2$. Then, reindexing S_2 along the second projection $p_2 \colon \mathscr{C}_1^{S_1} \times \mathscr{C}_2 \to \mathscr{C}_2$, we obtain $p_2^* S_2 \colon \mathscr{C}_1^{S_1} \times \mathscr{C}_2 \nrightarrow \mathscr{C}_1^{S_1} \times \mathscr{C}_2^{S_2}$, as desired. Effectiveness follows directly.

4.3 Vertical signatures

Let us now introduce the more complex construction that will lead to our class of effective signatures for operational monads.

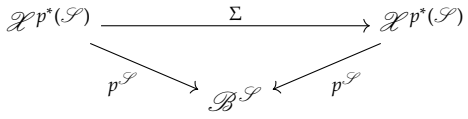
The idea is to first specify terms and configurations, and then transitions. So we want to start from a signature over the following category.

Definition 13. *Let* $\mathbf{B}(\mathscr{C}, \mathscr{E})$ *denote the category* $\mathbf{Mnd}_\phi(\mathscr{C}) \times [\mathscr{C}, \mathscr{E}]^2_\phi$ *of triples of a monad and two configuration functors.*

Let $\mathscr{S}$ denote any signature for $\mathbf{B}(\mathscr{C},\mathscr{E})$. The idea is to lift $\mathscr{S}$ to $\mathbf{OMnd}(\mathscr{C},\mathscr{E})$, by reindexing along the forgetful functor $\mathbf{p}\colon \mathbf{OMnd}(\mathscr{C},\mathscr{E}) \to \mathbf{B}(\mathscr{C},\mathscr{E})$, thus yielding a signature $\mathbf{p}^*(\mathscr{S})$ over $\mathbf{OMnd}(\mathscr{C},\mathscr{E})$, and specify transitions as a signature over models of $\mathbf{p}^*(\mathscr{S})$.

This may in fact be done abstractly, as follows.

Definition 14. *Given a functor $p\colon \mathscr{X} \to \mathscr{B}$ and a signature $\mathscr{S}$ over $\mathscr{B}$, a* $(p,\mathscr{S})$-vertical *endofunctor is an endofunctor Σ on $\mathscr{X}^{p^*(\mathscr{S})}$ over $\mathscr{B}^{\mathscr{S}}$, i.e., such that the following triangle commutes, where $p^{\mathscr{S}}$ denotes the forgetful functor as in* (1).

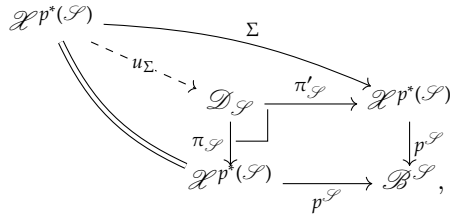


We expect the models of any $(p,\mathscr{S})$-vertical endofunctor Σ to be certain algebras:

Definition 15. *For any functor $F\colon \mathscr{C}_1 \to \mathscr{C}_2$, a morphism $h\colon x \to x'$ in $\mathscr{C}_1$ is* vertical *when $F(h) = id_{F(x)}$.*

The models *of an $(p,\mathscr{S})$-vertical endofunctor Σ are those Σ-algebras whose structure map is* vertical *in this sense.*

Let us now show that any $(p,\mathscr{S})$-vertical endofonctor Σ induces a signature over $\mathscr{X}^{p^*(\mathscr{S})}$, whose models are vertical Σ-algebras. Indeed, defining $\mathscr{D}_{\mathscr{S}}$ and $\pi_{\mathscr{S}}$ by the pullback



Σ induces the arity $A_{\mathscr{S},\Sigma} \coloneqq (\mathscr{D}_{\mathscr{S}}, \pi_{\mathscr{S}}, u_\Sigma, \Delta)$, where u_Σ denotes the displayed mediating morphism and the diagonal functor Δ is defined similarly from the trivial commuting square $p^{\mathscr{S}} \circ id = p^{\mathscr{S}} \circ id$. Let us note that giving a section of $\pi_{\mathscr{S}}$ is equivalent to giving an $(p,\mathscr{S})$-vertical functor $\mathscr{X}^{p^*(\mathscr{S})} \to \mathscr{X}^{p^*(\mathscr{S})}$. We finally obtain our notion of vertical signature, by combining $\mathscr{S}$ and Σ.

Definition 16. *A signature over $\mathscr{X}$ is* p-vertical *iff it has the shape $A_{\mathscr{S},\Sigma} \circ p^*(\mathscr{S})$ for some effective signature $\mathscr{S}$ over $\mathscr{B}$ and $(p,\mathscr{S})$-vertical endofunctor Σ. It is* finitary *when all restrictions Σ to fibres of p are.*

A vertical signature *is a p-vertical signature for some p.*

Here is thus our main effectiveness result:

Theorem 4. *If $\mathscr{X}$ has an initial object and p is a Grothendieck fibration with cocomplete fibres, then any finitary p-vertical signature is effective.*

Proof. One checks that the category of vertical algebras is fibered over the category $\mathscr{B}^{\mathscr{S}}$. Thus, the initial object is the initial object in the initial fibre, so it suffices to find an initial algebra for the endofunctor Σ restricted to the initial fiber. This follows from finitarity by Adámek's theorem [1].

Let us now show that the theorem applies to operational monads.

Lemma 5. *The forgetful functor* $\mathbf{p}\colon \mathbf{OMnd}(\mathscr{C},\mathscr{E}) \to \mathbf{B}(\mathscr{C},\mathscr{E})$ *is a Grothendieck fibration whose fibres are cocomplete.*

5 Specifying monads and functors

In §4, we have defined general vertical signatures, which instantiate to a class of effective signatures for operational monads. We now want to exploit this by covering the announced examples. In preparation for this, we devote this section to recasting existing techniques for specifying the underlying monad and configuration functors as general signatures over the category $\mathbf{B}(\mathscr{C},\mathscr{E})$ of Definition 13.

First of all, by Proposition §2, taking as parameters the base category $\mathscr{C}$ and the target category $\mathscr{E}$ for configuration functors, giving effective signatures $\mathscr{S}_0, \mathscr{S}_1$, and $\mathscr{S}_2$, respectively for $\mathbf{Mnd}_\phi(\mathscr{C}), [\mathscr{C},\mathscr{E}]_\phi$, and $[\mathscr{C},\mathscr{E}]_\phi$, automatically yield an effective signature for $\mathbf{B}(\mathscr{C},\mathscr{E})$. We thus first consider monads in §5.1 and functors in §5.2. (More detail may be found in Appendices A and B.)

5.1 Specifying monads

In this section, we build on [11] to construct signatures for monads. Let $\mathbb{I}$ be a set of *types*.

Definition 17. *A* $\mathbb{I}$-binding arity *is a tuple* $((\vec{t_1}, u_1), \ldots, (\vec{t_m}, u_m), u) \in (\mathbb{I}^\star \times \mathbb{I})^\star \times \mathbb{I}$, *where, for any set* X, $X^\star$ *denotes the set of finite sequences of elements of* X. *We denote such a tuple by*

$$\Theta_{u_1}^{\vec{t_1}} \times \ldots \times \Theta_{u_m}^{\vec{t_m}} \Rightarrow \Theta_u.$$

An $\mathbb{I}$-binding signature *is a family of binding* $\mathbb{I}$*-arities.*

The intuition is that an operation of this arity takes m arguments and returns a result of type u. The i^{th} argument must be of type u_i and binds n_i variables of types $\vec{t_i} = (t_i^1, \ldots, t_i^{n_i})$.

Example 6. We define an $\mathbb{I}$-binding signature for $\overline{\lambda}\mu$-calculus, with $\mathbb{I} = \{\mathbf{t}, \mathbf{s}\}$: it consists of three operations, $\cdot$, λ, and μ, with respective arities $(\Theta_{\mathbf{t}} \times \Theta_{\mathbf{s}} \Rightarrow \Theta_{\mathbf{s}})$, $(\Theta_{\mathbf{t}}^{\mathbf{t}} \Rightarrow \Theta_{\mathbf{t}})$, and $(\Theta_{\mathbf{t}}^{\mathbf{s}} \times \Theta_{\mathbf{s}}^{\mathbf{s}} \Rightarrow \Theta_{\mathbf{t}})$.

Following [6], any $\mathbb{I}$-binding arity B induces an arity for monads in the sense of §4: we take as forgetful functor $p_{\mathbf{Set}^{\mathbb{I}},\mathbf{Set}}\colon \mathbf{Mod}_\phi(\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}) \to \mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})$, so with sections determined by

$$B^-(T) = T_{u_1}^{\vec{t_1}} \times \ldots \times T_{u_m}^{\vec{t_m}} \qquad \text{and} \qquad B^+(T) = T_u,$$

where for any $M \in T\text{-}\mathbf{Mod}_\phi(\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}})$, we define $M_t^{\vec{l}} \in T\text{-}\mathbf{Mod}_\phi(\mathbf{Set}^{\mathbb{I}}, \mathbf{Set})$ by $X \mapsto M(X + \mathbf{y}_{\vec{l}})_\tau$, with $\mathbf{y}_{(l_1,\dots,l_n)} \coloneqq \mathbf{y}_{l_1} + \dots + \mathbf{y}_{l_n}$.

Theorem 5. *The signature for finitary monads on* $\mathbf{Set}^{\mathbb{I}}$ *induced by any* $\mathbb{I}$*-binding signature* Σ *has a model category* $\mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})^\Sigma$ *equivalent to the category of models of* Σ *in the sense of [11]. As a consequence,* Σ *is effective.*

Example 7. For simply-typed λ-abstraction, the source module of $(\Theta_{\mathbf{t}}^{\mathbf{t}} \Rightarrow \Theta_{\mathbf{t}})$ maps any monad T to $T_{\mathbf{t}}^{\mathbf{t}}$, and its target to $T_{\mathbf{t}}$. Thus, a model comes equipped with maps $T(X + 1) \to T(X)$ for all sets X, which are natural in X. Operations like $(-)^{\mathbf{t}}$ may thus be thought of as introducing a fresh variable of type $\mathbf{t}$.

Example 8. A model of the $\mathbb{I}$-binding signature for $\overline{\lambda}\mu$-calculus (Example 6) is a finitary monad T on $\mathbf{Set}^{\mathbb{I}}$ with T-module morphisms $(T_{\mathbf{t}} \times T_{\mathbf{s}} \to T_{\mathbf{s}})$, $(T_{\mathbf{t}}^{\mathbf{t}} \to T_{\mathbf{t}})$, and $(T_{\mathbf{t}}^{\mathbf{s}} \times T_{\mathbf{s}}^{\mathbf{s}} \to T_{\mathbf{t}})$. Morphisms of models are monad morphisms commuting with these module morphisms in the obvious sense.

Remark 3. We could generalise our approach by considering equations on terms, as we do in the next section for configurations. But we refrain from doing it as it is not needed in the covered examples.

5.2 Specifying functors

Let us now adapt binding arities for specifying objects of $[\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}}]_\phi$ (denoted by $\mathscr{C}$ in this section), that is, finitary functors $\mathbf{Set}^{\mathbb{I}} \to \mathbf{Set}^{\mathbb{J}}$, for any sets $\mathbb{I}$ and $\mathbb{J}$. In order to cover the example of π-calculus, we need to be slightly more general than with $\mathbb{I}$-binding arities for monads. Calling $\mathbb{I}^\star \times (\mathbb{I}^\star \times \mathbb{J})^\star \times \mathbb{J}$ the set of $(\mathbb{I}, \mathbb{J})$-*binding arities* and families thereof $(\mathbb{I}, \mathbb{J})$-*binding signatures*, as we did for $\mathbb{I}$-binding arities, we may recast any $(\mathbb{I}, \mathbb{J})$-binding arity as a pair of sections of the first projection functor $p_{\mathbb{I},\mathbb{J}} \colon [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}}]_\phi \times [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}]_\phi \to [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}}]_\phi$, which amount to functors $[\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}}]_\phi \to [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}]_\phi$. Indeed, let us adapt the notation for $\mathbb{I}$-binding arities, denoting by $\underline{\mathrm{Id}}_t$ arguments of the shape $t \in \mathbb{I}$ and by $\Theta_u^{\vec{t}}$ arguments in $(\mathbb{I}^\star \times \mathbb{J})^\star$ (where we omit u if $\mathbb{J} \cong 1$), any B of the form

$$(\underline{\mathrm{Id}}_{v_1} \times \dots \times \underline{\mathrm{Id}}_{v_p} \times \Theta_{u_1}^{\vec{t_1}} \times \dots \times \Theta_{u_m}^{\vec{t_m}} \Rightarrow \Theta_u)$$

corresponds to functors $B^-, B^+ \colon [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}}]_\phi \to [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}]_\phi$ with

$$B^-(S)(X) = X(v_1) \times \dots \times X(v_p) \times S(X + \mathbf{y}_{\vec{t_1}})(u_1) \times \dots \times S(X + \mathbf{y}_{\vec{t_m}})(u_m),$$

and $B^+(S)(X) = S(X)(u)$. Any $(\mathbb{I}, \mathbb{J})$-binding signature thus induces a signature, still denoted by Σ, hence a category $[\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}}]_\phi^\Sigma$ of models consisting of finitary functors $F \colon \mathbf{Set}^{\mathbb{I}} \to \mathbf{Set}^{\mathbb{J}}$ equipped with an *action* $B^-(F) \to B^+(F)$, for each binding arity B in Σ. Morphisms are natural transformations commuting with these actions in the obvious sense.

Lemma 6. *The signature induced by any* $(\mathbb{I}, \mathbb{J})$*-binding signature is effective.*

Example 9. The configuration functor for $\overline{\lambda}\mu$-calculus has $\mathbb{I} = \{\mathbf{t}, \mathbf{s}\}$, $\mathbb{J} = 1$, and just one arity, $\underline{\mathrm{Id}}_{\mathbf{t}} \times \underline{\mathrm{Id}}_{\mathbf{s}} \Rightarrow \Theta$. Models are just given by the coslice under $\mathrm{Id}_{\mathbf{t}} \times \mathrm{Id}_{\mathbf{s}}$.

Example 10. Let us now consider the π-calculus [30, §1.1 and 1.2], omitting sums and silent actions for readability. As explained in §1, terms should be the things by which we substitute, i.e., in this case, mere channels. The monad should thus be the identity monad, that is, the initial monad, which is specified by the empty binding signature. Furthermore, since reduction relates processes, the latter should be considered as configurations, in the sense that $S_1(X)$ and $S_2(X)$ should both be the set of processes with free channels in X.

We may specify this as follows. First of all, in this case $\mathbb{I} = \mathbb{J} = 1$, so binding arities boil down to a natural number p in $1^\star$, plus a sequence of natural numbers, the successive lengths $\vec{n} = (n_1, \ldots n_q)$ of the relevant sequences of the unique element of 1. We denote such an arity by $(\underline{\mathrm{Id}}^p \times \Theta^{(n_1)} \times \ldots \times \Theta^{(n_q)} \Rightarrow \Theta)$ where n_i is omitted if null, p is omitted if equal to 1, and $\underline{\mathrm{Id}}^p$ is omitted if p is null.

Using this notation, e.g., input $a(b).P$ has as $(1,1)$-binding arity $\underline{\mathrm{Id}} \times \Theta^{(1)} \Rightarrow \Theta$, parallel composition $P|Q$ has arity $\Theta \times \Theta \Rightarrow \Theta$, and output $\overline{a}\langle b\rangle.P$ has arity $\underline{\mathrm{Id}}^2 \times \Theta \Rightarrow \Theta$. Models correspond to functors S equipped with natural transformations $X \times S(X+1) \to S(X)$, $S(X)^2 \to S(X)$, and $X^2 \times S(X) \to S(X)$.

In order to fully specify the π-calculus, however, we need to quotient out processes by equations such as associativity of parallel composition

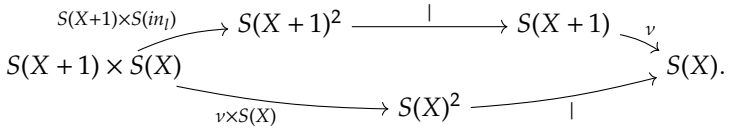
$$P|(Q|R) \equiv (P|Q)|R. \tag{2}$$

The following notion of equation allows to do just this. Equation (2) may be decomposed into its *metavariables*, P, Q, and R, and two *metaterms*, the left- and right-hand sides. In this case, viewing the (non-quotiented) syntax as specified by a $(1,1)$-binding signature Σ, we think of metavariables as the domain of the binding arity $B = (\Theta \times \Theta \times \Theta \Rightarrow \Theta)$. Then, $B^- \colon [\mathbf{Set}, \mathbf{Set}]_\phi \to [\mathbf{Set}, \mathbf{Set}]_\phi$ maps any endofunctor S and set X to $S(X)^3$. A metaterm then should associate to any $S \in \mathscr{C}$ with Σ-model structure a natural transformation $B^-(S) \to B^+(S)$. In the general case, this yields:

Definition 18. *Given a $(\mathbb{I},\mathbb{J})$-binding signature Σ over $\mathscr{C} = [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}}]_\phi$ with $\mathscr{U}^\Sigma \colon \mathscr{C}^\Sigma \to \mathscr{C}$ the forgetful functor, a* Σ-equation *$u = v \colon B$ consists of a $(\mathbb{I},\mathbb{J})$-binding arity B together with two natural transformations $u, v \colon B^- \circ \mathscr{U}^\Sigma \to B^+ \circ \mathscr{U}^\Sigma$ called the* metaterms.

Remark 4. In the untyped case, we have $B^+(S)(X) = S(X)$, so $B^+ = \mathrm{Id}$ and hence $B^+ \circ \mathscr{U}^\Sigma \cong \mathscr{U}^\Sigma$.

Example 11. The equations of *structural equivalence* [30, Table 1.1] may be encoded in this way. For (2), the involved metaterms $B^- \circ \mathscr{U}^\Sigma \to B^+ \circ \mathscr{U}^\Sigma$ at any Σ-model S have components $S(X)^3 \to S(X)$ mapping any triple (P, Q, R) to $(P|Q)|R$ and $P|(Q|R)$, where $|$ follows from the Σ-model structure of S. Other equations are treated similarly, the only subtle one being $\nu a.(P|Q) \equiv (\nu a.P)|Q$ when $a \notin Q$. For this, the relevant binding arity B is $\Theta^{(1)} \times \Theta \Rightarrow \Theta$ for the metavariables P and Q, and the metaterms have components at S and X given by



Proposition 3. *Any* $(\mathbb{I},\mathbb{J})$*-binding signature* Σ *with a family* $(u^i = v^i \colon B^i)_{i\in I}$ *of* Σ*-equations induces an effective general signature for* $\mathscr{C}$ *whose models are exactly those models* M *of* Σ *such that* $u^i_M = v^i_M$ *for all* i*.*

Proof. The proof, detailed in Appendix B, consists in rephrasing these constructions in terms of *equational systems* [12], and exploiting [12, Theorem 5.1 (1)]. The induced signature is a composite of two arities. The first component is the one induces by Σ. The second component consists of the equational arity $(\mathscr{D}, p, u, v)$ induced by a single equation $u = v : B$: $\mathscr{D}$ is the category consisting of finitary functors $F \colon \mathbf{Set}^{\mathbb{I}} \to \mathbf{Set}^{\mathbb{J}}$ equipped with a Σ-model structure, and a natural transformation $B^-(F) \to B^+(F)$. The forgetful functor p returns the underlying Σ-model, while u and v respectively assign to each model F of Σ the natural transformations u_F and v_F.

6 Transition rules

In this section, we design of way of presenting the $(p, \mathscr{S})$-vertical endofunctors of §4.3, akin to transition rules in the sense of operational semantics. We then introduce notation for it, which allows us to cover examples more conveniently in §7.

For this section, we restrict to cases where $\mathscr{C} = \mathbf{Set}^{\mathbb{I}}$ and $\mathscr{E} = \mathbf{Set}^{\mathbb{J}}$, and assume given a signature $\mathscr{S}$ for the category $\mathbf{B}(\mathscr{C}, \mathscr{E})$ of Definition 13. Typically, $\mathbb{J} = \mathbb{I}$, or $\mathbb{J} \cong 1$, and then $\mathscr{E} \cong \mathbf{Set}$. Consider, e.g., the left congruence rule of untyped λ-calculus (then, $\mathbb{I} = \mathbb{J} = 1$)

$$\frac{M \to M'}{M\ N \to M'\ N}\ . \tag{3}$$

In the spirit of equations in §5.2, it may be decomposed into metavariables, a sequence of premises, and a conclusion. Metavariables will be modelled as $\mathscr{S}$-modules, and premises and conclusion as pairs of metaterms, in the following sense.

Definition 19. *An* $\mathscr{S}$-module *is any functor* $\mathscr{V} \colon \mathbf{B}(\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}})^{\mathscr{S}} \to \mathbf{Mod}_{\phi}(\mathscr{C}, \mathbf{Set})$ *over* $\mathbf{Mnd}_{\phi}(\mathscr{C})$*, i.e., preserving the underlying monad.*

A metaterm $\mathscr{V} \to \mathscr{V}'$ *consists of a* vertical *natural transformation* α*, i.e., one such that* $p_{\mathscr{C},\mathscr{E}} \circ \alpha = id_{\mathscr{U}}$*, where* $\mathscr{U} = \pi_1 \circ \mathscr{U}^{\mathscr{S}}$*.*

Example 12. Rule (3) comprises three metavariables M, M', and N, so the metavariable $\mathscr{S}$-module maps any (T, S_1, S_2) to T^3.

We may now introduce transition rules.

Definition 20. *A* configuration pair $(\vec{I}, \tau, P)$ *over* $\mathscr{V}$ *consists of* $\vec{I} \in \mathbb{I}^\star$ *and* $\tau \in \mathbb{J}$*, together with a metaterm pairing* $\mathscr{V} \xrightarrow{P} \kappa^{\vec{I}}_{\tau}$*, where* $\kappa(T, S_1, S_2) = \overline{S_1 \times S_2}$ $(= \overline{S_1} \times \overline{S_2})$*. When* $\vec{I}$ *is empty, the configuration pair is deemed* normal*.*

A transition rule $(\mathscr{V}, L, C)$ *consists of an* $\mathscr{S}$*-module* $\mathscr{V}$*, together with a sequence* L *of* premise *configuration pairs and a* conclusion *normal configuration pair* C*.*

We may at last define:

Definition 21. *An* operational specification *consists of an* $\mathbb{I}$*-binding signature* $\mathscr{S}_1$ *and* $(\mathbb{I}, \mathbb{J})$*-binding signatures* $\mathscr{S}_2$ *and* $\mathscr{S}_3$*, equipped with a family of transition rules over the tensor product* $\mathscr{S} = \mathscr{S}_1 \otimes \mathscr{S}_2 \otimes \mathscr{S}_3$*.*

As announced, we now show that operational specifications induce signatures.

Lemma 7. *Any operational specification* $\mathscr{R}$ *gives rise to an* $(\mathbf{p}, \mathscr{S})$*-vertical endofunctor* $F_{\mathscr{R}}$*, which is finitary on all fibres.*

Proof. Let us fix $B = (T, S_1, S_2) \in \mathbf{B}(\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}})^{\mathscr{S}}$, and describe the restriction of the endofunctor $F_{\mathfrak{R}}$ induced by any transition rule $\mathfrak{R} = (\mathscr{V}, L, C)$ on the fibre over B, i.e., the slice category $\mathscr{X}/\kappa(B)$, where $\mathscr{X} = T\text{-}\mathbf{Mod}_{\phi}(\mathbf{Set}^{\mathbb{J}})$.

First, each element P^i of the sequence $L = (P^1, \ldots, P^n)$ of premises is a configuration pair $P^i \colon \mathscr{V} \to \kappa(B)^{\vec{I^i}}_{\tau^i}$. Taking components at B and letting $\nabla(F) = \prod_{i \in n} F^{\vec{I^i}}_{\tau^i}$, we obtain the tupling $\mathscr{V}(B) \xrightarrow{\langle P^i_B \rangle_{i \in n}} \nabla(\kappa(B))$, which we denote by $\langle L_B \rangle$. Now, by the well-known adjunction $(-)_\tau \dashv (-) \cdot \mathbf{y}_\tau$, the conclusion $\mathscr{V} \xrightarrow{C} \kappa(B)_\tau$ yields a morphism $\mathscr{V}(B) \cdot \mathbf{y}_\tau \xrightarrow{\check{C}_B} \kappa(B)$. We have thus constructed a span

$$\nabla(\kappa(B)) \cdot \mathbf{y}_\tau \xleftarrow{\langle L_B \rangle \cdot \mathbf{y}_\tau} \mathscr{V}(B) \cdot \mathbf{y}_\tau \xrightarrow{\check{C}_B} \kappa(B) \qquad \text{in } \mathscr{X}.$$

Let now the $(\mathbf{p}, \mathscr{S})$-vertical endofunctor $F_{\mathfrak{R}}$ associated to $\mathfrak{R}$ be given over B by the composite

$$\mathscr{X}/\kappa(B) \xrightarrow{\nabla \cdot \mathbf{y}_\tau / \kappa(B)} \mathscr{X}/\nabla(\kappa(B)) \cdot \mathbf{y}_\tau \xrightarrow{\Delta_{\langle L_B \rangle \cdot \mathbf{y}_\tau}} \mathscr{X}/\mathscr{V}(B) \cdot \mathbf{y}_\tau \xrightarrow{\Sigma_{\check{C}_B}} \mathscr{X}/\kappa(B), \qquad (4)$$

where $\Delta_{\langle L_B \rangle}$ denotes pullback along $\langle L_B \rangle$ and $\sum_{\check{C}_B}$ postcomposition with $\check{C}_B$. Similarly, the $(\mathbf{p}, \mathscr{S})$-vertical endofunctor $F_{\mathscr{R}}$ associated to any family $\mathscr{R} = (\mathfrak{R}_i)_{i \in I}$ of transition rules is the coproduct $\sum_i F_{\mathfrak{R}_i}$.

Finally, to show that any $F_{\mathscr{R}}$ is finitary on fibres, it suffices to prove that each $F_{\mathfrak{R}_i}$ is, hence that each component of (4) is. But, exploiting the characterisation of T-modules as covariant presheaves over the Kleisli category $\mathbf{Kl}(T)$ of T [21], we obtain that $\mathscr{X}$ is equivalent to the category of covariant presheaves over $\mathbf{Kl}(T)_f \times \mathbb{J}$, where $\mathbf{Kl}(T)_f$ denotes the full subcategory of finitely presentable objects. Thus, both $\sum_C$ and $\Delta_{\langle L_B \rangle}$ are left adjoints, hence in particular finitary. But then, any forgetful functor from a slice creates colimits, so we reduce to

proving that ∇ is finitary. It is also well-known that filtered limits commute with finite limits in **Set**, hence in any presheaf category, so we further reduce to proving that $M \mapsto M^{\vec{l}}_{\tau}$ itself is, which clearly holds pointwise.

Corollary 1. *Any operational specification induces an effective signature.*

It may not be entirely clear at this stage that this construction produces meaningful signatures. In the hope of rectifying this, let us introduce our notation for transition rules and use it to cover some simple examples.

Any transition rule can be written as

$$\frac{M_1(\vec{X}) \rightsquigarrow N_1(\vec{X}) : \tau_1 \qquad \dots \qquad M_n(\vec{X}) \rightsquigarrow N_n(\vec{X}) : \tau_n}{M_0(\vec{X}) \rightsquigarrow N_0(\vec{X}) : \tau_0},$$

where M_i and N_i are expressions depending on metavariables $\vec{X} = (X_1, \dots, X_q)$, and $\tau_i \in \mathbb{J}$ (omitted when $\mathbb{J} \cong 1$). Each pair (M_i, N_i) defines a configuration pair

$$\begin{aligned} P_i \colon \Gamma_1 \times \cdots \times \Gamma_q &\to (\sigma_1)^{\vec{l_i}}_{\tau_i} \times (\sigma_2)^{\vec{l_i}}_{\tau_i} \\ (X_1, \dots, X_q) &\mapsto (M_i(X_1, \dots, X_q), N_i(X_1, \dots, X_q)), \end{aligned}$$

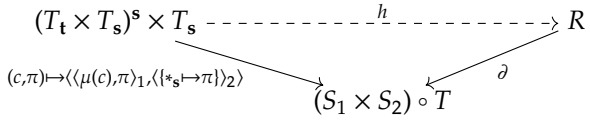
where $\vec{l_0} = \varnothing$, and the $\mathscr{S}$-modules $\Gamma_1, \dots, \Gamma_q$, the sequences $\vec{l_1}, \dots, \vec{l_n}$ are inferred to make this well-defined for all $i \in \{0, \dots, n\}$. The denoted reduction rule has as $\mathscr{S}$-module $\mathscr{V} = \Gamma_1 \times \dots \times \Gamma_q$, and as premises and conclusion the configuration pairs $(\vec{l_i}, \tau_i, P_i)_{i \in \{1,\dots,n\}}$ and $(\varnothing, \tau_0, P_0)$. Typically, $\Gamma_i = \Theta^{\vec{l}}_t$ for $\vec{l} \in \mathbb{I}^\star$ and $t \in \mathbb{I}$.

Remark 5. A reduction rule written in this way remains ambiguous as one should choose an ordering on metavariables for defining the relevant $\mathscr{S}$-module. However, all choices yield equivalent model categories.

Example 13 (Reduction rules for $\overline{\lambda}\mu$-calculus). Consider the reduction rule

$$\langle \mu\alpha.c | \pi \rangle \to c[\alpha \mapsto \pi]$$

of $\overline{\lambda}\mu$-calculus, recalling that in this case $\mathbb{I} = \{\mathbf{t}, \mathbf{s}\}$ and $\mathbb{J} = 1$. Furthermore, an object $B = (T, S_1, S_2)$ of $\mathbf{B}(\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}})^{\mathscr{S}}$ is in particular equipped with a T-module morphism $\mu \colon (T_{\mathbf{t}} \times T_{\mathbf{s}})^{\mathbf{s}} \to T_{\mathbf{t}}$ corresponding to μ-abstraction, and with natural transformations $\langle\rangle_i \colon (-)_{\mathbf{t}} \times (-)_{\mathbf{s}} \to S_i$, for $i = 1, 2$. The rule may thus be understood as $\langle \mu(c), \pi \rangle_1 \rightsquigarrow \langle c\{*_{\mathbf{s}} \mapsto \pi\} \rangle_2$, where $-\{*_{\mathbf{s}} - \}$ denotes substitution of the additional variable $*_{\mathbf{s}}$ from $\mathbf{y}_{\mathbf{s}}$. The $\mathscr{S}$-module $\mathscr{V}$ is $\Theta^{\mathbf{s}}_{\mathbf{t}} \times \Theta^{\mathbf{s}}_{\mathbf{s}} \times \Theta_{\mathbf{s}}$ for (c, π), as c is a process, i.e., a pair consisting of a program and a stack. The corresponding endofunctor takes any operational monad $\partial \colon R \to \overline{S_1} \times \overline{S_2}$ and first computes the pullback along $\langle L_B \rangle$, which, because there are no premises, is the terminal morphism. The resulting image is then $\mathscr{V}(B)$ itself. An algebra structure thus amounts to a morphism h making the following triangle commute.



Example 14. For an example with premises, recall the left congruence (3) rule of (untyped) λ-calculus. In this case, T comes with a module morphism $@\colon T^2 \to T$ for application, and each S_i comes with a natural transformation $\mathrm{Id} \to S_i$, which we treat as implicit. The premise configuration pair H is then $T^3 \xrightarrow{\langle \pi_1, \pi_2 \rangle} T^2 \hookrightarrow S_1 \circ T \times S_2 \circ T$, and the pullback of $\partial\colon R \to S_1 \circ T \times S_2 \circ T$ along H yields the module $H^*(R)$ such that for any set X, $H^*(R)(X)$ consists of all tuples (r, M, M', N) with $r \in R(X)$ a reduction, such that $\partial(r) = (M, M')$. Algebra structure then amounts to a morphism h making the following square commute,

$$\begin{array}{ccc} H^*(R) & \overset{h}{\dashrightarrow} & R \\ {\scriptstyle H^*(\partial)}\downarrow & & \downarrow{\scriptstyle \partial} \\ T^3 & \xrightarrow[\langle @\circ\langle \pi_1, \pi_3\rangle, @\circ\langle \pi_2, \pi_3\rangle\rangle]{} & T^2 \hookrightarrow S_1 \circ T \times S_2 \circ T \end{array}$$

i.e., mapping any such (r, M, M', N) so some reduction over $(@(M, N), @(M', N))$, as desired.

Example 15. Without entering into such detail, recall from Examples 10 and 11 the binding signature and equations for the syntax of π-calculus modulo structural congruence, thought of as a configuration functor. From there, reduction rules are easy:

$$\frac{}{out(a, b, x) | in(a, y) \rightsquigarrow x | (y\{* \mapsto b\})} \qquad \frac{x \rightsquigarrow x'}{x | y \rightsquigarrow x' | y} \qquad \frac{x \rightsquigarrow x'}{\nu(x) \rightsquigarrow \nu(x')}\,.$$

Example 16. Let us finally treat the case of GSOS rules [10], which differ from previous examples in that they specify labelled transitions $t \xrightarrow{a} u$: the intuition is that t reduces to u, exchanging information a with its environment. Fixing a set $\mathbb{A}$ of *labels* and a standard signature, i.e., a family of operations with arities in $\mathbb{N}$ (= untyped syntax, no binding), *Positive GSOS* rules have the shape

$$\frac{x_i \xrightarrow{a_{i,j}} y_{i,j}}{op(x_1, \ldots, x_n) \xrightarrow{c} M}\,,$$

where op has arity n, the x_i's and $y_{i,j}$'s are pairwise distinct variables, and M is a term potentially depending on them. A *Positive GSOS specification* is a family of such rules.

Apart from the labels, Positive GSOS rules look much like reduction rules from §6. But in fact, taking $S_1 = \mathrm{Id}$ and $S_2 = \mathbb{A} \times \mathrm{Id}$ as configuration functors directly handles labels. Both functors are easily specified: S_1 by just one operation of arity $\underline{\mathrm{Id}} \Rightarrow \Theta$ and S_2 by $\mathbb{A}$ operations of the same arity.

Remark 6. Negative GSOS rules allow hypotheses of the form $x \overset{a}{\not\to} y$, which may prevent reductions from forming a module (by not being stable under substitution), hence do not fit into our framework in general.

7 Call-by-value, simply-typed λ-calculus, big-step style

Let us finally consider the more advanced case of the call-by-value, simply-typed λ-calculus in big-step style. As already mentioned, the monad should consist of values. This requires us first to define values without having an application operation at hand, and then to characterise general terms as a module over them without an abstraction operation. Let first $\mathbb{I}$ denote the set of simple types over a fixed set of base types, and take $\mathscr{C} = \mathscr{E} = \mathbf{Set}^{\mathbb{I}}$. We first specify the monad T for values. The idea here is to encode each layer of applications by a single operation $@_\beta$, where β is the corresponding (well-typed) binary tree. Formally, we postulate an operation $\lambda_\beta \colon \Theta^A_{A_1} \times \ldots \times \Theta^A_{A_n} \Rightarrow \Theta_{A\to B}$ for each typed binary tree $A_1, \ldots, A_n \vdash \beta : B$, as inductively defined by the rules

$$\frac{}{B \vdash B : B} \qquad \frac{\Gamma \vdash \beta_1 : C \to D \qquad \Delta \vdash \beta_2 : C}{\Gamma, \Delta \vdash \beta_1\ \beta_2 : D}\ .$$

Example 17. The value $\lambda y : A \to A \to B.(y\ x)\ x$ becomes $\lambda_{(A\to A\to B)\ A\ A}(*, x, x)$, where the variables $x, * \in \{x\} + 1$ are implicitly embedded by the monad unit η.

Let us now specify configuration functors. In big-step style, we are specifying an evaluation relation, i.e., a relation between general terms and values. Thus, the first configuration functor S, for general terms, is specified a bit like the value monad, but without abstracting, i.e., by operations $@_\beta \colon \underline{\mathrm{Id}}_{A_1} \times \ldots \times \underline{\mathrm{Id}}_{A_n} \Rightarrow \Theta_B$. In particular, there is an embedding of variables into general terms $V_A \colon \underline{\mathrm{Id}}_A \Rightarrow \Theta_A$ by considering the trivial tree $A \vdash A \colon A$. The specified second configuration functor should be the identity, and is thus specified by arities $W_A \colon \underline{\mathrm{Id}}_A \Rightarrow \Theta_A$ for each type A.

Finally, transitions are specified by the axiom $V_A(x) \rightsquigarrow W_A(x) \colon A$ and the following rule, for all types A, B and tree β.

$$\frac{x \rightsquigarrow \lambda_\beta(x') \colon A \to B \qquad y \rightsquigarrow y' \colon A \qquad @_\beta(x'\{* \mapsto y'\}) \rightsquigarrow z \colon B}{@_{(A\to B)\,B}(x, y) \rightsquigarrow z : B}$$

8 Conclusion and perspectives

We have introduced operational monads as a generalisation of reduction monads, and demonstrated that they cover relevant new examples. We also have proved that they are equivalent to a certain category of relative monads, and introduced a notion of signature for them. We have finally characterised a class of effective signatures called vertical, together with a format for them close to standard operational semantics called operational specifications.

In future work, we plan on investigating other forms of configuration modules, e.g., using a single free module seems to cover type systems enjoying a substitution lemma, and using an arbitrary module covers the subtle labelled transition system for π-calculus.

Finally, it would be relevant to prove general theorems about the operational monads that we now know how to generate, typically about sufficient conditions for bisimilarity to be a congruence [29].

A Specifying monads

In this section, we recast $\mathbb{I}$-binding signatures (without equations) into the framework of general signatures. Such binding signatures have been used to specify monads on $\mathbf{Set}^{\mathbb{I}}$ in [6] and we could replay this work in the current restricted finitary setting. Before, binding signatures have been considered by [11] to specify relative monads on the injection of the free cocartesian cocompletion on $\mathbb{I}$ into $\mathbf{Set}^{\mathbb{I}}$: these are actually equivalent to finitary monads on $\mathbf{Set}^{\mathbb{I}}$. We focus here the latter approach to benefit from this equivalence.

We describe Fiore and Hur's [11] category of models of an $\mathbb{I}$-binding signature, slightly departing from their presentation, in the sense that we transfer their definition across the chain of equivalences

$$[\mathbb{F}[\mathbb{I}], \mathbf{Set}]^{\mathbb{I}} \simeq [\mathbf{Set}^{\mathbb{I}}_f, \mathbf{Set}^{\mathbb{I}}] \simeq [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{I}}]_\phi, \tag{5}$$

where $\mathbb{F}$ denotes free cocartesian completion. Eluding technical details, for any $\mathbb{I}$-binding signature Σ, their definition comprises two components, one for interpreting the operations specified by Σ, and the other for (capture-avoiding) substitution. Under the equivalence (5), the latter amounts to restricting attention to (finitary) monads.

Let us now describe the former, i.e., the interpretation of operations. Following [20], this relies on modules over monads.

Example 18. An important module, on any monad T on sets, is given by $T^T(X) = T(X+1)$, which indeed is the exponential object T^T in the category $T\text{-}\mathbf{Mod}_\phi(\mathbf{Set})$ of T-modules [3, Proposition 13]. The syntax of pure λ-calculus may be specified on $\mathscr{C} = \mathbf{Set}$: a model is a monad T with module morphisms

$$\lambda \colon T^T \to T \qquad\qquad @ \colon T \times T \to T,$$

where $T^T(X) = T(X+1)$. The next example will legitimate the notation in a more general setting.

Example 19. The previous example extends to the simply-typed setting, using the following basic constructions, for any set $\mathbb{I}$, $t \in \mathbb{I}$, and monad T on $\mathbf{Set}^{\mathbb{I}}$.

- The assignment $X \mapsto X(t)$ extends to a T-module which we denote by T_t.
- Given any T-module M, the assignment $X \mapsto M(X + \mathbf{y}_t)$ extends to a T-module M^{T_t} (that we denote M^t), which is indeed an exponential object in $T\text{-}\mathbf{Mod}_\phi$.

We may use this to model the syntax of $\overline{\lambda}\mu$-calculus, taking $\mathbb{I} = \{\mathbf{t}, \mathbf{s}\}$, where $\mathbf{t}$ stands for terms and $\mathbf{s}$ for stacks. E.g., the push operation $e \cdot \pi$ may be specified as module morphism

$$\cdot \colon T_{\mathbf{t}} \times T_{\mathbf{s}} \to T_{\mathbf{s}}.$$

Furthermore, we may specify λ and μ abstraction as module morphisms

$$\lambda \colon T_{\mathbf{t}}^{T_{\mathbf{t}}} \to T_{\mathbf{t}} \qquad\qquad \mu \colon (T_{\mathbf{t}} \times T_{\mathbf{s}})^{T_{\mathbf{s}}} \to T_{\mathbf{t}}$$

(processes being understood as mere pairs of a term and a stack). Indeed, by the above observation, understanding $T_{\mathbf{t}}(X)$ as terms with free term variables in $X(\mathbf{t})$ and stack variables in $X(\mathbf{s})$, $\lambda_X \colon T_{\mathbf{t}}(X + \mathbf{y}_{\mathbf{t}}) \to T_{\mathbf{t}}(X)$ takes a term with one additional term variable and returns a term, as desired. Similarly, $\mu_X \colon T_{\mathbf{t}}(X + \mathbf{y}_{\mathbf{s}}) \times T_{\mathbf{s}}(X + \mathbf{y}_{\mathbf{s}}) \to T_{\mathbf{t}}(X)$ takes a process with one additional stack variable and returns a term.

Notation 6. *If $\vec{t} = (t_1, \dots, t_n)$ is a list of elements of $\mathbb{I}$, we denote $M^{\vec{t}}$ for the T-module $M^{T_{t_1} \times \dots \times T_{t_n}}$ if M is a T-module.*

Let us now return to defining the models of an $\mathbb{I}$-binding signature Σ. By the last example, any binding $\mathbb{I}$-arity B, say

$$\Theta_{u_1}^{\vec{t_1}} \times \dots \times \Theta_{u_m}^{\vec{t_m}} \Rightarrow \Theta_u,$$

induces for any monad T two T-modules,

$$B^-(T) = T_{u_1}^{\vec{t_1}} \times \dots \times T_{u_m}^{\vec{t_m}} \qquad \text{and} \qquad B^+(T) = T_u,$$

which allows us to define:

Definition 22. *A* model *of any $\mathbb{I}$-binding signature $\Sigma = (B_o)_{o \in O}$ is any finitary monad $T \in \mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})$, equipped with an O-indexed family of T-module morphisms $a_o \colon B_o^-(T) \to B_o^+(T)$. Models form a category $\mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})^\Sigma$, by taking as morphisms $(T, (a_o)_{o \in O}) \to (T', (a'_o)_{o \in O})$ the monad morphisms $\alpha \colon T \to T'$ (i.e., natural transformations preserving multiplication and unit) such that for all $o \in O$ the square*

$$\begin{array}{ccc} B_o^-(T) & \xrightarrow{B_o^-(\alpha)} & B_o^-(T') \\ {\scriptstyle a_o}\downarrow & & \downarrow{\scriptstyle a'_o} \\ B_o^+(T) & \xrightarrow[B_o^+(\alpha)]{} & B_o^+(T') \end{array}$$

commutes, where $B_o^-(\alpha)$ and $B_o^+(\alpha)$ are defined in the obvious way.

Lemma 8. *This category $\mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})^\Sigma$ is equivalent to Fiore and Hur's category of Σ-models [11].*

Proof. Across the equivalence (5), restricting attention to monads corresponds to their restriction to monoids, and their compatibility condition [11, page 7] corresponds to our requirement that each a_o be a module morphism. The only subtlety lies in the interpretation of operations. Let us consider the case of a single $\mathbb{I}$-binding arity $B = (b_1 \times \dots \times b_n \Rightarrow \Theta_t)$ for simplicity, where each b_i is $\Theta_{u_i}^{\Theta_{t_1^i} \times \dots \times \Theta_{t_{n_i}^i}}$. They interpret this as requiring an algebra structure on any model X, for the functor $[\mathbb{F}[\mathbb{I}], \mathbf{Set}]^{\mathbb{I}} \to [\mathbb{F}[\mathbb{I}], \mathbf{Set}]^{\mathbb{I}}$ corresponding accross the equivalence with $[\mathbf{Set}_f^{\mathbb{I}}, \mathbf{Set}^{\mathbb{I}}] \to [\mathbf{Set}_f^{\mathbb{I}}, \mathbf{Set}^{\mathbb{I}}]$ to

$$F_B(X)(\gamma) = \Big(\prod_{1 \le i \le n} F_{b_i}(X)(\gamma)\Big) \cdot \mathbf{y}_t,$$

with

$$F_{\Theta_u}^{\Theta_{t_1}\times\ldots\times\Theta_{t_n}}(X)(\gamma) = X(\gamma + \mathbf{y}_{t_1} + \ldots + \mathbf{y}_{t_n})(u).$$

Thus, an F_B-algebra structure on X corresponds to the desired module morphism via

$$\begin{aligned}\int_\gamma [(\textstyle\prod_{1\le i\le n} F_{b_i}(X)(\gamma)) \cdot \mathbf{y}_t, X(\gamma)] &\cong \int_\gamma [\textstyle\prod_{1\le i\le n} F_{b_i}(X)(\gamma), [\mathbf{y}_t, X(\gamma)]] \\ &\cong \int_\gamma [\textstyle\prod_{1\le i\le n} F_{b_i}(X)(\gamma), X(\gamma)(t)] \quad \text{(by Yoneda)} \\ &\cong \int_\gamma [B^-(X)(\gamma), B^+(X)(\gamma)] \\ &\cong [B^-(X), B^+(X)],\end{aligned}$$

slightly abusing notation for the last line.

Now that we have defined the models of Σ, let us turn such $\mathbb{I}$-binding signatures into general signatures for monads. For this, we need to find a notion of arity for monads, or more precisely a functor $\mathscr{D} \to \mathbf{Mnd}_\phi(\mathscr{C})$ whose sections would denote arities for monads. But as we have seen when defining models, a natural notion of arity for a monad T is a T-module. Our functor $\mathscr{D} \to \mathbf{Mnd}_\phi(\mathscr{C})$ should thus have T-$\mathbf{Mod}_\phi(\mathscr{E})$ as its fibre over any T. For this, we resort to the fibration $p_{\mathscr{C},\mathscr{E}} \colon \mathbf{Mod}_\phi(\mathscr{C},\mathscr{E}) \to \mathbf{Mnd}_\phi(\mathscr{C})$ of the total category of modules into monads (Definition 3). Let us detail the construction of this fibration through the Grothendieck construction [26].

Lemma 9. *The assignment $T \mapsto T$-$\mathbf{Mod}_\phi(\mathscr{E})$ extends to a functor*

$$\mathbf{Mod}_\phi(\mathscr{E}) \colon \mathbf{Mnd}_\phi(\mathscr{C})^{op} \to \mathbf{CAT}.$$

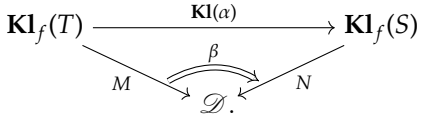
Proof. Any monad morphism $\varphi \colon T \to T'$ yields a functor $\mathbf{Kl}(\varphi)_f \colon \mathbf{Kl}_f(T) \to \mathbf{Kl}_f(T')$ mapping any C to itself, and any $f \colon C \to T(C')$ to the composite

$$C \xrightarrow{f} T(C') \xrightarrow{\varphi_{C'}} T'(C').$$

The action $\mathbf{Mod}_\phi(\mathscr{E})(\varphi) \colon T'$-$\mathbf{Mod}_\phi(\mathscr{E}) \to T$-$\mathbf{Mod}_\phi(\mathscr{E})$ is thus given by mapping any $M \colon \mathbf{Kl}(T')_f \to \mathscr{E}$ to the composite $\mathbf{Kl}(T)_f \xrightarrow{\mathbf{Kl}(\varphi)_f} \mathbf{Kl}(T')_f \xrightarrow{M} \mathscr{D}$.

Definition 23. *Let $\mathbf{Mod}_\phi(\mathscr{C},\mathscr{E})$ denote the total category of $\mathbf{Mod}_\phi(\mathscr{E})$.*

Concretely, an object is a pair (T,M) of a monad and a module over it, and morphisms $(T,M) \to (S,N)$ are pairs of a monad morphism $\alpha \colon T \to S$, together with a natural transformation



By construction, the category $\mathbf{Mod}_\phi(\mathscr{C},\mathscr{E})$ comes with a projection functor $p_{\mathscr{C},\mathscr{E}} \colon \mathbf{Mod}_\phi(\mathscr{C},\mathscr{E}) \to \mathbf{Mnd}_\phi(\mathscr{C})$, which gives the desired arity functor.

Definition 24. *Sections of $p_{\mathscr{C},\mathscr{E}}$ are called* parametric modules.

Example 20. The modules of Examples 18 and 19 all extend to parametric modules, which we respectively denote by Θ, Θ^{Θ}, $\Theta\times\Theta$, Θ_t, and M^{Θ_t}, which justifies the notation of Definition 17 in hindsight.

It remains to recast general binding arities as parametric modules.

Definition 25. *Let $\mathbb{I}$ be a set of simple types. For any $\mathbb{I}$-binding arity $B = (D \Rightarrow \Theta_t)$, let*

$$\overline{B} = (\mathbf{Mod}_\phi(\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}), p_{\mathbf{Set}^{\mathbb{I}},\mathbf{Set}}, D, \Theta_t)$$

denote the induced arity over $\mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})$. Furthermore, for any $\mathbb{I}$-binding signature $\Sigma = (B_o)_{o\in O}$, let $\overline{\Sigma} = (\overline{B_o})_{o\in O}$ denote the induced signature.

Lemma 10. *Models of the general signature $\overline{\Sigma}$ are ismorphic to models of Σ, i.e.,*

$$\mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})^{\overline{\Sigma}} \simeq \mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})^{\Sigma},$$

over $\mathbf{Mnd}_\phi(\mathbf{Set}^{\mathbb{I}})$.

Proof. Concretely, models of $\overline{\Sigma}$ consist of a finitary monad T, equipped with module morphisms $\rho_o \colon D_o(T) \to \Theta_{t_o}(T)$, for all $o \in O$, and morphisms $(T,\rho) \to (U,\sigma)$ are monad morphisms $f \colon T \to U$ making the following square commute for all o.

$$\begin{array}{ccc} D_o(T) & \xrightarrow{\Sigma(f)} & D_o(U) \\ {\scriptstyle\rho}\downarrow & & \downarrow{\scriptstyle\sigma} \\ \Theta_{t_o}(T) & \xrightarrow[f]{} & \Theta_{t_o}(U) \end{array}$$

Clearly, we have $D_o = B_o^-$ and $\Theta_{t_o} = B_o^+$, hence the result.

Corollary 2. *Let $\mathbb{I}$ be a set of simple types and Σ be an $\mathbb{I}$-binding signature. Then, Σ is effective.*

Proof. By [11, §8], the forgetful functor from Σ-models is monadic, so the result follows from Lemmas 8 and 10.

To conclude this section on signatures for monads, let us mention that Fiore and Hur [11] propose an extended notion that incorporates equations, which we do not need in examples.

B Specifying functors

In this section, we prove that any $\mathbb{I},\mathbb{J}$-binding signature Σ with a set E of Σ-equations induce an effective signature over $\mathscr{C} = [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}^{\mathbb{J}}]_\phi$ whose category of models is the fullsubcategory of models M of Σ such that $u_M = v_M$ for any Σ-equation $u = v \colon B$ in E,

The steps consist in first constructing an *equational system* from [12] with the adequate notion of models, and then show that any equational system induces a general signature with the adequate notion of models.

We show that any $\mathbb{I},\mathbb{J}$-binding signature induces an endofunctor $\mathscr{C}$.

Definition 26. *Let* $B = (\underline{\mathrm{Id}}_{v_1} \times \ldots \times \underline{\mathrm{Id}}_{v_p} \times \Theta_{u_1}^{\vec{t_1}} \times \ldots \times \Theta_{u_m}^{\vec{t_m}} \Rightarrow \Theta_u)$ *be a* $\mathbb{I},\mathbb{J}$*-binding arity. We denote* $\mathscr{F}(B)$ *the endofunctor* $B^- \times \mathbf{y}u$ *on* $\mathscr{C}$*, where* $\_ \times \mathbf{y}u$ *is the left adjoint to the functor* $\_{}_u : \mathscr{C} \to [\mathbf{Set}^{\mathbb{I}}, \mathbf{Set}]_\phi$ *taking the* u^{th} *component. Then, an algebra for* $\mathscr{F}(B)$ *is the same thing as a finitary functor* $F : \mathbf{Set}^{\mathbb{I}} \to \mathbf{Set}^{\mathbb{J}}$ *with a natural transformation* $B^-(F) \to B^+(f)$*.*

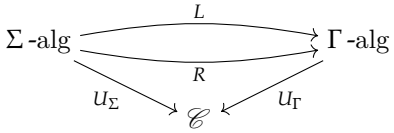
Let Σ *be a* $\mathbb{I},\mathbb{J}$*-binding signature. We denote* $\mathscr{F}(\Sigma)$ *the coproduct of all the endofunctors* $\mathscr{F}(B)$ *for each binding arity* B *in* Σ*.*

Lemma 11. *Let* Σ *be a* $\mathbb{I},\mathbb{J}$*-binding signature. The category of algebras of* $\mathscr{F}(\Sigma)$ *is isomorphic to the category of models of* Σ*.*

Now, given a $\mathbb{I},\mathbb{J}$-binding signature Σ, a Σ-equation $u = v : B$ defines (through the above adjunction) a pair of functors $\mathscr{F}(u)$ and $\mathscr{F}(v)$ from $\mathscr{F}(\Sigma)$-alg to $\mathscr{F}(B)$-alg preserving the underlying object.

This corresponds to the definition of equational system from [12].

Definition 27 ([12, Definition 3.3]). *An* equational system $\mathbb{S} = (\mathscr{C} : \Sigma \rhd \Gamma \vdash L = R)$ *consists of a category* $\mathscr{C}$*, endofunctors* $\Sigma, \Gamma : \mathscr{C} \to \mathscr{C}$*, endofunctors* $\Sigma, \Gamma : \mathscr{C} \to \mathscr{C}$*, and functors* $L, R : \Sigma\text{-alg} \to \Gamma\text{-alg}$ *making the following triangle commute, where* U_Σ *and* U_Γ *are the forgetful functors.*



Definition 28. *Given an equational system* $\mathbb{S} = (\mathscr{C} : \Sigma \rhd \Gamma \vdash L = R)$*, a model for* $\mathbb{S}$*, or an* $\mathbb{S}$-algebra*, is a* Σ*-algebra* $\rho : \Sigma X \to X$ *for which* $L(X,\rho) = R(X,\rho)$*. More generally, let* $\mathbb{S}$-alg*, the category of* $\mathbb{S}$*-algebras, denote the following equaliser in* **CAT***.*

$$\mathbb{S}\text{-alg} \hookrightarrow \Sigma\text{-alg} \underset{R}{\overset{L}{\rightrightarrows}} \Gamma\text{-alg}$$

Given a binding signature Σ and a set E of Σ-equations, we have for each equation $u = v : B \in E$ an equational system induced by $\mathscr{F}(u), \mathscr{F}(v) : \mathscr{F}(\Sigma)\text{-alg} \to \mathscr{F}(B)\text{-alg}$. We gather all of them into a pair of functors $\mathscr{F}(\Sigma)\text{-alg} \to \mathscr{F}(E)\text{-alg}$ where

$$\mathscr{F}(E) \coloneqq \sum_{u=v:B \in E} \mathscr{F}(B).$$

Then, the category of algebras of the induced equational system is isomorphic to the full subcategory of models M of Σ such that $u_M = v_M$, for any Σ-equation $u = v : B \in E$.

We have an effectivity result for these equational systems.

Theorem 7. *The category of algebras of an equational system induced by a* $\mathbb{I},\mathbb{J}$*-binding signature* Σ *and a set* E *of* Σ*-equations as above has an initial object.*

Proof. $\mathscr{C}$ is cocomplete, $\mathscr{F}(\Sigma)$ and $\mathscr{F}(E)$ are easily shown to be finitary. Then, by [12, Theorem 5.1 (1)], the forgetful functor from $\mathbb{S}$-alg to $\mathscr{C}$ has a left adjoint: by cocontinuity, the image of the initial object by this adjoint is initial.

It remains to show that any equational system $\mathbb{S} = (\mathscr{C} : \Sigma \rhd \Gamma \vdash L = R)$ induces a general signature over the category $\mathscr{C}$ with a category of models isomorphic to the category of $\mathbb{S}$-algebras. This signature is of size two. The first family consists of a single arity whose category of models are Σ-algebras (Example 4). The second family consists of consider the following equational arity $(\mathscr{D}, a, u, v)$ over the category of Σ-algebras enforcing the required equation.

– $\mathscr{D}$ is the category of objects $c \in \mathscr{C}$ equipped with an algebra structure for both Σ and Γ. More formally, $\mathscr{D}$ is defined as the pullback

$$\begin{array}{ccc} \mathscr{D} & \longrightarrow & \Gamma\text{-alg} \\ {\scriptstyle a}\downarrow & \lrcorner & \downarrow \\ \Sigma\text{-alg} & \longrightarrow & \mathscr{C} \end{array} .$$

– $a\colon \mathscr{D} \to \Sigma$-alg is induced by the definition of $\mathscr{D}$ as a pullback.
– u maps a Σ-algebra c to the underlying object of c equipped with the same Σ-algebra structure and the Γ-algebra given by $L(c)$.
– v maps a Σ-algebra c to the underlying object of c equipped with the same Σ-algebra structure and the Γ-algebra given by $R(c)$.

C Proof of Lemma 3

Let us first check the monad axioms with unit and bind. For the first unit axiom, we have

$$\widetilde{\eta_C}^\star = \eta_C^\star = id_{TC}, \tag{6}$$

hence

$$U(\widetilde{\eta_C}^\star) = U(id_{TC}) = id_{UTC}$$

as desired.

For the second unit axiom, for any $f\colon C \to UTC'$, we have $\tilde{f} = \tilde{f}^\star \circ \eta_C$ because T is a relative monad, hence $f = U(\tilde{f}^\star) \circ \eta_C^-$ by transposition.

Finally, for the bind axiom, consider any $f\colon C \to UTC'$ and $g\colon C' \to UTC''$. By functoriality of U, it is enough to show

$$\tilde{g}^\star \circ \tilde{f}^\star = (\widetilde{U(\tilde{g}^\star) \circ f})^\star. \tag{7}$$

But because T is a relative monad, we have $\tilde{g}^\star \circ \tilde{f}^\star = (\tilde{g}^\star \circ \tilde{f})^\star$, so it is further enough to show

$$\tilde{g}^\star \circ \tilde{f} = (\widetilde{U(\tilde{g}^\star) \circ f}).$$

Transposing, we have

$$\begin{aligned}(\tilde{g}^\star \circ \tilde{f})^- &= U(\tilde{g}^\star \circ \tilde{f}) \circ \eta^J_C \\ &= U(\tilde{g}^\star) \circ U(\tilde{f}) \circ \eta^J_C \\ &= U(\tilde{g}^\star) \circ f,\end{aligned}$$

as desired.

Finally, let $M(C) = T(C)$ on objects, and $M(C \xrightarrow{f} UTC') = \tilde{f}^\star$. Functoriality boils down to precisely (6) and (7), which we have already shown.

Bibliography

[1] Adámek, J.: Free algebras and automata realizations in the language of categories. Commentationes Mathematicae Universitatis Carolinae **015**(4), 589–602 (1974), http://eudml.org/doc/16649

[2] Ahrens, B.: Modules over relative monads for syntax and semantics. Mathematical Structures in Computer Science **26**, 3–37 (2016)

[3] Ahrens, B., Hirschowitz, A., Lafont, A., Maggesi, M.: High-level signatures and initial semantics. In: Ghica, D.R., Jung, A. (eds.) Proc. 27th EACSL Annual Conference on Computer Science Logic. Leibniz International Proceedings in Informatics (LIPIcs), vol. 119, pp. 4:1–4:22. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2018). https://doi.org/10.4230/LIPIcs.CSL.2018.4, https://doi.org/10.4230/LIPIcs.CSL.2018.4

[4] Ahrens, B., Hirschowitz, A., Lafont, A., Maggesi, M.: Modular specification of monads through higher-order presentations. In: Geuvers, H. (ed.) Proc. 4th International Conference on Formal Structures for Computation and Deduction. Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2019)

[5] Ahrens, B., Hirschowitz, A., Lafont, A., Maggesi, M.: Reduction monads and their signatures (2019), submitted

[6] Ahrens, B., Zsido, J.: Initial semantics for higher-order typed syntax in coq. J. Formalized Reasoning **4**(1), 25–69 (2011). https://doi.org/10.6092/issn.1972-5787/2066, https://doi.org/10.6092/issn.1972-5787/2066

[7] Altenkirch, T., Chapman, J., Uustalu, T.: Monads need not be endofunctors. Logical Methods in Computer Science **11**(1) (2015). https://doi.org/10.2168/LMCS-11(1:3)2015, https://doi.org/10.2168/LMCS-11(1:3)2015

[8] Altenkirch, T., Morris, P., Forsberg, F.N., Setzer, A.: A categorical semantics for inductive-inductive definitions. In: Corradini, A., Klin, B., Cîrstea, C. (eds.) Proc. 4th International Conference on Algebra and Coalgebra in Computer Science LNCS, vol. 6859, pp. 70–84. Springer (2011). https://doi.org/10.1007/978-3-642-22944-2_6, https://doi.org/10.1007/978-3-642-22944-2_6

[9] Bird, R.S., de Moor, O.: Algebra of programming. Prentice Hall International series in computer science, Prentice Hall (1997)

[10] Bloom, B., Istrail, S., Meyer, A.R.: Bisimulation can't be traced. J. ACM **42**(1), 232–268 (1995). https://doi.org/10.1145/200836.200876, https://doi.org/10.1145/200836.200876

[11] Fiore, M.P., Hur, C.K.: Second-order equational logic. In: Proceedings of the 19th EACSL Annual Conference on Computer Science Logic (CSL 2010) (2010)

[12] Fiore, M., Hur, C.K.: On the construction of free algebras for equational systems. Theoretical Computer Science **410**, 1704–1729 (2009)
[13] Fiore, M., Plotkin, G., Turi, D.: Abstract syntax and variable binding. In: Proc. 14th Symposium on Logic in Computer Science IEEE (1999)
[14] Fiore, M.P.: Second-order and dependently-sorted abstract syntax. In: Proc. 23rd Symposium on Logic in Computer Science pp. 57–68. IEEE (2008). https://doi.org/10.1109/LICS.2008.38
[15] Fiore, M.P., Staton, S.: A congruence rule format for name-passing process calculi from mathematical structural operational semantics. In: Proc. 21st Symposium on Logic in Computer Science pp. 49–58. IEEE (2006). https://doi.org/10.1109/LICS.2006.7
[16] Fiore, M.P., Turi, D.: Semantics of name and value passing. In: Proc. 16th Symposium on Logic in Computer Science pp. 93–104. IEEE (2001). https://doi.org/10.1109/LICS.2001.932486
[17] Garner, R.: Combinatorial structure of type dependency. CoRR **abs/1402.6799** (2014), `http://arxiv.org/abs/1402.6799`
[18] Hamana, M.: Term rewriting with variable binding: An initial algebra approach. In: Proc. 5th International Conference on Principles and Practice of Declarative Programming ACM (2003). https://doi.org/10.1145/888251.888266
[19] Herbelin, H.: Séquents qu'on calcule: de l'interprétation du calcul des séquents comme calcul de lambda-termes et comme calcul de stratégies gagnantes. Ph.D. thesis, Paris Diderot University, France (1995), `https://tel.archives-ouvertes.fr/tel-00382528`
[20] Hirschowitz, A., Maggesi, M.: Modules over monads and linearity. In: WoLLIC. LNCS, vol. 4576, pp. 218–237. Springer (2007). https://doi.org/10.1007/3-540-44802-0_3
[21] Hirschowitz, A., Maggesi, M.: Modules over monads and initial semantics. Information and Computation **208**(5), 545–564 (2010). https://doi.org/10.1016/j.ic.2009.07.003
[22] Hirschowitz, T.: Cartesian closed 2-categories and permutation equivalence in higher-order rewriting. Logical Methods in Computer Science **9**(3) (2013). https://doi.org/10.2168/LMCS-9(3:10)2013, `http://hal.archives-ouvertes.fr/hal-00540205`
[23] Hirschowitz, T.: Cellular monads from positive GSOS specifications. In: Pérez, J.A., Rot, J. (eds.) Proc. 26th International Workshop on Expressiveness in Concurrency and 16th Workshop on Structural Operational Semantics, EXPRESS/SOS. EPTCS, vol. 300, pp. 1–18 (2019). https://doi.org/10.4204/EPTCS.300.1, `https://doi.org/10.4204/EPTCS.300.1`
[24] Hirschowitz, T.: Familial monads and structural operational semantics. PACMPL **3**(POPL), 21:1–21:28 (2019), `https://dl.acm.org/citation.cfm?id=3290334`
[25] J.A. Goguen, J.T., Wagner, E.: An initial algebra approach to the specification, correctness and implementation of abstract data types. In: Yeh, R. (ed.) Current Trends in Programming Methodology, IV: Data Structuring. pp. 80–144. Prentice-Hall (1978)

[26] Jacobs, B.: Categorical Logic and Type Theory. No. 141 in Studies in Logic and the Foundations of Mathematics, North Holland, Amsterdam (1999)
[27] Kelly, G.M.: Elementary observations on 2-categorical limits. Bulletin of the Australian Mathematical Society **39**, 301–317 (1989)
[28] Lafont, A.: Signatures and models for syntax and operational semantics in the presence of variable binding. Ph.D. thesis, École Nationale Superieure Mines – Telecom Atlantique Bretagne Pays de la Loire – IMT Atlantique (2019), `https://arxiv.org/abs/1910.09162v2`, forthcoming
[29] Pitts, A.M.: Howe's Method for Higher-Order Languages, Cambridge Tracts in Theoretical Computer Science, vol. 52, chap. 5. Cambridge University Press (2011)
[30] Sangiorgi, D., Walker, D.: The π-calculus - a theory of mobile processes. Cambridge University Press (2001)
[31] Staton, S.: General structural operational semantics through categorical logic. In: Proc. 23rd Symposium on Logic in Computer Science pp. 166–177. IEEE (2008). https://doi.org/10.1109/LICS.2008.43
[32] Street, R.: The formal theory of monads. Journal of Pure and Applied Algebra **2**, 149–168 (1972)
[33] Turi, D., Plotkin, G.D.: Towards a mathematical operational semantics. In: Proc. 12th Symposium on Logic in Computer Science pp. 280–291 (1997). https://doi.org/10.1109/LICS.1997.614955