Variable binding and substitution for (nameless) dummies

ANDRÉ HIRSCHOWITZ, Univ. Côte d'Azur, CNRS, LJAD, 06103, France
TOM HIRSCHOWITZ, Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, 73000, France
AMBROISE LAFONT, University of New South Wales, Australia
MARCO MAGGESI, Università degli Studi di Firenze, Italy

By abstracting over well-known properties of De Bruijn's representation with nameless dummies, we design a new theory of syntax with variable binding and capture-avoiding substitution. We propose it as a simpler alternative to Fiore, Plotkin, and Turi's approach, with which we establish a strong formal link. We also show that our theory easily incorporates simple types and equations between terms.

Additional Key Words and Phrases: syntax ; variable binding ; substitution ; category theory

1 INTRODUCTION

There is a standard notion of signature for syntax with variable binding called *binding signature*. Such a signature consists of a set of operation symbols, together with, for each of them, a *binding arity*. A binding arity is a list $(n_1, \dots, n_p)$ of natural numbers, whose meaning is that the considered operation has p arguments, with n_i variables bound in the ith argument, for all $i \in \{1, \dots, p\}$.

Example 1.1.

- λ-abstraction has binding arity (1) (one argument, with one bound variable);
- application has binding arity (0, 0) (two arguments, with no bound variables);
- unary explicit substitution $e[x \mapsto f]$ has binding arity (1, 0) (two arguments, with one variable bound in the first and none in the second).

Among the many possible representations of the syntax specified by a binding signature S, we focus on De Bruijn's encoding with nameless dummies [De Bruijn 1972]. Let us stress some of its features.

Inductive definition The set DB_S of terms is the least fixed point of a suitable endofunctor on sets, derived from S. In particular, there is a **variables** map $v\colon \mathbb{N} \to \mathrm{DB}_S$ and, for each operation o in S with binding arity $(n_1, \dots, n_p)$, a map $o_{\mathrm{DB}_S}\colon \mathrm{DB}_S^p \to \mathrm{DB}_S$.

Substitution DB_S is equipped with a **(parallel) substitution** map

$$-[-]\colon \mathrm{DB}_S \times \mathrm{DB}_S^{\mathbb{N}} \to \mathrm{DB}_S,$$

which satisfies three standard substitution lemmas (**associativity**, **left** and **right unitality**). Furthermore, substitution is compatible with operations, in the sense that it satisfies the following crucial **binding conditions**: for each operation o with binding arity $(n_1, \dots, n_p)$, $e_1, \dots, e_p \in \mathrm{DB}_S$, and $f\colon \mathbb{N} \to \mathrm{DB}_S$,

$$o_{\mathrm{DB}_S}(e_1, \dots, e_p)[f] = o_{\mathrm{DB}_S}(e_1[\Uparrow^{n_1} f], \dots, e_p[\Uparrow^{n_p} f]), \tag{1}$$

Authors' addresses: André Hirschowitz, Univ. Côte d'Azur, CNRS, LJAD, 06103, Nice, France; Tom Hirschowitz, Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, 73000, Chambéry, France; Ambroise Lafont, University of New South Wales, Sydney, Australia; Marco Maggesi, Università degli Studi di Firenze, Florence, Italy.

where $\Uparrow$ is a unary operation defined on $\mathrm{DB}_S^{\mathbb{N}}$ by

$$\begin{aligned}(\Uparrow \sigma)(0) &= v(0)\\ (\Uparrow \sigma)(n+1) &= \sigma(n)[p \mapsto v(p+1)].\end{aligned}$$

In the present work, by abstracting over these properties, we propose a simple theory for syntax with variable binding, which we summarise as follows.

De Bruijn monad A De Bruijn monad consists of a set X, equipped with **variables** and **substitution** maps, say $v\colon \mathbb{N} \to X$ and $-[-]\colon X \times X^{\mathbb{N}} \to X$, satisfying the abstract counterparts of the above substitution lemmas.

De Bruijn S-algebra A De Bruijn S-algebra[1] is a De Bruijn monad $(X, -[-], v)$ equipped with operations from the signature S, satisfying the abstract counterpart of the above binding condition.

The term De Bruijn S-algebra We define the set DB_S by an abstract counterpart of the above inductive definition. The substitution map $-[-]\colon \mathrm{DB}_S \times \mathrm{DB}_S^{\mathbb{N}} \to \mathrm{DB}_S$ is then the unique map satisfying left unitality and the binding conditions. Furthermore, it satisfies both other substitution lemmas, hence upgrades DB_S into a De Bruijn S-algebra.

Category of De Bruijn S-algebras De Bruijn S-algebras are the objects of a category S - **DBAlg**, whose morphisms are all maps between underlying sets that commute with variables, substitution, and operations.

Initial-algebra Semantics Finally, DB_S is initial in S - **DBAlg**, which provides a relevant induction/recursion principle.

We thus propose a theory for syntax with substitution, which is an alternative to the mainstream initial-algebra semantics of [Fiore et al. 1999]. We have experienced the simplicity of our theory by formalising it not only in Coq, but also in HOL Light, which does not support dependent types.

Our theory is similar to the mainstream theory [Fiore et al. 1999], in the following aspects.

- Our and their basic definitions of syntax can be recast using relative monads: De Bruijn monads are monads relative to the functor $1 \to$ **Set** selecting $\mathbb{N}$, while their substitution monoids are monads relative to the full and faithful embedding into **Set** of the category of finite ordinals and arbitrary maps between them.
- We find (Theorem 5.22) that both approaches, in their own ways, include exotic models, and that when freed from them, our category of De Bruijn S-algebras and their category of S-models become equivalent. In this sense, both semantics differ only marginally.
- In §6, we show how De Bruijn S-algebras can be defined by resorting to (a slight generalisation [Borthelle et al. 2020] of) *pointed strong endofunctors*, in the spirit of [Fiore et al. 1999].
- Their framework accomodates simple types and equations [Fiore and Hur 2010]; we also provide such extensions of our theory in §8 and §9.

Related work

Abstract frameworks for variable binding. One of the mainstream such frameworks is [Fiore et al. 1999]. This has been our main reference and in §6 we establish a strong link between this framework and our proposal. This link could probably be extended to variants such as [Arkor and McDermott 2021; Hirschowitz and Maggesi 2007, 2010].

In a more recent work, Allais et al. [Allais et al. 2018] introduce a universe of syntaxes, which widely generalises binding signatures. Their framework is designed to facilitate the definition of

[1] There is a slightly different notion of De Bruijn algebra in the literature, see the related work section.

so-called **traversals**, i.e., functions defined by structural induction, "traversing" their argument. We leave for future work the task of adapting our approach to such traversals.

In a similar spirit, let us mention the recent work of Gheri and Popescu [Gheri and Popescu 2020], which presents a theory of syntax with binding, mechanised in Isabelle/HOL. Potential links with our approach remain unclear to us at the time of writing.

Finally, the nominal sets approach [Gabbay and Pitts 1999] was extended by Power [Power 2007] to incorporate built-in substitution. As for Fiore, Plotkin, and Turi's framework, our work is very close in spirit: we consider that our treatment is more elementary.

Proof assistant libraries. De Bruijn representation benefits from well-developed proof assistant libraries, in particular Autosubst [Schäfer et al. 2015b; Stark et al. 2019]. They introduce a notion of De Bruijn algebra, and design a sound and complete decision procedure for their equational theory, which they furthermore implement for Coq.

Our notion of De Bruijn algebra differs from theirs, notably in that their substitutions are finitely generated. Our approach makes the theoretical development significantly simpler, but of course finite generation is crucial for their main purpose, namely decidability.

General notation

We denote by $A^* = \sum_{n\in\mathbb{N}} A^n$ the set of finite sequences of elements of A, for any set A. In any category $\mathbf{C}$, we tend to write $[C, D]$ for the hom-set $\mathbf{C}(C, D)$ between any two objects C and D. Finally, let F - **alg** denote the usual category of F-algebras and morphisms between them.

2 DE BRUIJN MONADS

In this section, we start by introducing De Bruijn monads. Then, we define lifting of assignments, the binding conditions, and the models of a binding signature S in De Bruijn monads, De Bruijn S-algebras. Finally, we construct the term De Bruijn S-algebra.

2.1 Definition of De Bruijn monads

We start by fixing some terminology and notation, and then give the definition.

Definition 2.1. Given a set X, an X**-assignment** is a map $\mathbb{N} \to X$. We sometimes merely use "assignment" when X is clear from context.

Notation 1. *Consider any map $s\colon X \times Y^{\mathbb{N}} \to Z$.*

- *For all $x \in X$ and $g\colon \mathbb{N} \to Y$, we write $x[g]_s$ for $s(x, g)$, or even $x[g]$ when s is clear from context.*
- *Furthermore, s gives rise to the map*

$$\begin{array}{rcl} X^{\mathbb{N}} \times Y^{\mathbb{N}} & \to & Z^{\mathbb{N}} \\ (f, g) & \mapsto & n \mapsto s(f(n), g). \end{array}$$

We use similar notation for this map, i.e., $f[g](n) := f(n)[g]_s$.

Definition 2.2. A **De Bruijn monad** is a set X, equipped with

- a **substitution** map $s\colon X \times X^{\mathbb{N}} \to X$, which takes an element $x \in X$ and an assignment $f\colon \mathbb{N} \to X$, and returns an element $x[f]$, and
- a **variables** map $v\colon \mathbb{N} \to X$,

satisfying, for all $x \in X$, and $f, g\colon \mathbb{N} \to X$:

- **associativity**: $x[f][g] = x[f[g]]$,
- **left unitality**: $v(n)[f] = f(n)$, and
- **right unitality**: $x[v] = x$.

Example 2.3. The set $\mathbb{N}$ itself is clearly a De Bruijn monad, with variables given by the identity and substitution $\mathbb{N} \times \mathbb{N}^{\mathbb{N}} \to \mathbb{N}$ given by evaluation. This is in fact the initial De Bruijn monad, as should be clear from the development below.

Example 2.4. The set $\Lambda := \mu X.\mathbb{N} + X + X^2$ of λ-terms forms a De Bruijn monad. The variables map $\mathbb{N} \to \Lambda$ is the obvious one, while the substitution map $\Lambda \times \Lambda^{\mathbb{N}} \to \Lambda$ is less obvious but standard. In Example 2.16, as an application of Theorem 3.6, we will characterise this De Bruijn monad by a universal property.

2.2 Lifting assignments

Given a De Bruijn monad M, we define an operation called **lifting** on its set of assignments $\mathbb{N} \to M$. It is convenient to stress that only part of the structure of De Bruijn monad is needed for this definition.

Definition 2.5. Consider any set M, together with a **lift** map $\ell \colon M \to M$, and a **first variable** $z \in M$. For any $\sigma \colon \mathbb{N} \to M$, we define the assignment $\Uparrow \sigma := \delta_{\ell,z}\sigma \colon \mathbb{N} \to M$ by

$$\begin{aligned} \Uparrow \sigma(0) &= z \\ \Uparrow \sigma(n+1) &= \ell(\sigma(n)). \end{aligned}$$

In particular, for any set M equipped with maps $s \colon M \times M^{\mathbb{N}} \to M$ and $v \colon \mathbb{N} \to M$, we take by default $z = v(0)$ and $\ell(x) = x[\uparrow]_s$, where $\uparrow \colon \mathbb{N} \to X$ maps any n to $v(n+1)$.

Of course we may iterate lifting:

Definition 2.6. Let $\Uparrow^0 A = A$, and $\Uparrow^{n+1} A = \Uparrow (\Uparrow^n A)$.

2.3 Binding arities and binding conditions

Our treatment of binding arities reflects the separation between the first-order part of the arity, namely its length, which concerns the syntax, and the binding information, namely the binding numbers, which concerns the compatibility with substitution.

Definition 2.7.

- A **first-order arity** is a natural number.
- A **binding arity** is a sequence $(n_1, \dots, n_p)$ of natural numbers, i.e., an element of $\mathbb{N}^*$.
- The first-order arity $|a|$ associated with a binding arity $a = (n_1, \dots, n_p)$ is its length p.

Let us now axiomatise what we call an operation of a given binding arity.

Definition 2.8. Let $a = (n_1, \dots, n_p)$ be any binding arity, M be any set, $s \colon M \times M^{\mathbb{N}} \to M$, and $v \colon \mathbb{N} \to M$. An operation **of binding arity** a is a map $o \colon M^p \to M$ satisfying the following a-**binding condition** w.r.t. (s, v):

$$\forall \sigma \colon \mathbb{N} \to M, x_1, \dots, x_p \in M, \quad o(x_1, \dots, x_p)[\sigma] = o(x_1[\Uparrow^{n_1} \sigma], \dots, x_p[\Uparrow^{n_p} \sigma]). \tag{2}$$

Remark 1. *Let us emphasise the dependency of this definition on v and s – which is hidden in the notations for substitution and lifting.*

2.4 Binding signatures and algebras

In this section, we recall the standard notions of first-order (resp. binding) signatures, and adapt the definition of algebras to our De Bruijn context. Let us first briefly recall the former.

Definition 2.9. A **first-order signature** consists of a set O of **operations**, equipped with an **arity** map $ar \colon O \to \mathbb{N}$.

Definition 2.10. For any first-order signature $S := (O, ar)$, an S-**algebra** is a set X, together with, for each operation $o \in O$, a map $o_X : X^{ar(o)} \to X$.

Let us now generalise this to binding signatures.

Definition 2.11.

- A **binding signature** [Plotkin 1990] consists of a set O of **operations**, equipped with an **arity** map $ar : O \to \mathbb{N}^*$. Intuitively, the arity of an operation specifies the number of bound variables in each argument.
- The first-order signature $|S|$ associated with a binding signature $S := (O, ar)$ is $|S| := (O, |ar|)$, where $|ar| : O \to \mathbb{N}$ maps any $o \in O$ to $|ar(o)|$.

Example 2.12. The binding signature for λ-calculus has two operations lam and app, of respective arities (1) and (0, 0). The associated first-order signature has two operations lam and app, of respective arities 1 and 2.

Let us now present the notion of De Bruijn S-algebra:

Definition 2.13. For any binding signature $S := (O, ar)$, a **De Bruijn** S-**algebra** is a De Bruijn monad (X, s, v) equipped with an operation o_X of binding arity $ar(o)$, for all $o \in O$.

In order to state our characterisation of the term model, we associate to any binding signature an endofunctor on sets, as follows.

Definition 2.14. The endofunctor Σ_S associated to a binding signature (O, ar) is defined by $\Sigma_S(X) = \sum_{o \in O} X^{|ar(o)|}$.

Remark 2. *The induced endofunctor just depends on the underlying first-order signature.*

Remark 3. *As is well known, for any binding signature, the initial $(\mathbb{N} + \Sigma_S)$-algebra has as carrier the least fixed point $\mu A.\mathbb{N} + \Sigma_S(A)$.*

The following theorem defines the term model of a binding signature.

Theorem 2.15. *Consider any binding signature $S = (O, ar)$, and let DB_S denote the initial $(\mathbb{N} + \Sigma_S)$-algebra, with structure maps $v : \mathbb{N} \to \mathrm{DB}_S$ and $a : \Sigma_S(\mathrm{DB}_S) \to \mathrm{DB}_S$. Then,*

(i) *There exists a unique map $s : \mathrm{DB}_S \times \mathrm{DB}_S{}^{\mathbb{N}} \to \mathrm{DB}_S$ such that*
   - *for all $n \in \mathbb{N}$ and $f : \mathbb{N} \to \mathrm{DB}_S$, $s(v(n), f) = f(n)$, and*
   - *for all $o \in O$, the map o_{DB_S} satisfies the $ar(o)$-binding condition w.r.t. (s, v).*

(ii) *This map turns (DB_S, v, s, a) into a De Bruijn S-algebra.*

Proof. We have proved the result in both HOL Light [Maggesi 2021] and Coq [Lafont 2021]. □

Remark 4. *Point* (i) *may be viewed as an abstract form of recursive definition for substitution in the term model. The theorem thus allows us to construct the term model of a signature in two steps: first the underlying set, constructed as the inductive datatype $\mu Z.\mathbb{N} + \Sigma_S(Z)$, and then substitution, defined by the binding conditions viewed as recursive equations.*

Example 2.16. For the binding signature of λ-calculus (Example 2.12), the carrier of the initial model is $\mu Z.\mathbb{N} + Z + Z^2$, and substitution is defined inductively as usual:

$$\begin{aligned} v(n)[\sigma] &= \sigma(n) \\ \lambda(e)[\sigma] &= \lambda(e[\Uparrow \sigma]) \\ (e_1\ e_2)[\sigma] &= e_1[\sigma]\ e_2[\sigma]. \end{aligned}$$

3 INITIAL-ALGEBRA SEMANTICS OF BINDING SIGNATURES IN DE BRUIJN MONADS

In this section, for any binding signature S, we organise De Bruijn S-algebras into a category, S - **DBAlg**, and prove that the term De Bruijn S-algebra is initial therein.

3.1 A category of De Bruijn monads

Let us start by organising general De Bruijn monads into a category:

Definition 3.1. A morphism $(X, s, v) \to (Y, t, w)$ between De Bruijn monads is a set-map $f : X \to Y$ commuting with substitution and variables, in the sense that for all $x \in X$ and $g : \mathbb{N} \to X$ we have $f(x[g]) = f(x)[f \circ g]$ and $f \circ v = w$.

Remark 5. *More explicitly, the first axiom says:* $f(s(x, g)) = t(f(x), f \circ g)$.

Notation 2. *De Bruijn monads and morphisms between them form a category, which we denote by* **DBMnd**.

3.2 De Bruijn monads as relative monads and as monoids

In this subsection, we briefly mention a categorical point of view on the category of De Bruijn monads for the categorically-minded reader, in terms of **relative** monads [Altenkirch et al. 2015].

Proposition 3.2. *The category* **DBMnd** *is canonically isomorphic to the category of monads relative to the functor* $1 \to \mathbf{Set}$ *picking* $\mathbb{N}$.

Remark 6. *Canonicity here means that the isomorphism lies over the canonical isomorphism* $[1, \mathbf{Set}] \cong \mathbf{Set}$.

According to the theory of [Altenkirch et al. 2015], this yields:

Corollary 3.3. *The tensor product* $X \otimes Y := X \times Y^{\mathbb{N}}$ *induces a skew monoidal [Szlachányi 2012] structure on* **Set**, *and* **DBMnd** *is precisely the category of monoids therein.*

Proof. To see this, let us observe that, by viewing any set X, in particular $\mathbb{N}$, as a functor $1 \to \mathbf{Set}$, one may compute the left Kan extension of X along $\mathbb{N}$, which is a functor $\mathrm{Lan}_{\mathbb{N}}(X) : \mathbf{Set} \to \mathbf{Set}$. By the standard formula for left Kan extensions [Mac Lane 1998], we have $\mathrm{Lan}_{\mathbb{N}}(X)(Y) \cong X \times Y^{\mathbb{N}} = X \otimes Y$. □

3.3 Categories of De Bruijn algebras

In this section, for any binding signature S, we organise De Bruijn S-algebras into a category S - **DBAlg**.

Let us start by recalling the category of S-algebras for a first-order S:

Definition 3.4. For any first-order signature S, a morphism $X \to Y$ of S-algebras is a map between underlying sets commuting with operations, in the sense that for each $o \in O$, letting $p := ar(o)$, we have $f(o_X(x_1, \ldots, x_p)) = o_Y(f(x_1), \ldots, f(x_p))$.

We denote by S - **alg** the category of S-algebras and morphisms between them.

We now exploit this to define De Bruijn S-algebras:

Definition 3.5. For any binding signature S, a morphism of De Bruijn S-algebras is a map $f : X \to Y$ between underlying sets, which is a morphism both of De Bruijn monads and of $|S|$-algebras. We denote by S - **DBAlg** the category of De Bruijn S-algebras and morphisms between them.

Theorem 3.6. *Consider any binding signature $S = (O, ar)$, and let DB_S denote the initial $(\mathbb{N} + \Sigma_S)$-algebra. Then, the De Bruijn S-algebra structure of Theorem 2.15 on DB_S makes it initial in S - **DBAlg**.*

Proof. Again, see §4. □

4 MECHANISED PROOFS

Our theoretical framework is meant to help specify and reason mechanically about binding syntax using De Bruijn representation. To give practical examples, we describe in this section two implementations, in Coq and HOL Light, that cover several crucial parts of the theory of this paper, and, in particular Theorems 2.15 and 3.6. The full source code is available on github [Lafont 2021; Maggesi 2021].

4.1 HOL Light proof

Let us start by discussing the HOL Light formalization. One key fact is that, despite being much weaker than ZFC set theory or the Calculus of Inductive Constructions, HOL is expressive enough to program the De Bruijn encoding and to reason about it. For instance, other representations, such as those based on monads over sets [Hirschowitz and Maggesi 2010] (possibly via the nested datatype technique [Bird and Paterson 1999; Hirschowitz and Maggesi 2012]) are not directly implementable in HOL.

The reader does not need to be familiar with higher-order logic (HOL) to follow the essential ideas of this section. HOL is based on simply-typed λ-calculus. Following a standard denotational semantics in Zermelo-Fraenkel set theory, types in HOL can be thought of as non-empty sets.

Our HOL Light implementation is divided into two main parts. The first part treats the specific case of λ-calculus and can be useful to illustrate the essential ideas of this paper in a simple –yet paradigmatic– setting. In the following, identifiers in `teletype` font in parenthesis indicate names of the associated theorems in the code. HOL terms and formulas are enclosed in backquotes as in `` `2 + 2` ``, types are prefixed by a colon as in `` `:bool` ``.

The type `:dblambda` of λ-calculus is simply defined as the following inductive type

```
let dblambda_INDUCT, dblambda_RECURSION = define_type
  "dblambda = REF num | APP dblambda dblambda | ABS dblambda";;
```

while the HOL Light definition of substitution fits into the following five lines of code.

```
|- (!f i. SUBST f (REF i) = f i) /\
   (!f x y. SUBST f (APP x y) = APP (SUBST f x) (SUBST f y)) /\
   (!f x. SUBST f (ABS x) = ABS (SUBST (LIFT f) x)) /\
   (!f. LIFT f 0 = REF 0) /\
   (!f i. LIFT f (SUC i) = SUBST (REF o SUC) (f i))
```

The names `` `SUC` `` and `` `o` `` respectively denote successor and function composition. The symbols `` `/\` `` and `` `!` `` are HOL notations for conjunction and universal quantification. We recognise:

- in the first line, the variables map,
- in the next two lines, the binding conditions for application and abstraction, and
- in the final two lines, the recursive equations defining the lifting of an assignment.

This definition formalises the first point of Theorem 2.15 for the case of λ-calculus. More precisely, the first equation corresponds to the first item and the second and third equations are the two binding conditions for our signature and correspond to the second item.

The second point of the Theorem translates in the laws of associativity (`SUBST_SUBST`) and two-sided unit (`SUBST_REF` and the first equation of the above definition).

We also provide the classical definition of unary substitution (SUBST1, as found e.g., in Huet [1994]) and then show how the latter is an instance of the former (SUBST1_EQ_SUBST). As shown by other authors [Abadi et al. 1990; Schäfer et al. 2015a], reasoning on parallel substitution can be significantly easier. Here for instance, we prove the associativity of unary substitution in a few lines (SUBST1_SUBST1) by reducing to parallel substitution. However, proving the same result directly for unary substitution is less intuitive: the mere statement of the property to be proved by induction is tricky to devise.

Next, we introduce the category of De Bruijn monads (MONAD, MONAD_MOR) and their associated modules (MODULE, MODULE_MOR). Our Definition 2.2 presents De Bruijn monads using a set-theoretic style, hence as a triple constituted by a set, an associative substitution operator and a two-side identity. In HOL, this is translated into a type `:A` together with a substitution operation `op:(num->A)->A->A` and a unit `e:num->A`.[2] However, the unit is uniquely determined by the substitution operator and it is denoted `UNIT op` in our implementation. Moreover, the type `:A` is automatically inferred. Thus, we simply identify a De Bruijn monad by its substitution operator, that is, we write `op IN MONAD` to indicate that `op` is (the substitution operator of) a monad

```
|- !op. op IN MONAD <=>
        (!f g x:A. op g (op f x) = op (op g o f) x) /\
        (!f n. op f (UNIT op n) = f n) /\
        (!x. op (UNIT op) x = x)
```

The set of morphisms between two De Bruijn monads `op1` and `op2` is then defined as follows

```
|- MONAD_MOR (op1,op2) =
   {h:A->B | op1 IN MONAD /\ op2 IN MONAD /\
             (!n. h (UNIT op1 n) = UNIT op2 n) /\
             (!f x. h (op1 f x) = op2 (h o f) (h x))}
```

Modules are implemented using a similar style. We implemented the constructions on modules needed for interpreting binding signatures: product (MPROD) and derivation (DMOP).

In this setup, we can state and prove Theorem 3.6 for the λ-calculus. The models of our syntax are De Bruin monads endowed with functions `app` and `abs` that are module morphisms (DBLAMBDA_MODEL)

```
app IN MODULE_MOR op (MPROD op op, op)
lam IN MODULE_MOR op (DMOP op op, op)
```

Model morphisms (DBLAMBDA_MODEL_MOR) are morphisms of De Bruijn monads that commute with `app` and `abs`. We then have the universal property of λ-calculus by giving an initial model morphism (DBLAMBDAINIT, DBLAMBDAINIT_IN_DBLAMBDA_MODEL_MOR)

```
|- !op app lam.
      (op,app,lam) IN DBLAMBDA_MODEL
      ==> DBLAMBDAINIT (op,app,lam) IN
          DBLAMBDA_MODEL_MOR ((SUBST,UNCURRY APP,ABS),(op,app,lam))
```

and by proving its uniqueness (DBLAMBDAINIT_UNIQUE). This part closes with the analogous theorem for the initial semantics of λ-calculus modulo $\beta\eta$ equivalence (EXP_MONAD_MOR_LC_EXPMAP, LC_EXPMAP_UNIQUE). The style is the one proposed in [Hirschowitz and Maggesi 2010] which uses exponential monads, that is, monads endowed with a module isomorphism (that gives *abstraction* of the monad) with their derivation.

The second part of the HOL Light code implements Theorems 2.15 and 3.6 for arbitrary signatures.

[2]We warn the reader that in this code op is used for substitution operation and has not be confused with the operations of the syntax $o \in O$ of the previous sections.

The increased generality comes at a cost in this implementation: since HOL does not feature dependent types; it is impossible to implement the term algebra of a given binding signature as a mere type: one has to resort to a "well-formedness" predicate. From this perspective, it may be instructive to compare with our Coq implementation, which takes advantage of dependent types.

The above difficulty is solved in the standard way in HOL Light. First, we build a type `rterm` for *raw terms* over a "full" signature, i.e., one with countably many operations of each arity. We then introduce an inductive set (i.e., an inductive predicate) of well-formed terms (`WELLFORMED_RULES`) that selects the terms respecting a given signature. Besides this technical difficulty, the formal development follows the same pattern as for λ-calculus.

The substitution operator is specified by two equations (`TMSUBST_CALUSES`)

```
|- (!f i. TMSUBST f (TMREF i) = f i) /\
   (!f c args. TMSUBST f (FN c args) =
               FN c (MAP (\(k,x). k,TMSUBST (TMDERIV k f) x) args))
```

where `` `TMREF` `` denotes variables and `` `FN` `` denotes operations from the signature. The latter takes two arguments, the *name* of the construction `` `c` `` (a natural number) and a list of pairs `` `(k,x)' `` where `` `k` `` is a natural number denoting the number of bound variables and `` `x` `` is a λ-term.

We formulate the appropriate notion of category of models in this setting (`MODEL`, `MODEL_MOR`) of which the above data constitutes an object (`RTERM_IN_MODEL`). Then we prove the universal property by giving the initial morphism (`INITIAL_MORPHISM_IN_MODEL_MOR`) and show its uniqueness (`INITIAL_MORPHISM_UNIQUE`).

We stress that our framework allows us to relate and reason on different syntaxes simultaneously and, for instance, prove theorems on a family of syntaxes. To give an example, we start by considering the trivial category structure on signatures given by inclusion. Then, we prove the functoriality (i.e., monotonicity) of our constructions (`MODEL_MONO`, `MODEL_MOR_MONO`), and we show an easy but significant modularity result (`INITIAL_MORPHISM_MODULARITY`).

Finally, to tie the knot, we derive again the universal property for λ-calculus as an instance of this new, more general, framework (`DBLAMBDA_UNIVERSAL`).

4.2 Coq proof

We now discuss the Coq formalisation. Our implementation addresses only the general case of arbitrary signatures since, as said before, we do not have significant advantages in treating a particular case separately thanks to the use of dependent types.

The reader who wants to skim through the main definitions and constructions of this implementation can look at file `Summary.v` that contains a sequence of checks of the *interface* based on the Coq mechanism of type inference and normalization.

The formalisation has an idiomatic style: the minute details of implementation pose no significant problem. Therefore, we just point the reader to the most relevant parts.

File `syntaxdb.v` starts with the notion of binding condition (`binding_condition`) and the definition of the category of models associated to a binding signatures `S` (`model`, `model_mor`).

Next, we introduce the algebra of terms `Z`, together with the substitution operator `Z_subst` and verify its properties.

Then we define the initial morphism `ini_mor` and prove its uniqueness. All this, again, constitutes a full formal counterpart of Theorem 3.6.

This part is free from axioms. In particular, we avoid the axiom of functional extensionality, stating that pointwise equal functions are equal. Instead, we require that substitution in a De Bruijn monad X is compatible with pointwise equality: if $f, g : \mathbb{N} \to X$ are pointwise equal, then $x[f] = x[g]$ for any $x \in X$.

The other main file is `quotsyntax.v`. There, the previous constructions are upgraded to the context of syntax with equation (`equational_theory`). Here we use an axiomatic theory of quotients (developed in file `Quot.v`) to build the initial model `ZE`, and to define the initial morphism `int_morE`.

In the same file, we also provide the instatiation on the equational signature `LC`$\beta\eta$`_sig` of λ-calculus modulo β and η equations.

Overall, about one-third of the formalisation is devoted to the definitions and proofs up to Theorems 2.15 and 3.6. The rest focus on initiality for signatures with equations (Theorem 9.6), in the untyped setting.

5 RELATION TO PRESHEAF-BASED MODELS

The classical initial-algebra semantics introduced in Fiore et al. [1999] associates in particular to each binding signature S a category, say Φ_S - **Mon** of models, while we have proposed in §3 an alternative category of models S - **DBAlg**. In this section, we are interested in comparing both categories of models.

In fact, we find that both include exotic models, in the sense that we do not see any loss in ruling them out. And when we do so, we obtain equivalent categories.

5.1 Trimming down presheaf-based models

First of all, in this subsection, let us recall the mainstream approach we want to relate to, and exclude some exotic objects from it.

5.1.1 Presheaf-based models. We start by recalling the presheaf-based approach. The ambient category is the category of functors $[\mathbb{F}, \mathbf{Set}]$, where $\mathbb{F}$ denotes the category of finite ordinals, and all maps between them.

Definition 5.1. Let $[\mathbf{Set}, \mathbf{Set}]_f$ denote the full subcategory of $[\mathbf{Set}, \mathbf{Set}]_f$ spanning **finitary** functors, i.e., those preserving directed colimits (= colimits of directed posets).

Proposition 5.2. *The restriction functor* $[\mathbf{Set}, \mathbf{Set}]_f \to [\mathbb{F}, \mathbf{Set}]$ *is an equivalence.*

Proof. By Adámek and Rosicky [1994, Theorem 2.26(ii)] and Adámek and Rosicky [1994, Remark 2.26(1)], with $\mathscr{K} = \mathbf{Set}$, observing that finitely presentable objects of sets are equivalent to $\mathbb{F}$, so we may take $\mathscr{A} = \mathbb{F}$. □

Definition 5.3. Let $(\otimes, I)$ denote the monoidal structure on $[\mathbb{F}, \mathbf{Set}]$ inherited from the composition monoidal structure on $[\mathbf{Set}, \mathbf{Set}]_f$ through the equivalence of Proposition 5.2.

By construction, monoids in $[\mathbb{F}, \mathbf{Set}]$ are thus equivalent to finitary monads on sets.

The idea is then to interpret binding signatures S as endofunctors Φ_S on $[\mathbb{F}, \mathbf{Set}]$, and to define models as monoids equipped with Φ_S-algebra structure and satisfying a suitable compatibility condition.

The definition of Φ_S relies on an operation called derivation:

Definition 5.4 (Endofunctor associated to a binding signature).

- Let the **derivative** X' of any functor $X \colon \mathbb{F} \to \mathbf{Set}$ be defined by $X'(n) = X(n+1)$.
- Furthermore, let $X^{(0)} = X$, and $X^{(n+1)} = (X^{(n)})'$.
- For any binding arity $a = (n_1, \ldots, n_p)$, let $\Phi_a(X) = X^{(n_1)} \times \ldots \times X^{(n_p)}$.
- For any binding signature $S = (O, ar)$, let $\Phi_S = \sum_{o \in O} \Phi_{ar(o)}$.

Proposition 5.5. *Through the equivalence with finitary functors, derivation becomes* $F'(A) = F(A+1)$*, for any finitary* $F \colon \mathbf{Set} \to \mathbf{Set}$ *and* $A \in \mathbf{Set}$.

Example 5.6. For the binding signature S_λ of Example 2.12 for λ-calculus we get $\Phi_{S_\lambda}(X)(n) = X(n)^2 + X(n+1)$.

Next, we want to express the relevant compatibility condition between algebra and monoid structure. For this, let us briefly recall the notion of pointed strength, see Fiore [2008]; Fiore et al. [1999] for details.

Definition 5.7. A **pointed strength** on an endofunctor $F\colon \mathbf{C} \to \mathbf{C}$ on a monoidal category $(\mathbf{C}, \otimes, I, \alpha, \lambda, \rho)$ is a family of morphisms $st_{C,(D,v)}\colon F(C) \otimes D \to F(C \otimes D)$, natural in $C \in \mathbf{C}$ and $(D, v\colon I \to D) \in I/\mathbf{C}$, the coslice category below D, making the following diagrams commute,

$$\begin{array}{ccc} & F(A) & \\ {\scriptstyle \rho_{F(A)}}\swarrow & & \searrow{\scriptstyle F(\rho_A)} \\ F(A)\otimes I & \xrightarrow[st_{A,(I,\mathrm{id})}]{} & F(A\otimes I) \end{array}$$

$$\begin{array}{ccccc} (F(A)\otimes X)\otimes Y & \xrightarrow{st_{A,(X,v_X)}\otimes Y} & F(A\otimes X)\otimes Y & \xrightarrow{st_{A\otimes X,(Y,v_Y)}} & F((A\otimes X)\otimes Y) \\ {\scriptstyle \alpha_{F(A),X,Y}}\downarrow & & & & \downarrow{\scriptstyle F(\alpha_{A,X,Y})} \\ F(A)\otimes(X\otimes Y) & & \xrightarrow[st_{A,(X\otimes Y,v_{X\otimes Y})}]{} & & F(A\otimes(X\otimes Y)) \end{array}$$

where $v_X\colon I \to X$ and $v_Y\colon I \to Y$ are the given points, and $v_{X\otimes Y}$ denotes the composite

$$I \xrightarrow{\rho_I} I \otimes I \xrightarrow{v_X \otimes v_Y} X \otimes Y.$$

The next step is to observe that binding signatures generate pointed strong endofunctors.

Definition 5.8. The derivation endofunctor $X \mapsto X'$ on $[\mathbb{F}, \mathbf{Set}]$ has a pointed strength, defined through the equivalence with finitary functors by

$$G(F(X)+1) \xrightarrow{G(F(X)+v_1)} G(F(X)+F(1)) \xrightarrow{G[F(in_1),F(in_2)]} G(F(X+1)).$$

Product, coproduct, and composition of endofunctors lift to pointed strong endofunctors, which yields:

Corollary 5.9 (Fiore [2008]; Fiore et al. [1999]). *For all binding signatures S, Φ_S is pointed strong.*

We arrive at last at the definition of models.

Definition 5.10. For any pointed strong endofunctor F on $\mathbf{C}$, an F-**monoid** is an object X equipped with F-algebra and monoid structure, say $a\colon F(X) \to X$, $s\colon X \otimes X \to X$, and $v\colon I \to X$, such that the following pentagon commutes.

$$\begin{array}{ccccc} F(X)\otimes X & \xrightarrow{st_{X,(X,v)}} & F(X\otimes X) & \xrightarrow{F(s)} & F(X) \\ {\scriptstyle a\otimes X}\downarrow & & & & \downarrow{\scriptstyle a} \\ X\otimes X & & \xrightarrow[s]{} & & X \end{array}$$

A morphism of F-monoids is a morphism in $\mathbf{C}$ which is a morphism both of F-algebras and of monoids. We let F - **Mon** denote the category of F-monoids and morphisms between them.

Example 5.11. For the binding signature S_λ of Example 2.12, a Φ_{S_λ}-monoid is an object X, equipped with maps $X' \to X$ and $X^2 \to X$, and compatible monoid structure. Compatibility describes how substitution should be pushed down through abstractions and applications.

5.1.2 Well-behaved presheaves. The exoticness we want to rule out only concerns the underlying functor of a model, so we just have to define well-behaved functors in $[\mathbb{F}, \mathbf{Set}]$.

Well-behavedness for a functor $T : \mathbb{F} \to \mathbf{Set}$ is about getting closed terms right. More precisely, for some finite sets m and n, an element of $T(m + n)$ which both exists in $T(m)$ and $T(n)$ should also exist in $T(\emptyset)$, and uniquely so. This says exactly that T should preserve the pullback

$$\begin{array}{ccc} \emptyset & \longrightarrow & n \\ \downarrow & & \downarrow \\ m & \longrightarrow & m + n. \end{array}$$

Furthermore, the following two known results will help us show that all non-empty pullbacks are automatically preserved.

Proposition 5.12 (Trnková [1969, Proposition 2.1]). *All endofunctors of sets preserve non-empty intersections.*

Thus, by Proposition 5.2 all functors $\mathbb{F} \to \mathbf{Set}$ preserve non-empty intersections, and we have:

Corollary 5.13. *A functor $\mathbb{F} \to \mathbf{Set}$ preserves (binary) intersections iff it preserves empty (binary) intersections.*

Definition 5.14.

- A functor $\mathbb{F} \to \mathbf{Set}$ is **well-behaved** iff it preserves binary intersections, or equivalently empty binary intersections. Let $[\mathbb{F}, \mathbf{Set}]_{wb}$ denote the full subcategory spanned by well-behaved functors.
- For any binding signature S, an object of Φ_S - **Mon** is **well-behaved** iff the underlying functor is. Let Φ_S - $\mathbf{Mon}_{wb}$ denote the full subcategory spanned by well-behaved objects.

Example 5.15. As an example of a non well-behaved finitary monad, consider the monad L of λ-calculus but edited so that $L(\emptyset) = \emptyset$.

The important result for comparing the presheaf-based approach with ours is the following.

Proposition 5.16. *The subcategory Φ_S - $\mathbf{Mon}_{wb}$ includes the initial object.*

Proof. Roughly, closed terms are isomorphic to terms in two free variables that use neither the first, nor the second. □

Let us conclude this subsection with the following observation, that for a wide class of signatures all models are in fact well behaved.

Proposition 5.17. *If the initial object DB_S of Φ_S - **Mon** has at least one closed term (i.e., $\mathrm{DB}_S(\emptyset) \neq \emptyset$), then Φ_S - $\mathbf{Mon}_{wb} = \Phi_S$ - **Mon**.*

Proof. If T is a Φ_S-monoid, then by initiality there is a morphism $\mathrm{DB}_S \to T$, and in particular a map $\mathrm{DB}_S(0) \to T(0)$. Since $\mathrm{DB}_S(0)$ is not empty by assumption, $T(0)$ cannot be empty. The result then follows from [Adámek et al. 2012, Proposition VII.7]: a monad T on **Set** either preserves the initial object, or is well-behaved. □

Remark 7. *The signatures for which the initial model has at least one closed term are those specifying at least a constant or an operation binding (at least) one variable in each argument.*

5.2 Trimming down De Bruijn monads

Let us now turn to well-behaved De Bruijn algebras. Here well-behavedness is about finitariness. However, it may not be immediately clear how to define finitariness of a De Bruijn monad.

Definition 5.18. A De Bruijn monad (X, s, v) is finitary iff each of its elements $x \in X$ has a (finite) support $N_x \in \mathbb{N}$, in the sense that for all $f : \mathbb{N} \to \mathbb{N}$ fixing the first N_x numbers, the corresponding renaming $v \circ f$ fixes x.

Proposition 5.19. *Let $x \in X$ be an element of a De Bruijn monad (X, s, v). The following are equivalent:*

(1) x has support N;
(2) given assignments $f_1, f_2 : \mathbb{N} \to X$ which coincide on the first N numbers, $x[f_1] = x[f_2]$.
(3) for any assignment $f : \mathbb{N} \to X$ fixing the first N variables (in the sense that $f(n) = v(n)$ for any $n < N$), $x[f] = x$;

Proof. (1) $\Rightarrow$ (2) Suppose given two assignments $f_1, f_2 : \mathbb{N} \to X$ such that $f_1(n) = f_2(n)$ for any $n < N$. Consider a bijection $s : \mathbb{N} \to \mathbb{N} \amalg \mathbb{N}$. For $i \in \{1, 2\}$, let $h_i : \mathbb{N} \to \mathbb{N}$ be the function fixing the first N numbers and mapping $n \geq N$ to $s^{-1}(in_i(n))$, where $in_1, in_2 : \mathbb{N} \to \mathbb{N} \amalg \mathbb{N}$ are the canonical injections. Let $u : \mathbb{N} \to X$ mapping $n < N$ to $f_1(n) = f_2(n)$ and $n \geq N$ to $[f_1, f_2](s(n))$, where $[f_1, f_2] : \mathbb{N} \amalg \mathbb{N} \to \mathbb{N}$ is the canonical morphism induced by f_1 and f_2. Then, $f_i = u \circ h_i$, for $i \in \{1, 2\}$. Thus, $x[f_i] = x[u \circ h_i] = x[v \circ h_i][u]$. Now, since x has support N, $x[v \circ h_i] = x$. hence, $x[f_1] = x[u] = x[f_2]$.
(2) $\Rightarrow$ (3) Suppose given an assignment $f : \mathbb{N} \to X$ fixing the first N variables. Then, f coincides with v on the first N variables. Thus, $x[f] = x[v] = x$.
(3) $\Rightarrow$ (1) Let $f : \mathbb{N} \to \mathbb{N}$ fixing the first N numbers. Then, $v \circ f$, as an assignment, also does. Thus, $x[v \circ f] = x$. □

Definition 5.20. For any binding signature S, let S - $\mathbf{DBAlg}_{wb}$ denote the full subcategory spanning De Bruijn S-algebras whose underlying De Bruijn monad is finitary.

Proposition 5.21. *The subcategory S - $\mathbf{DBAlg}_{wb}$ includes the initial object.*

5.3 Bridging the gap

We may at last state the relationship between initial-algebra semantics in presheaves and in De Bruijn monads:

Theorem 5.22. *Consider any binding signature S. The subcategories Φ_S - $\mathbf{Mon}_{wb}$ and S - $\mathbf{DBAlg}_{wb}$ are equivalent.*

Proof. See [Hirschowitz et al. 2021b, Appendix A]. □

Remark 8. *The moral of this is that, if one removes exotic objects from both Φ_S - **Mon** and S - **DBAlg**, then one obtains equivalent categories, which both retain the initial object. Thus, the two approaches to initial-algebra semantics of binding signatures differ only marginally.*

6 STRENGTH-BASED INTERPRETATION OF THE BINDING CONDITIONS

In the previous section, we have compared the category S - **DBAlg** of models of a binding signature in De Bruijn monads with the standard category of S-monoids [Fiore et al. 1999].

In this section, we recast the development of §3 in an abstract framework for variable binding, also initiated by Fiore et al. [1999]. This framework was originally based on **pointed strong** endofunctors on **monoidal** categories [Fiore 2008; Fiore et al. 1999]. However, we have seen in §3.2

that $\mathbb{N}$ and the tensor product equip **Set** with skew monoidal structure, so we need to use the corresponding generalisation of pointed strengths proposed by Borthelle et al. [2020].

In summary, we show that any binding signature $S = (O, ar)$ gives rise to a so-called **structurally strong** endofunctor Σ_S on **Set**, equipped with the skew monoidal structure of Corollary §3.3. The corresponding category of models Σ_S - **Mon**, as defined by Borthelle et al. [2020], is isomorphic to S - **DBAlg** – in fact essentially equal, since the only difference lies in the difference between a family $(M^{|ar(o)|} \to M)_{o\in O}$ of operations and its cotupling $\sum_{o\in O} M^{|ar(o)|} \to M$: one could easily adjust the presentation to get an exact match.

First, in §6.1, we briefly recall structurally strong endofunctors. In §6.2, we will then interpret binding signatures as such endofunctors, we recall the category of Σ-monoids, for any structurally strong endofunctor Σ, and establish the announced isomorphism of categories.

6.1 Structural strengths

We start by introducing a notion of set equipped with variables and renamings, in Definition 6.3 below. Recalling from Example 2.3 that $\mathbb{N}$ forms a De Bruijn monad, we have:

Definition 6.1. An $\mathbb{N}$-**module** is a set X equipped with an action of the monoid $\mathbb{N}^{\mathbb{N}}$. We typically denote the action by $r : X \times \mathbb{N}^{\mathbb{N}} = X \otimes \mathbb{N} \to X$.

Notation 3. *We generally denote $r(x, f)$ by $x[f]_r$, or merely $x[f]$ when clear from context.*

Example 6.2. Any De Bruijn monad X (in particular $\mathbb{N}$ itself) has a canonical structure of $\mathbb{N}$-module given by $r(x, f) = x[v \circ f]$.

Definition 6.3.

- A **pointed** $\mathbb{N}$-**module** is an $\mathbb{N}$-module (X, r), equipped with a map $v : \mathbb{N} \to X$ which is a morphism of $\mathbb{N}$-modules.
- A morphism of pointed $\mathbb{N}$-modules is a map commuting with action and point, in the obvious sense.
- Let $\mathbb{N}$ - $\mathbf{Mod}_{\mathbb{N}}$ denote the category of pointed $\mathbb{N}$-modules.

Example 6.4. The variables map of any De Bruijn monad makes it a pointed $\mathbb{N}$-module.

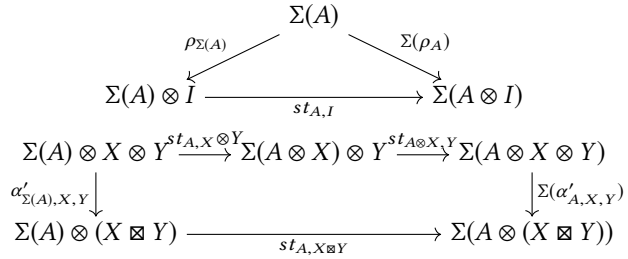
We now define a tensor product on $\mathbb{N}$-modules. Following [Lack and Street 2014, (8.1)], tensor product of (pointed) $\mathbb{N}$-modules is given by the following coequaliser in **Set**, where $r_X : X \otimes \mathbb{N} \to X$ denotes the $\mathbb{N}$-module structure on X.

$$(X \otimes \mathbb{N}) \otimes Y \xrightarrow{\alpha_{X,\mathbb{N},Y}} X \otimes (\mathbb{N} \otimes Y) \xrightarrow{X\otimes\lambda_Y} X \otimes Y \xrightarrow{\kappa_{X,Y}} X \boxtimes Y, \qquad (r_X \otimes Y) \tag{3}$$

where α denotes the associator of (**Set**, $\otimes$, $\mathbb{N}$).

By [Lack and Street 2014, Theorem 8.1], $\mathbb{N}$ - **Mod** is a skew monoidal category (with invertible right unit), and the forgetful functor is monoidal and creates monoids. In fact, this extends to $\mathbb{N}$ - $\mathbf{Mod}_{\mathbb{N}}$, and we define:

Definition 6.5 (Borthelle et al. [2020, Definition 2.11]). A **structural strength** on an endofunctor Σ : **Set** $\to$ **Set** is a natural transformation $st_{X,Y} : \Sigma(X) \otimes Y \to \Sigma(X \otimes Y)$, where X is any set and Y is a pointed $\mathbb{N}$-module, making the following diagrams commute,



where ρ denotes the right unit of $(\mathbf{Set}, \otimes, \mathbb{N})$, and $\alpha'_{A,X,Y}$ is $(A \otimes \kappa_{X,Y}) \circ \alpha_{A,X,Y}$.

6.2 De Bruijn algebras as Σ-monoids

Let us now interpret binding signatures as structurally strong endofunctors, and show that the corresponding category of models coincides with De Bruijn algebras.

The main strength of De Bruijn representation, with the current viewpoint, is that we can readily equip the endofunctor Σ_S associated to any binding signature S with a structural strength $\mathbf{dbs}_S$, which we call the **De Bruijn** strength prescribed by S on Σ_S.

Remark 9. *Let us recall that by definition, Σ_S ignores the binding information in S: we have*

$$\Sigma_S(X) = \sum_{o \in O} X^{p_o},$$

where $ar(o) = (n_1^o, \ldots, n_{p_o}^o)$ for all $o \in O$.

In order to define $\mathbf{dbs}_S$, we start by generalising Definition 2.5 from De Bruijn monads to pointed $\mathbb{N}$-modules.

Definition 6.6. Let (Y, v, r) be a pointed $\mathbb{N}$-module. Then:

- Let $\uparrow_Y : \mathbb{N} \to Y$ map any n to $v(n+1)$.
- For any assignment $\sigma : \mathbb{N} \to Y$, let

$$\begin{array}{rcl} \Uparrow \sigma(0) & = & v(0) \\ \Uparrow \sigma(n+1) & = & \sigma(n)[\uparrow]. \end{array}$$

- For any assignment $\sigma : \mathbb{N} \to Y$, we define by induction on $n \in \mathbb{N}$:

$$\begin{array}{rcl} \Uparrow^0 \sigma & = & \sigma \\ \Uparrow^{n+1} \sigma & = & \Uparrow (\Uparrow^n \sigma). \end{array}$$

Let us first define the De Bruijn strengths of the identity functor.

Definition 6.7. The **first De Bruijn strength** of the identity functor is the map

$$\begin{array}{rcl} \mathbf{dbs}_{\mathrm{id},X,Y} : X \otimes Y & \to & X \otimes Y \\ (x, \sigma) & \mapsto & (x, \sigma'), \end{array}$$

defined for all $X \in \mathbf{Set}$ and Y a pointed $\mathbb{N}$-module.

Proposition 6.8. *The first De Bruijn strength is a structural strength on the identity functor.*

This may of course by iterated:

Definition 6.9. Let the **nth De Bruijn strength** of the identity functor, $\mathbf{dbs}^n$, be defined by $\mathbf{dbs}^n_{\mathrm{id},X,Y}(x, \sigma) = (x, \Uparrow^n \sigma)$.

Proposition 6.10. *Structurally strong endofunctors compose, in the sense that if F and G are structurally strong endofunctors, then so is $G \circ F$, with structural strength given by the composite*

$$G(F(X)) \otimes Y \to G(F(X) \otimes Y) \to G(F(X \otimes Y)).$$

Thus $\mathrm{id} \circ \mathrm{id} = \mathrm{id}$ is structurally strong, and we have:

Proposition 6.11. *Each* $\mathbf{dbs}^n$ *is a structural strength on the identity functor.*

Let us now extend this to general binding arities:

Definition 6.12. For any binding arity $a = (n_1, \dots, n_p)$, the **De Bruijn strength** $\mathbf{dbs}_a$ of a on the functor $X \mapsto X^p$ is defined by

$$\begin{array}{rcl} \mathbf{dbs}_{a,X,Y} \colon X^p \otimes Y & \to & (X \otimes Y)^p \\ ((x_1, \dots, x_p), \sigma) & \mapsto & ((x_1, \Uparrow^{n_1} \sigma), \dots, (x_p, \Uparrow^{n_p} \sigma)), \end{array}$$

for all sets X and pointed $\mathbb{N}$-modules Y.

As promised, the binding condition may be expressed in terms of strengths:

Proposition 6.13. *For any binding arity $a = (n_1, \dots, n_p)$ and De Bruijn monad (M, s, v), a map $o \colon M^p \to M$ satisfies the a-binding condition iff the following pentagon commutes.*

$$\begin{array}{ccccc} X^p \otimes X & \xrightarrow{\mathbf{dbs}_{a,X,X}} & (X \otimes X)^p & \xrightarrow{s^p} & X^p \\ {\scriptstyle o \otimes X}\downarrow & & & & \downarrow{\scriptstyle o} \\ X \otimes X & & \xrightarrow[s]{} & & X \end{array} \tag{4}$$

At last, let us now define the De Bruijn strength of the endofunctor Σ_S induced by an arbitrary binding signature S. Recalling that an element of $\Sigma_S(X)$ has the form $(o, (x_1, \dots, x_{p_o}))$ for some $o \in O$, we have:

Definition 6.14. For any binding signature $S = (O, ar)$, the **De Bruijn strength** $\mathbf{dbs}_S$ of the induced endofunctor Σ_S is defined by $\mathbf{dbs}_S = \sum_{o \in o} \mathbf{dbs}_{ar(o)}$,or more concretely

$$\begin{array}{rcl} \Sigma_S(X) \otimes Y & \to & \Sigma_S(X \otimes Y) \\ ((o, (x_1, \dots, x_{p_o})), \sigma) & \mapsto & (o, ((x_1, \Uparrow^{n_1} \sigma), \dots, (x_{p_o}, \Uparrow^{n_{p_o}} \sigma))), \end{array}$$

for all sets X and pointed $\mathbb{N}$-modules Y.

Proposition 6.15. *The De Bruijn strength of Σ_S is a structural strength.*

In order to relate the initial-algebra semantics of §3 to the strength-based approach of Borthelle et al. [2020]; Fiore [2008]; Fiore et al. [1999], let us recall the definition of models in the latter.

Definition 6.16. Given a structurally strong endofunctor Σ, a Σ-**monoid** is an object X, equipped with monoid and Σ-algebra structure, say $s \colon X \otimes X \to X$, $v \colon \mathbb{N} \to X$, and $a \colon \Sigma(X) \to X$, making the following pentagon commute.

$$\begin{array}{ccccc} \Sigma(X) \otimes X & \xrightarrow{\mathbf{dbs}_{S,X,X}} & \Sigma(X \otimes X) & \xrightarrow{\Sigma(s)} & \Sigma(X) \\ {\scriptstyle a \otimes X}\downarrow & & & & \downarrow{\scriptstyle a} \\ X \otimes X & & \xrightarrow[s]{} & & X \end{array} \tag{5}$$

A morphism of Σ-monoids is a map which is both a monoid and a Σ-algebra morphism.

Let Σ - **Mon** denote the category of Σ-monoids and morphisms between them.

We may at last relate the initial-algebra semantics of §3 with the strength-based approach:

Proposition 6.17. *For any binding signature $S = (O, ar)$ and De Bruijn monad (M, s, v) equipped with a map $o_M \colon M^p \to M$ for all $o \in O$ with $ar(o) = (n_1, \ldots, n_p)$, the following are equivalent:*

(i) *each map $o_M \colon M^p \to M$ satisfies the a-**binding condition** w.r.t. (s, v);*
(ii) *the corresponding map $\Sigma_S M \to M$ renders the pentagon (5) (with $\Sigma := \Sigma_S$) commutative.*

Corollary 6.18. *For any binding signature S, we have an isomorphism Σ_S - **Mon** $\cong$ S - **DBAlg** of categories over* **Set**.

This readily entails the following (bundled) reformulation of Theorems 2.15 and 3.6.

Corollary 6.19. *Consider any binding signature $S = (O, ar)$, and let DB_S denote the initial $(\mathbb{N} + \Sigma_S)$-algebra, with structure maps $v \colon \mathbb{N} \to \mathrm{DB}_S$ and $a \colon \Sigma_S(\mathrm{DB}_S) \to \mathrm{DB}_S$. Then:*

(i) *There exists a unique substitution map $s \colon \mathrm{DB}_S \otimes \mathrm{DB}_S \to \mathrm{DB}_S$ such that*
   - *the map $\mathbb{N} \otimes \mathrm{DB}_S \xrightarrow{v \otimes \mathrm{DB}_S} \mathrm{DB}_S \otimes \mathrm{DB}_S \xrightarrow{s} \mathrm{DB}_S$ coincides with the left unit of the skew monoidal structure $(n, f) \mapsto f(n)$, and*
   - *the pentagon (5) (with $\Sigma := \Sigma_S$) commutes.*
(ii) *This substitution map turns (DB_S, v, s, a) into a Σ_S-monoid.*
(iii) *This Σ_S-monoid is initial in Σ_S - **Mon**.*

Proof. Let **Mon**(**Set**) denote the category of monoids in **Set** for the skew monoidal structure. We have an equality **Mon**(**Set**) = **DBMnd** of categories, and the algebra structure $\Sigma_S(\mathrm{DB}_S) \to \mathrm{DB}_S$ is merely the cotupling of the maps o_{DB_S} of Theorem 2.15. This correspondence translates one statement into the other. □

Remark 10. *This result hints at a potential push-button proof of Theorems 2.15 and 3.6 (and Corollary 6.19). Indeed, it is almost an instance of Borthelle et al. [2020, Theorem 2.15]: the latter is stated for general skew monoidal categories instead of merely* **Set**, *but does not directly apply in the present setting, because it assumes that the tensor product is finitary in the second argument.*

7 MODULE-BASED INTERPRETATION OF THE BINDING CONDITIONS

In the previous section, we have recast the initial-algebra semantics of §3 into the strength-based approach to variable binding [Borthelle et al. 2020; Fiore et al. 1999]. In this section, we present an alternative recasting, into the module-based approach to variable binding [Hirschowitz and Maggesi 2007, 2010]. The present section may safely be skipped, as it is not used in the rest of the paper.

In the context of monads over sets, it is well-known that, for a model X of a signature S, the conditions expressing the compatibility of operations with substitution may be synthetised by saying that the structure map $\Sigma_S(X) \to X$ is a module morphism with respect to suitable, natural module structures on $\Sigma_S(X)$ and X, respectively: this interpretation, which involves an operation called **derivation** of modules, goes back at least to Hirschowitz and Maggesi [2007]. Let us now explain how it may be adapted to the framework of the present paper. A main feature is that here derivation of modules does not change the support, hence concerns substitution only. In §7.1, we introduce modules over a De Bruijn monad. In §7.2, we define module derivation. Finally, in §7.3, we present our alternative interpretation of the binding conditions.

7.1 Modules over De Bruijn monads and first-order signatures

There is a general notion of module over a monoid in a monoidal (or skew monoidal see Borthelle et al. [2020]) category, which we do not refer to: we just give the instance we are concerned with. Intuitively, if X is a De Bruijn monad, an X-module is a set that admits substitution of variables by elements of X:

Definition 7.1. For any De Bruijn monad (X, s, v), an X-**module** is a set A equipped with a map

$$\begin{array}{rcl} A \times X^{\mathbb{N}} & \to & A \\ (a, f) & \mapsto & a[f], \end{array}$$

such that, for all $a \in A$ and $f, g \in X^{\mathbb{N}}$, we have $a[f][g] = a[f[g]]$ and $a[v] = a$.

Of course, we have:

Example 7.2. Any De Bruijn monad X is itself an X-module, with map $X \times X^{\mathbb{N}} \to X$ given by substitution, which is coherent with our abbreviation.

As algebras for the monad $- \otimes X$, cartesian product of sets lifts to X-modules, which yields in particular the right module for specifying application in the λ-calculus:

Example 7.3. For any De Bruijn monad X, X^2 is an X-module, with $(x_1, x_2)[f] = (x_1[f], x_2[f])$.

More generally:

Definition 7.4. For any sequence $A_1, \ldots, A_n$ of X-modules, the cartesian product $A_1 \times \ldots \times A_n$ is again an X-module, with action given by $(a_1, \ldots, a_n)[f] = (a_1[f], \ldots, a_n[f])$.

Viewing the arity of an operation as an X-module A, compatibility with substitution may be stated by requiring that the structure map $A \to X$ be a module morphism, in the following sense.

Definition 7.5.

- An X-module morphism $(A, \rho) \to (B, \tau)$, or $A \to B$ for short, is a map $f \colon A \to B$ which commutes with substitution, in the sense that $f(a)[g] = f(a[g])$, for all $a \in A$ and $g \in X^{\mathbb{N}}$.
- Let X - **Mod** denote the category of X-modules, with X-module morphisms between them.

Proposition 7.6. *For any first-order signature S, equipping a De Bruijn monad (X, s, v) with a S - **DBAlg** structure is equivalent to providing a module morphism $o_X \colon X^p \to X$ for each operation o with arity p.*

Proof. The action of X on X^p is by definition

$$\begin{array}{rcl} X^p \times X^{\mathbb{N}} & \to & X \\ ((x_1, \ldots, x_p), f) & \mapsto & (x_1[f], \ldots, x_p[f]). \end{array}$$

Thus, the module morphism condition $o_X(x_1, \ldots, x_p)[f] = o_X((x_1, \ldots, x_p)[f])$ agrees with the binding condition (2) specialised to first-order signatures. □

We now need to generalise this to binding arities. The good news is that these also form modules, as we now show.

7.2 Derivation of substitution for modules

In this subsection, we explain the derivation of modules. This derivation does not change the carrier of the module, hence it acts on the substitution map only. In fact, it acts via the second argument of substitution, namely the assignment, as in §3.

Our starting point is the functor $- \times X \colon X$ - **Mod** $\to X$ - **Mod** on X-modules mapping any A to $A \times X$. Let us prove:

Proposition 7.7. *The endofunctor $- \times X$ has a right adjoint mapping A to $A^{(1)}$, called the **derivative** of A, where $A^{(1)}$ has the same carrier as A, with action given by*

$$\begin{array}{rcl} A \times X^{\mathbb{N}} & \to & A \\ (a, f) & \mapsto & a[f'], \end{array}$$

where f' is as in Definition 2.5.

Of course we may iterate this endofunctor:

Definition 7.8. Let $A^{(0)} = A$, and $A^{(n+1)} = (A^{(n)})^{(1)}$.

7.3 Interpreting the binding conditions

Starting from a De Bruijn monad X with the corresponding "tautological" X-module X, and combining derivations and cartesian products, we get, for each binding arity $a := (n_1, \ldots, n_p)$ of length p, a specific X-module $X^{(a)} := X^{(n_1)} \times \ldots \times X^{(n_p)}$ with carrier X^p.

They allow us to reformulate the binding conditions as follows:

Proposition 7.9. *Let $a = (n_1, \ldots, n_p)$ be a binding arity of length p, and (M, s, v) be a De Bruijn monad. Then a map $o\colon M^p \to M$ satisfies the a-**binding condition** w.r.t. (s, v) iff it is an M-module morphism from $M^{(a)}$ to M.*

Summing over the operations of the given binding signature, we get:

Proposition 7.10. *For any binding signature S, equipping a De Bruijn monad (X, s, v) with a S-**DBAlg** structure is equivalent to providing a module morphism $\sum_{o\in O} X^{(ar(o))} \to X$.*

8 SIMPLY-TYPED EXTENSION

In this section, we extend the framework of §2–3, which is untyped, to the simply-typed case. The development essentially follows the same pattern, replacing **Set** with $\mathbf{Set}^{\mathbb{T}}$.

We fix in the whole section a set $\mathbb{T}$ of **types**, and call **$\mathbb{T}$-sets** the objects of $\mathbf{Set}^{\mathbb{T}}$. A morphism $X \to Y$ is a family $(X(\tau) \to Y(\tau))_{\tau\in\mathbb{T}}$ of maps.

8.1 De Bruijn $\mathbb{T}$-monads

In this subsection, we define the typed analogue of De Bruijn monads.

The role of $\mathbb{N}$ will be played in the typed context by the following $\mathbb{T}$-set.

Definition 8.1. Let $\mathbf{N} \in \mathbf{Set}^{\mathbb{T}}$ be defined by $\mathbf{N}(\tau) = \mathbb{N}$.

Definition 8.2. Given a $\mathbb{T}$-set X, an **X-assignment** is a morphism $\mathbf{N} \to X$. We sometimes merely use "assignment" when X is clear from context.

Notation 4.

- *We observe that $\mathbb{T}$-sets form a cartesian closed category, where the exponential object X^Y is given by $(X^Y)(\tau) = X(\tau)^{Y(\tau)}$.*
- *We distinguish it from the hom-set by writing the latter $[Y, X]$.*
- *For any set A and $\mathbb{T}$-set X, let $A \cdot X = \sum_{a\in A} X$ denote the A-fold coproduct of X.*

Example 8.3. Consider arbitrary $\mathbb{T}$-sets X, Y, and Z.

- The $\mathbb{T}$-set $[X, Y] \cdot Z$ is such that for all types τ, we have
$$([X, Y] \cdot Z)(\tau) = [X, Y] \cdot Z(\tau) = [X, Y] \times Z(\tau).$$
- The $\mathbb{T}$-set $Y^X \times Z$ is such that for all types τ, we have
$$(Y^X \times Z)(\tau) = Y(\tau)^{X(\tau)} \times Z(\tau).$$

We will use the former for generalising substitution to the typed case.

Notation 5. *For coherence with the untyped case, we tend to write an element of $([\mathbf{N}, Y] \cdot X)(\tau)$ as (x, f), with $x \in X(\tau)$ and $f\colon \mathbf{N} \to Y$.*

Notation 6. *Consider any map $s\colon [\mathbf{N}, Y] \cdot X \to Z$.*

- *For all $\tau \in \mathbb{T}$, $x \in X(\tau)$, and $g\colon \mathbf{N} \to Y$, we write $x[g]_{s,\tau}$ for $s_\tau(x, g)$, or even $x[g]$ when s and τ are clear from context.*

- *Furthermore, s gives rise to the map* $\begin{array}{rcl}[\mathbf{N}, Y] \cdot X^{\mathbf{N}} & \to & Z^{\mathbf{N}} \\ (g, f) & \mapsto & \tau, n \mapsto f(n)[g]_{s,\tau}.\end{array}$
  We use similar notation for this map, i.e., $f[g]_{s,\tau}(n) := f(n)[g]_{s,\tau}$, or $f[g](n) = f(n)[g]$ when s and τ are clear from context.

The definition of De Bruijn monads generalises almost *mutatis mutandis*:

Definition 8.4. A **De Bruijn $\mathbb{T}$-monad** is a $\mathbb{T}$-set X, equipped with

- a **substitution** morphism $s\colon [\mathbf{N}, X]\cdot X \to X$, which takes an element $x \in X$ and an assignment $f\colon \mathbf{N} \to X$, and returns an element $x[f]$, and
- a **variables** morphism $v\colon \mathbf{N} \to X$,

such that for all $x \in X$, and $f, g\colon \mathbf{N} \to X$, we have

$$x[f][g] = x[f[g]] \qquad v(n)[f] = f(n) \qquad x[v] = x.$$

Example 8.5. The $\mathbb{T}$-set $\mathbf{N}$ itself is clearly a De Bruijn $\mathbb{T}$-monad, with variables given by the identity and substitution $[\mathbf{N}, \mathbf{N}] \cdot \mathbf{N} \to \mathbf{N}$ given by evaluation. It is in fact initial in $\mathbf{DBMnd}(\mathbb{T})$.

Example 8.6. The set Λ_{ST} of simply-typed λ-terms with free variables of type τ in $\mathbb{N} \times \{\tau\}$, considered equivalent modulo α-renaming, forms a De Bruijn monad. Variables $\mathbf{N} \to \Lambda_{\mathrm{ST}}$ are given by mapping, at any τ, any $n \in \mathbb{N}$ to the variable (n, τ). Substitution $[\mathbf{N}, \Lambda_{\mathrm{ST}}] \cdot \Lambda_{\mathrm{ST}} \to \Lambda_{\mathrm{ST}}$ is standard, capture-avoiding substitution. One main purpose of this section is to characterise Λ_{ST} by a universal property, and reconstruct it categorically.

8.2 Morphisms of De Bruijn $\mathbb{T}$-monads

Definition 8.7. A morphism $(X, s, v) \to (Y, t, w)$ between De Bruijn $\mathbb{T}$-monads is a morphism $f\colon X \to Y$ of $\mathbb{T}$-sets commuting with substitution and variables, in the sense that for all $\tau \in \mathbb{T}$, $x \in X(\tau)$, and $g\colon \mathbf{N} \to X$ we have $f_\tau(x[g]) = f_\tau(x)[f \circ g]$ and $f \circ v = w$.

Remark 11. *More explicitly, the first axiom says: $f_\tau(s_\tau(x, g)) = t_\tau(f_\tau(x), f \circ g)$.*

Proposition 8.8. *De Bruijn $\mathbb{T}$-monads and morphisms between them form a category* $\mathbf{DBMnd}(\mathbb{T})$.

8.3 De Bruijn $\mathbb{T}$-monads as relative monads

The presentation based on relative monads extends to the typed setting, by replacing the functor $\mathbb{N}\colon 1 \to \mathbf{Set}$ with $\mathbf{N}\colon 1 \to \mathbf{Set}^{\mathbb{T}}$, so that $\mathrm{Lan}_{\mathbf{N}}(X)(Y) \cong [\mathbf{N}, Y] \cdot X$. Thus, $\mathbf{DBMnd}(\mathbb{T})$ is equivalently the category of monads relative to the functor $1 \to \mathbf{Set}^{\mathbb{T}}$ picking $\mathbf{N}$. For the record, let us explicitly introduce the corresponding tensor product.

Definition 8.9. For any $\mathbb{T}$-sets X and Y, let $X \otimes Y = [\mathbf{N}, Y] \cdot X$.

Notation 7. *For coherence with the untyped case, we tend to write an element of $(X \otimes Y)(\tau)$ as (x, f), with $x \in X(\tau)$ and $f\colon \mathbf{N} \to Y$.*

8.4 Initial-algebra semantics

We now adapt the initial-algebra semantics of §3 to the typed case.

Let us start by generalising lifting to the typed case. This relies on a typed form of lifting, which acts on all variables of a given type, leaving all other variables untouched.

Definition 8.10. Let (X, s, v) denote any De Bruijn $\mathbb{T}$-monad. We first define a typed analogue $\uparrow_{@\tau, X}$ of the $\uparrow$ of Definition 2.5, as below left, and then the **lifting** of any assignment $\sigma\colon \mathbf{N} \to X$ as below right.

$$\begin{array}{rcll}(\uparrow_{@\tau,X})_\tau(n) & = & v_\tau(n+1) & \\ (\uparrow_{@\tau,X})_{\tau'}(n) & = & v_{\tau'}(n) & (\text{if } \tau \neq \tau')\end{array} \qquad \begin{array}{rcll}(\Uparrow^\tau \sigma_\tau)(0) & = & v_\tau(0) & \\ (\Uparrow^\tau \sigma_\tau)(n+1) & = & \sigma_\tau(n)[\uparrow_{@\tau,X}] & \\ (\Uparrow^\tau \sigma_{\tau'})(n) & = & \sigma_{\tau'}(n)[\uparrow_{@\tau,X}] & (\text{if } \tau \neq \tau').\end{array}$$

Finally, for any sequence $\gamma = (\tau_1, \ldots, \tau_n)$ of types, we define $\Uparrow^\gamma \sigma$ inductively, by $\Uparrow^\varepsilon \sigma = \sigma$ and $\Uparrow^{\gamma,\tau} \sigma = \Uparrow^\tau (\Uparrow^\gamma \sigma)$, where ε denotes the empty sequence.

8.4.2 Binding arities and binding conditions. We may now generalise binding arities and the binding conditions to the typed setting.

Definition 8.11.

- A **first-order arity** is a pair $((\tau_1, \ldots, \tau_p), \tau) \in \mathbb{T}^* \times \mathbb{T}$ of a list of types and a type.
- A **binding arity** is a tuple $a = (((\gamma_1, \tau_1), \ldots, (\gamma_p, \tau_p)), \tau)$, where each $\gamma_i \in \mathbb{T}^*$ is a list of types, and each τ_i, as well as τ, are types. In other words, $a \in (\mathbb{T}^* \times \mathbb{T})^* \times \mathbb{T}$.
- The **first-order arity** $|a|$ **associated** to a is $((\tau_1, \ldots, \tau_p), \tau) \in \mathbb{T}^* \times \mathbb{T}$.

Remark 12. *The arity $(((\gamma_1, \tau_1), \ldots, (\gamma_p, \tau_p)), \tau)$ of an operation $o \in O$ may be understood as follows:*

- *τ is the return type;*
- *$(\tau_1, \ldots, \tau_p)$ are the argument types;*
- *each list $\gamma_i = (\tau_1^i, \ldots, \tau_{q_i}^i)$ specifies that the ith argument should be considered as binding q_i variables, of respective types $\tau_1^i, \ldots, \tau_{q_i}^i$.*

Notation 8. *We write any arity $(((\gamma_1, \tau_1), \ldots, (\gamma_p, \tau_p)), \tau)$ as an inference rule* $\dfrac{\gamma_1 \vdash \tau_1 \qquad \ldots \qquad \gamma_p \vdash \tau_p}{\vdash \tau}$.

$$\frac{\gamma_1 \vdash \tau_1 \qquad \ldots \qquad \gamma_p \vdash \tau_p}{\vdash \tau}.$$

Example 8.12. The binding signature for simply-typed λ-calculus has two operations $\mathrm{lam}_{\tau,\tau'}$ and $\mathrm{app}_{\tau,\tau'}$ for all types $\tau, \tau' \in \mathbb{T}$, of respective arities $\dfrac{\tau \vdash \tau'}{\vdash \tau \to \tau'}$ and $\dfrac{\vdash \tau \to \tau' \qquad \vdash \tau}{\vdash \tau'}$.

This allows us to generalise binding conditions, as follows.

Definition 8.13. Let $a = (((\gamma_1, \tau_1), \ldots, (\gamma_p, \tau_p)), \tau)$ be any binding arity, and M be any set equipped with morphisms $s \colon [\mathbf{N}, M] \cdot M \to M$ and $v \colon \mathbf{N} \to M$. An **operation of binding arity** a is a map $o \colon M(\tau_1) \times \ldots \times M(\tau_p) \to M(\tau)$ satisfying the following a**-binding condition** w.r.t. (s, v):

$$\forall \sigma \colon \mathbf{N} \to M, x_1, \ldots, x_p \in M(\tau_1) \times \ldots \times M(\tau_p), \quad o(x_1, \ldots, x_p)[\sigma] = o(x_1[\Uparrow^{\gamma_1} \sigma], \ldots, x_p[\Uparrow^{\gamma_p} \sigma]). \tag{6}$$

8.4.3 Binding signatures and algebras. Finally, we generalise signatures and their models to the typed setting, and state a typed initiality theorem.

Definition 8.14. A **first-order typed signature** consists of a set O of **operations**, equipped with an **arity** map $ar \colon O \to \mathbb{T}^* \times \mathbb{T}$.

Definition 8.15. Consider a first-order signature $S := (O, ar)$.

- An S**-algebra** is a set X, together with, for each operation $o \in O$ with arity $((\tau_1, \ldots, \tau_p), \tau)$, a map $o_X \colon X(\tau_1) \times \ldots \times X(\tau_p) \to X(\tau)$.

- A morphism $X \to Y$ of S-algebras is a map between underlying sets commuting with operations, in the sense that for each $o \in O$, letting $((\tau_1, \dots, \tau_p), \tau) := ar(o)$, we have for all $x_1, \dots, x_p \in X(\tau_1) \times \dots \times X(\tau_p)$, $f_\tau(o_X(x_1, \dots, x_p)) = o_Y(f_{\tau_1}(x_1), \dots, f_{\tau_p}(x_p))$.

We denote by S - **alg** the category of S-algebras and morphisms between them.

Definition 8.16.

- A $\mathbb{T}$-**binding signature** consists of a set O of **operations**, equipped with an arity map $O \to (\mathbb{T}^* \times \mathbb{T})^* \times \mathbb{T}$.
- The first-order signature $|S|$ associated with a binding signature $S := (O, ar)$ is $|S| := (O, |ar|)$, where $|ar| \colon O \to \mathbb{T}^* \times \mathbb{T}$ maps any $o \in O$ to $|ar(o)|$.

Let us now present the notion of De Bruijn S-algebra:

Definition 8.17. Consider any $\mathbb{T}$-binding signature $S := (O, ar)$.

- A **De Bruijn** S-**algebra** consists of a De Bruijn $\mathbb{T}$-monad (X, s, v), together with, for all $o \in O$, an operation o_X of binding arity $ar(o)$.
- A morphism of De Bruijn S-algebras is a map $f \colon X \to Y$ between underlying sets, which is a morphism both of De Bruijn monads and of $|S|$-algebras.

We denote by S - **DBAlg** the category of De Bruijn S-algebras and morphisms between them.

In order to extend the initiality theorem to the typed case, we need to define the endofunctor induced by a $\mathbb{T}$-binding signature:

Definition 8.18 (Induced endofunctor).

- For any $\tau \in \mathbb{T}$, let $\mathbf{y}_\tau$ denote the $\mathbb{T}$-set defined by $\begin{aligned} \mathbf{y}_\tau(\tau) &= 1 \\ \mathbf{y}_\tau(\tau') &= \emptyset \quad (\text{if } \tau' \neq \tau). \end{aligned}$
- The endofunctor Σ_a induced by any arity $a = \dfrac{\tau_1^1, \dots, \tau_{q_1}^1 \vdash \tau_1 \quad \dots \quad \tau_1^p, \dots, \tau_{q_p}^p \vdash \tau_p}{\vdash \tau}$ is defined by $\Sigma_a(X) = (X(\tau_1) \times \dots \times X(\tau_p)) \cdot \mathbf{y}_\tau$. Thus, a Σ_a-algebra is a $\mathbb{T}$-set X equipped with a morphism $(X(\tau_1) \times \dots \times X(\tau_p)) \cdot \mathbf{y}_\tau \to X$,or equivalently a map $X(\tau_1) \times \dots \times X(\tau_p) \to X(\tau)$.
- The endofunctor Σ_S induced by any $\mathbb{T}$-binding signature $S = (O, ar)$ is defined by

$$\Sigma_S(X) = \sum_{o \in O} \Sigma_{ar(o)}(X).$$

Theorem 8.19. *For any $\mathbb{T}$-binding signature S, let DB_S denote the initial $(\mathbf{N} + \Sigma_S)$-algebra, with structure maps $v \colon \mathbf{N} \to \mathrm{DB}_S$ and $a \colon \Sigma_S(\mathrm{DB}_S) \to \mathrm{DB}_S$, inducing maps $o_{\mathrm{DB}_S} \colon \mathrm{DB}_S(\tau_1) \times \dots \times \mathrm{DB}_S(\tau_p) \to \mathrm{DB}_S(\tau)$ for all $o \in O$ with $ar(o) = (((\gamma_1, \tau_1), \dots, (\gamma_p, \tau_p)), \tau)$. Then:*

(i) *There exists a unique map $s \colon [\mathbf{N}, \mathrm{DB}_S] \cdot \mathrm{DB}_S \to \mathrm{DB}_S$ such that*
   - *for all $\tau \in \mathbb{T}$, $n \in \mathbb{N}$, and $f : \mathbf{N} \to \mathrm{DB}_S$, $s_\tau(v(n), f) = f_\tau(n)$, and*
   - *for all $o \in O$, the map o_{DB_S} satisfies the $ar(o)$-binding condition w.r.t. (s, v).*

(ii) *This map turns (DB_S, v, s, a) into a De Bruijn S-algebra.*

(iii) *This De Bruijn S-algebra is initial in S - **DBAlg**.*

8.5 Application: values in simply-typed λ-calculus

While we saw in Example 8.12 that the De Bruijn monad of simply-typed λ-calculus terms admits a simple signature, there is another relevant, related monad, whose elements at any type are **values** of that type. (Indeed, values are closed under value substitution.) For this monad, the situation is more complex, as already exposed in Hirschowitz et al. [2021a].

First of all, let us consider **application binary trees**, i.e., proof derivations generated by the following rules, $\dfrac{}{\sigma \vdash_{BT} \sigma}$ $\qquad$ $\dfrac{\Gamma \vdash_{BT} \sigma \to \tau \qquad \Delta \vdash_{BT} \sigma}{\Gamma, \Delta \vdash_{BT} \tau}$ where Γ, Δ denotes concatenation of lists of simple types. Thus a proof of $\Gamma \vdash_{BT} \tau$ is essentially a simply-typed term involving only application, with one, linearly used free variable for each type in Γ, in the same order.

Definition 8.20. Let BT^{Γ}_{σ} denote the set of such application binary trees with conclusion $\Gamma \vdash_{BT} \sigma$.

We then take as binding signature S_{λ_v} for simply-typed values the one with one operation $L_{\pi,\sigma}$ of arity $\dfrac{\sigma \vdash \tau_1 \qquad \dots \qquad \sigma \vdash \tau_n}{\sigma \to \tau}$ for each simple type σ and application binary tree $\pi \in BT^{\tau_1,\dots,\tau_n}_{\tau}$.

Example 8.21. For a simple example, if π is merely the axiom $\tau \mapsto \tau$, then $L_{\pi,\sigma}$ has the arity $\dfrac{\sigma \vdash \tau}{\vdash \sigma \to \tau}$ of λ-abstraction. In this case, $L_{\pi,\sigma}$ is thought of as forming $\lambda x : \sigma.v^{\tau}$ from any value v of type τ with an additional variable of type σ.

Example 8.22. For a less trivial, yet basic example, if π is

$$\frac{\tau_1 \to \tau_2 \vdash_{BT} \tau_1 \to \tau_2 \qquad \tau_1 \vdash_{BT} \tau_1}{\tau_1 \to \tau_2, \tau_1 \vdash_{BT} \tau_2}\,, \qquad \text{then } L_{\pi,\sigma} \text{ has arity} \qquad \frac{\sigma \vdash \tau_1 \to \tau_2 \qquad \sigma \vdash \tau_1}{\sigma \to \tau_2}\,.$$

This operation is thought of as forming $\lambda x : \sigma.(f^{\tau_1 \to \tau_2}\ a^{\tau_1})$ from values f and a.

By Theorem 8.19, the initial De Bruijn S_{λ_v}-algebra has as carrier the initial algebra for the induced endofunctor, which is by construction the subset of values.

9 EQUATIONS

In this section, we introduce a notion of equational theory for specifying (typed) De Bruijn monads, following ideas from Fiore and Hur [2009]. We start with an elementary presentation, and then give a module-based one.

9.1 Elementary presentation

Definition 9.1. A **De Bruijn equational theory** consists of

- two binding signatures S and T, and
- two functors $L, R \colon S\text{-}\mathbf{DBAlg} \to T\text{-}\mathbf{DBAlg}$ over $\mathbf{DBMnd}(\mathbb{T})$, i.e., making the following diagram commute serially, where U^S and U^T denote the forgetful functors.

$$\begin{array}{ccc} S\text{-}\mathbf{DBAlg} & \xrightarrow{\quad L,R \quad} & T\text{-}\mathbf{DBAlg} \\ & {\scriptstyle U^S} \searrow \quad \swarrow {\scriptstyle U^T} & \\ & \mathbf{DBMnd}(\mathbb{T}) & \end{array}$$

Example 9.2. Recalling the binding signature S_Λ for λ-calculus from Example 2.12, let us define a De Bruijn equational theory for β-equivalence. We take $T_\beta = (1, 0)$, and for any De Bruijn S_Λ-algebra X,

- $L(X)$ has as structure map $\begin{array}{rcl} X^2 & \to & X \\ (e_1, e_2) & \mapsto & \mathrm{app}(\mathrm{lam}(e_1), e_2) \end{array}$ while
- $R(X)$ has as structure map $\begin{array}{rcl} X^2 & \to & X \\ (e_1, e_2) & \mapsto & e_1[e_2 \cdot \mathrm{id}]. \end{array}$

  (Here $e_2 \cdot \mathrm{id}$ denotes the assignment $0 \mapsto e_2$, $n + 1 \mapsto v(n)$.)

Definition 9.3. Given an equational theory $E = (S, T, L, R)$, a De Bruijn E-**algebra** is a De Bruijn S-algebra X such that $L(X) = R(X)$.

Let E - **DBAlg** denote the category of E-algebras, with morphisms of De Bruijn S-algebras between them.

Remark 13. *The category E -* **DBAlg** *is an equaliser of L and R in* **CAT**.

Let us now turn to characterising the initial De Bruijn E-algebra, for any De Bruijn equational theory E. For this, we introduce the following relation.

Definition 9.4. For any De Bruijn equational theory $E = (S, T, L, R)$, with $S = (O, ar)$ and $T = (O, ar')$, let DB_S denote the initial $(\mathbf{N} + \Sigma_S)$-algebra. We define $\sim_E$ to be the smallest equivalence relation on DB_S satisfying the following rules,

$$\frac{}{o'_{L(\mathrm{DB}_S)}(e_1, \dots, e_p) \sim_E o'_{R(\mathrm{DB}_S)}(e_1, \dots, e_p)} \qquad \frac{e_1 \sim_E e'_1 \quad \dots \quad e_q \sim_E e'_q}{o_{\mathrm{DB}_S}(e_1, \dots, e_q) \sim_E o_{\mathrm{DB}_S}(e'_1, \dots, e'_q)}$$

for all $e, e_1, \dots$ in DB_S, $o' \in O'$ with $|ar'(o')| = p$, and $o \in O$ with $|ar(o)| = q$.

Example 9.5. For the equational theory of Example 9.2, the first rule instantiates precisely to the β-rule, while the second enforces congruence.

Theorem 9.6. *For any equational theory $E = (S, T, L, R)$, E -* **DBAlg** *admits an initial object, whose carrier set is the quotient $\mathrm{DB}_S/{\sim_E}$.*

Proof. This has been mechanised in Coq: existence is called `quotsyntax.ini_morE_model_mor`, while unicity is called `quotsyntax.ini_morE_unique`. □

Example 9.7. The initial model for the equational theory of Example 9.2 is the quotient of λ-terms in De Bruijn representation by β-equivalence.

Remark 14. *In [Hirschowitz et al. 2021b, §9], we mention an equivalent way of defining De Bruijn equational theories in terms of modules.*

9.2 Module-based presentation

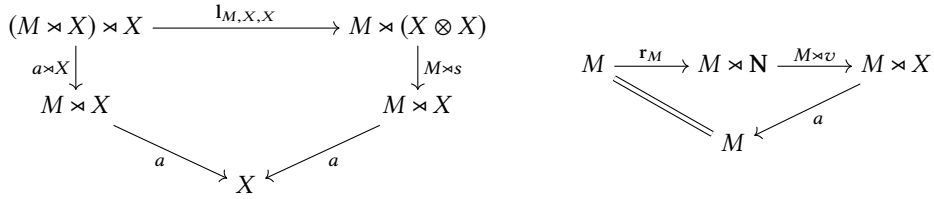
In this section, we mention an equivalent way of defining De Bruijn equational theories in terms of modules, which will be useful below. In the typed case, for any De Bruijn $\mathbb{T}$-monad X, it is convenient to define a "heterogeneous" notion of X-module:

Definition 9.8. For any object M of any category $\mathbf{C}$ with coproducts, and $\mathbb{T}$-set X, let $M \rtimes X = [\mathbf{N}, X] \cdot M$ (recall that $\mathbf{N} \in \mathbf{Set}^{\mathbb{T}}$ is defined by $\mathbf{N}(\tau) = \mathbb{N}$).

Definition 9.9.

- Let $\mathbf{a}_{M,X,Y} \colon (M \rtimes X) \rtimes Y \to M \rtimes (X \otimes Y)$ denote the composite
$$[\mathbf{N}, X] \cdot [\mathbf{N}, Y] \cdot M \cong ([\mathbf{N}, X] \times [\mathbf{N}, Y]) \cdot M \to [\mathbf{N}, [\mathbf{N}, Y] \cdot X] \cdot M,$$
where the involved map $[\mathbf{N}, X] \times [\mathbf{N}, Y] \to [\mathbf{N}, [\mathbf{N}, Y] \cdot X]$ sends any (f, g) to $\mathbf{N} \xrightarrow{f} X \xrightarrow{in_g} [\mathbf{N}, Y] \cdot X$.
- Let $\mathbf{r}_M \colon M \to M \rtimes \mathbf{N}$ denote $M \xrightarrow{in_{\mathrm{id}_{\mathbf{N}}}} [\mathbf{N}, \mathbf{N}] \cdot M$.

Definition 9.10. For any De Bruijn $\mathbb{T}$-monad (X, s, v) and category $\mathbf{C}$ with coproducts, a $\mathbf{C}$-**valued** X-**module** is an object $M \in \mathbf{C}$, equipped with an action $a \colon M \rtimes X \to M$ making the following diagrams commute.



An X-module morphism is a morphism between underlying objects that commutes with action. Let X - **Mod**(**C**) denote the category of **C**-valued X-modules and morphisms between them.

Remark 15. *When* $\mathbf{C} = \mathbf{Set}$*, denoting the action by* $(m, f) \mapsto m[f]$*, the axioms become:* $m[f][g] = m[f[g]]$ *and* $m[v] = m$ *for all* $f, g \colon \mathbf{N} \to X$ *and* $m \in M$*.*

Example 9.11.

- Each $X(\tau)$ is a set-valued X-module, with action given by substitution at τ.
- For any $\gamma \in \mathbb{T}^*$, if M is any **C**-valued X-module, then let $M^{(\gamma)}$ denote the **C**-valued X-module consisting of
  - the same underlying set M, and
  - the action given by $(m, f) \mapsto m[\Uparrow^\gamma f]$, or more generally

$$[\mathbf{N}, X] \cdot M \xrightarrow{(-)^{(\gamma)} \cdot M} [\mathbf{N}, X] \cdot M \xrightarrow{a} M.$$

Definition 9.12. A **modular equational theory** consists of two typed binding signatures S and T as before, plus, for each operation o from T with arity

$$\frac{\gamma_1 \vdash \tau_1 \qquad \dots \qquad \gamma_p \vdash \tau_p}{\vdash \tau}$$

and De Bruijn S-algebra X, two maps

$$L^o_X, R^o_X \colon X^{(\gamma_1)}(\tau_1) \times \ldots \times X^{(\gamma_p)}(\tau_p) \to X(\tau)$$

in X - **Mod**(**Set**), which are natural in X in the sense that for all morphisms $f \colon X \to Y$ in S - **DBAlg**, the following square commutes serially.

$$\begin{array}{ccc} X^{(\gamma_1)}(\tau_1) \times \ldots \times X^{(\gamma_p)}(\tau_p) & \xrightarrow{f^{(\gamma_1)}_{\tau_1} \times \ldots \times f^{(\gamma_p)}_{\tau_p}} & Y^{(\gamma_1)}(\tau_1) \times \ldots \times Y^{(\gamma_p)}(\tau_p) \\ {\scriptstyle L^o_X}\downarrow\uparrow{\scriptstyle R^o_X} & & {\scriptstyle L^o_Y}\downarrow\uparrow{\scriptstyle R^o_Y} \\ X(\tau) & \xrightarrow[f]{} & Y(\tau) \end{array}$$

We may also reformulate models:

Definition 9.13. A **modular model** of a modular equational theory (S, T, L, R) is a De Bruijn S-algebra X for which $L^o_X = R^o_X$ for all operations o from T.

Proposition 9.14. *Equational theories and modular equational theories are in one-to-one correspondence, and the model categories coincide on the nose.*

Proof. Straightforward by unfolding definitions. □

10 DE BRUIJN TRANSITION MONADS

In this section, following [Ahrens et al. 2020; Hirschowitz et al. 2020], we extend De Bruijn monads to a notion of programming language, which we then equip with a notion of signature.

10.1 A simple notion of programming language

Let us first introduce our notion of programming language. It is based on two sets $\mathbb{P}$ and $\mathbb{S}$, respectively of **program types** and **state types**.

Proposition 10.1. *For any $\mathbb{S}$-set M and De Bruijn $\mathbb{P}$-monad (X, s, v), each $(M \rtimes X)(\sigma)$ forms a (free) X-module valued in $\mathbb{S}$-sets, in the sense of Definition 9.10, with action*

$$\begin{array}{rcl}(M(\sigma) \times [\mathbf{N}, X]) \times [\mathbf{N}, X] & \to & M(\sigma) \times [\mathbf{N}, X] \\ ((m, f), g) & \mapsto & (m, f[g]).\end{array}$$

Definition 10.2. A **De Bruijn transition monad** consists of

- a De Bruijn $\mathbb{P}$-monad X,
- $\mathbb{S}$-set valued X-modules M_1, M_2, and R, and
- an X-module morphism $R \to M_1 \times M_2$.

Example 10.3. Taking $\mathbb{P} = \mathbb{S}$ to be the set of simple types over a fixed set of base types, we can take

- $X(\tau)$ to consist of simply-typed values of type τ; this is a monad because values are stable under value substitution;
- $M_1(\tau)$ to consist of general terms, which are also stable under value substitution;
- $M_2 = X$; and
- $R(\tau)$ to consist of all big-step evaluation proofs between a term and a value of type τ, which are again stable under value substitution.

Of course the X-module morphism $R \to M_1 \times M_2$ maps any evaluation proof to its source and target.

10.2 Registers of signatures

In Hirschowitz et al. [2020], a general, very simple framework for notions of signatures is introduced. Let us briefly recall it here.

Definition 10.4.

- A **semantic signature** over a category $\mathbf{C}$ is a category $\mathbf{E}$ with initial object, equipped with a **forgetful** functor to $\mathbf{C}$.
- The class of semantic signatures over a category $\mathbf{C}$ is denoted by $\mathbf{UR}_{\mathbf{C}}$.
- A **register** R over $\mathbf{C}$ consists of
  - a set $\mathbf{Sig}_R$ of **signatures**, together with
  - a **semantics** map $\mathbf{sem}_R \colon \mathbf{Sig}_R \to \mathbf{UR}_{\mathbf{C}}$ sending each signature to a semantic signature.

Definition 10.5. If $\mathbf{sem}_R(S)$ is the forgetful functor $U \colon \mathbf{E} \to \mathbf{C}$, we denote the initial object of $\mathbf{E}$ by $S^{\circledast}$, and $U(S^{\circledast})$ by $S^{\star}$. Thus, the former is the initial model, while the latter is its "carrier", an object of $\mathbf{C}$.

We have already seen a register; let us make this official:

Definition 10.6. The register $\mathbf{ET}(\mathbb{T})$ has

Signatures De Bruijn equational theories with types in $\mathbb{T}$ (§9) as signatures, and
Semantics its semantics map sends any E to E-**DBAlg**, with the obvious forgetful functor to **DBMnd**($\mathbb{T}$).

Validity proof: By Theorem 9.6. □

In the following sections, we design a register for De Bruijn transition monads, whose signatures will allow us to efficiently specify relevant programming languages.

A signature will consist of four components, from three different registers:

- a signature E for a De Bruijn monad (E will be a De Bruijn equational theory);
- signatures E_1 and E_2 for $E^\star$-modules; and
- a signature E_R for an object of $E^\star$ - $\mathbf{Mod}_f(\mathbf{Set}^{\mathbb{S}})/E_1^\star \times E_2^\star$.

For the first register, we take as signatures De Bruijn equational theories, the semantics of any E being E - **DBAlg**, as expected.

10.3 Signatures for modules over a De Bruijn monad

Definition 10.7. Let $\mathbf{Facet} = (\mathbb{P} + \mathbb{S}) \times \mathbb{P}^*$ denote the set of pairs $\varphi = (\zeta, (p_1, \ldots, p_n))$ with ζ a program or state type, and $p_1, \ldots, p_n$ a sequence of program types.

Definition 10.8. A **module arity** is a pair of a facet and a state type, i.e., an element of $\mathbf{Facet}^* \times \mathbb{S}$.

Notation 9.

- *For any state type $\sigma \in \mathbb{S}$, we denote by $\Theta_\sigma^{(p_1,\ldots,p_n)}$ the facet $(\sigma, (p_1, \ldots, p_n))$.*
- *For any program type $p \in \mathbb{P}$, we denote by $\mathbf{I}_p^{(p_1,\ldots,p_n)}$ the facet $(p, (p_1, \ldots, p_n))$.*
- *We omit the parentheses when $p_1, \ldots, p_n$ is the empty sequence.*
- *We write $\varphi_1 \times \ldots \times \varphi_n \to \Theta_\sigma$ for the module arity $((\varphi_1, \ldots, \varphi_n), \sigma)$.*

Definition 10.9 (Binding signatures for X-modules). Given a De Bruijn monad X, an X-**binding signature** consists of a set O of **operations**, equipped with an **arity** map $ar\colon O \to \mathbf{Facet}^* \times \mathbb{S}$.

Example 10.10. Recalling the value monad of §8.5, say $\mathscr{L}_v$, we would like to view the set of all terms as an $\mathscr{L}_v$-module. For this, we introduce the $\mathscr{L}_v$-binding signature S_{app} with operations

$$\text{val}_\tau\colon \mathbf{I}_\tau \to \Theta_\tau \qquad \text{and} \qquad \text{app}_{\sigma,\tau}\colon \Theta_{\sigma\to\tau} \times \Theta_\sigma \to \Theta_\tau,$$

for all types σ and τ. We will see below that this signature specifies the family of simply-typed terms, viewed as an $\mathscr{L}_v$-module.

Definition 10.11 (Models).

- We define the interpretation $\Sigma_\varphi^X(M)$ of any facet φ on any X-module M over any De Bruijn monad X by:
  - $\Sigma^X_{\mathbf{I}_p^{(p_1,\ldots,p_n)}}(M) = X_p^{(p_1,\ldots,p_n)}$, and
  - $\Sigma^X_{\Theta_\sigma^{(p_1,\ldots,p_n)}}(M) = M_\sigma^{(p_1,\ldots,p_n)}$.
- Given an X-binding signature $S = (O, ar)$, an S-**algebra** is an $\mathbb{S}$-set valued X-module M equipped with a morphism $o_M\colon \Sigma^X_{\varphi_1}(M) \times \ldots \times \Sigma^X_{\varphi_n}(M) \to M_\sigma$, for each $o \in O$, where $ar(o) = ((\varphi_1, \ldots, \varphi_n), \sigma)$
- An S-algebra morphism is an X-module morphism which commutes with operations.
- Let S - **alg** denote the category of S-algebras and morphisms between them.

We may at last introduce our register for modules over a given De Bruijn monad.

Definition 10.12. The register $\mathbf{ModReg}(X)$ has

Signatures as signatures all X-binding signatures, and
Semantics the semantics of any X-binding signature S is S - **alg** (over X - **Mod**).

Validity proof: The category X - **Mod** is the category of algebras for the monad $M \mapsto M \otimes X$ on sets. Since the monad is $\aleph_1$-accessible, X - **Mod** is locally $\aleph_1$-presentable. Furthermore, for any X-binding signature S, S - **alg** is the category of algebras for the induced endofunctor Σ_S^X on X - **Mod**, whose definition should be clear from Definition 10.11. This endofunctor being finitary, the forgetful functor S - **alg** $\to X$ - **Mod** admits a left adjoint, and hence an inital object, as desired. □

Let us now characterise the initial model of an X-binding signature.

Definition 10.13.

- The functor $F_\varphi^X : \mathbf{Set} \to \mathbf{Set}$ induced by any facet φ is defined by
  - $F^X_{\mathbf{I}_p^{(p_1,\dots,p_n)}}(M) = X_p$ and
  - $F^X_{\Theta_\sigma^{(p_1,\dots,p_n)}}(M) = M_\sigma$.
- The functor $F^X_{(\varphi_1,\dots,\varphi_p),\sigma} : \mathbf{Set} \to \mathbf{Set}$ induced by any module arity $((\varphi_1, \dots, \varphi_p), \sigma)$ is defined by
  - $F^X_{(\varphi_1,\dots,\varphi_p),\sigma}(M)(\sigma) = F^X_{\varphi_1}(M) \times \dots \times F^X_{\varphi_p}(M)$ and
  - $F^X_{(\varphi_1,\dots,\varphi_p),\sigma}(M)(\sigma') = \emptyset$ if $\sigma' \neq \sigma$.
- The functor $F_S^X : \mathbf{Set} \to \mathbf{Set}$ induced by any X-binding signature $S = (O, ar)$ is

$$F_S^X(M) = \sum_{o \in O} F^X_{ar(o)}(M).$$

Proposition 10.14. *Consider any X-binding signature S.*

- *The initial F_S^X-algebra possesses a unique X-module structure turning it into an S-algebra.*
- *This S-algebra is initial.*

Example 10.15. In Example 10.10, we announced that the signature S_{app} specifies the family of all terms as an $\mathscr{L}_v$-module. Indeed, the induced endofunctor on sets is determined by:

$$\Sigma^{\mathscr{L}_v}_{S_{\mathrm{app}}}(M)(\sigma) = \mathscr{L}_v(\sigma) + \sum_\tau M(\tau \to \sigma) \times M(\tau).$$

By Proposition 10.14, the initial S_{app}-algebra thus has as carrier the family of application binary trees with leaves in values, which is in one-to-one correspondence with that of terms.

10.4 Signatures for transition rules

We have defined the first two registers announced in the end of §10.2. Let us now introduce the last one. For this, let us fix a De Bruijn monad X, together with X-modules M_1 and M_2. We wish to design a register for X - **Mod**$/M_1 \times M_2$.

The idea here is to model a transition rule as follows:

- organise the "metavariables" into a set-valued X-module V,
- describe the conclusion as a module morphism $V \to (M_1 \times M_2)_\tau$, for some $\tau \in \mathbb{S}$, and
- describe each premise as a module morphism of the form $V \to (M_1 \times M_2)_\sigma^{(\gamma)}$, for a suitable $(\gamma, \sigma) \in \mathbb{T}^* \times \mathbb{T}$.

Let us first follow this procedure on an example, the left application rule in simply-typed λ-calculus:

$$\frac{e_1 \to e_1'}{e_1\ e_2 \to e_1'\ e_2}\ (\textsc{AppL}).$$

We first of all recall the relevant De Bruijn monad Λ_{ST} from Example 8.6, and take $M_1 = M_2 = \Lambda_{\mathrm{ST}}$, since reduction in the simply-typed λ-calculus relates mere terms. Now for the rule, it is implicitly parameterised over the (related) types of e_1 (and e_1') and e_2. Let us fix them by assuming $e_1, e_1' : \sigma \to \tau$ and $e_2 : \sigma$, potentially with other free variables. Accordingly, we take $V = \Lambda_{\mathrm{ST}}(\sigma \to \tau)^2 \times \Lambda_{\mathrm{ST}}(\sigma)$. We then describe the conclusion and premise.

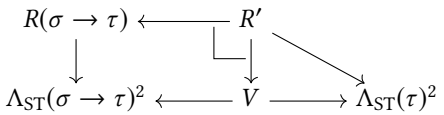
- The conclusion is the morphism
$$V \to (\Lambda_{\mathrm{ST}}(\sigma \to \tau) \times \Lambda_{\mathrm{ST}}(\sigma))^2 \xrightarrow{\mathrm{app}_{\sigma,\tau} \times \mathrm{app}_{\sigma,\tau}} \Lambda_{\mathrm{ST}}(\tau)^2.$$
- The premise is the projection $V \to \Lambda_{\mathrm{ST}}(\sigma \to \tau)^2$.

It is probably clear to the reader that the morphisms
$$\Lambda_{\mathrm{ST}}(\sigma \to \tau)^2 \leftarrow V \to \Lambda_{\mathrm{ST}}(\tau)^2$$
we just defined are related to the (AppL) rule, but maybe not exactly how. Let us now explain this.

The point is that the morphisms induce an endofunctor Σ_{AppL} on Λ_{ST} - **Mod**$/\Lambda_{\mathrm{ST}}^2$, given for any object $R \to \Lambda_{\mathrm{ST}}^2$ over Λ_{ST}^2 by

- evaluating at $\sigma \to \tau$ to get a morphism $R(\sigma \to \tau) \to \Lambda_{\mathrm{ST}}(\sigma \to \tau)^2$;
- pulling back the obtained morphism along the premise morphism $V \to \Lambda_{\mathrm{ST}}(\sigma \to \tau)^2$ to get a morphism $R' \to V$; and finally
- post-composing with the conclusion morphism $V \to \Lambda_{\mathrm{ST}}(\tau)^2$ to get a morphism $R' \to \Lambda_{\mathrm{ST}}(\tau)^2$,

all as in the following diagram.



The intuition is that R' is the set of instances of the premise, each instance lying over the potential conclusion in $\Lambda_{\mathrm{ST}}(\tau)^2$.

A model of the rule is thus an $R \to \Lambda_{\mathrm{ST}}^2$ equipped with a map $R' \to R$ over Λ_{ST}^2, providing a witness that for each instance of the premise the corresponding conclusion is satisfied.

Equivalently, it is Σ_{AppL}-algebra structure.

Let us now formalise all this. We first introduce a notion of rule, then define the endofunctor induced by a rule, and finally our register for transition relations, whose signatures are families of rules.

Definition 10.16. Given a De Bruijn monad X and $\mathbb{S}$-valued X-modules M_1 and M_2, a **rule** over (X, M_1, M_2) consists of

- a **conclusion type** $\tau \in \mathbb{S}$,
- a set-valued X-module V of **metavariables**,
- a **conclusion** morphism $C : V \to (M_1 \times M_2)_\tau$,
- a finite set I of **premise indices**,
- for each $i \in I$, a sequent $(\gamma_i, \tau_i) \in \mathbb{T}^* \times \mathbb{S}$, together with a **premise** morphism $P_i : V \to (M_1 \times M_2)_{\tau_i}^{(\gamma_i)}$.

Definition 10.17. The endofunctor $\Sigma_\rho : X$ - **Mod**$/M_1 \times M_2 \to X$ - **Mod**$/M_1 \times M_2$ induced by a rule ρ as in Definition 10.16 maps any $d : R \to M_1 \times M_2$

- first to $\prod_i R_{\tau_i}^{(\gamma_i)} \to \prod_i (M_1 \times M_2)_{\tau_i}^{(\gamma_i)}$,
- which it then pulls back along $\langle P_i \rangle_i : V \to \prod_i (M_1 \times M_2)_{\tau_i}^{(\gamma_i)}$,

- and postcomposes with $C\colon V \to (M_1 \times M_2)_\tau$,
- before at last embedding this set-valued module morphism into a $\mathbb{S}$-set valued one, mapping $R' \to (M_1 \times M_2)_\tau$ to $R' \cdot \mathbf{y}_\tau \to M_1 \times M_2$, where

$$\begin{aligned} R' \cdot \mathbf{y}_\tau(\tau) &= R' \\ R' \cdot \mathbf{y}_\tau(\sigma) &= \emptyset \quad (\text{when } \sigma \neq \tau). \end{aligned}$$

We may now define our register for transition rules:

Definition 10.18. The register $\mathbf{TR}(X, M_1, M_2)$ has

Signatures as signatures all families $(\rho_j)_{j\in J}$ of rules over (X, M_1, M_2), and

Semantics the semantics of a signature $(\rho_j)_{j\in J}$ is the category $(\sum_j \Sigma_{\rho_j})$ - **alg** of algebras for the induced endofunctor on X - $\mathbf{Mod}/M_1 \times M_2$.

Let us conclude with a last rule example, illustrating the possibility of deriving in the premises of a rule.

Example 10.19. Let us now consider the ξ-rule in simply-typed λ-calculus:

$$\frac{e \to e'}{\lambda x.e \to \lambda x.e'}\ (\xi)\cdot$$

As before, the relevant De Bruijn monad is Λ_{ST}, and $M_1 = M_2 = \Lambda_{\mathrm{ST}}$. Furthermore, the rule is implicitly parameterised over the types of e (and e') and x. Let us fix them by assuming $x : \sigma \vdash e, e' : \tau$, potentially with other free variables. Accordingly, we take $V = \Lambda^{(\sigma)}_{\mathrm{ST}}(\tau)^2$. We then describe the conclusion and premise.

- The conclusion is the morphism

$$V = \Lambda^{(\sigma)}_{\mathrm{ST}}(\tau)^2 \xrightarrow{\mathrm{lam}_{\sigma,\tau} \times \mathrm{lam}_{\sigma,\tau}} \Lambda_{\mathrm{ST}}(\sigma \to \tau)^2.$$

- The premise is the identity morphism on V, since it precisely relates the two available metavariables.

10.5 Signatures for De Bruijn transition monads

Let us now wrap up and define our register for De Bruijn transition monads.

Definition 10.20. The register **TM** is defined as follows.

Signatures Signatures are all tuples (E, S_1, S_2, T), where
- E is a De Bruijn equational theory, specifying a De Bruijn monad $X := E^\star$,
- S_1 and S_2 are X-binding signatures, specifying X-modules $M_i = S_i^\star$, and
- T is a signature in $\mathbf{TR}(X, M_1, M_2)$, i.e., a family of rules.

Semantics The semantic signature associated with any signature (E, S_1, S_2, T) as above is the category of Σ_T-algebras, i.e., $\mathbf{sem}(T)$, with its forgetful functor to X - $\mathbf{Mod}/M_1 \times M_2$.

11 CONCLUSION AND PERSPECTIVES

We have proposed a simple, set-based theory of syntax with variable binding, which associates a notion of model (or algebra) to each binding signature, and constructs a term model following De Bruijn representation. The notion of model features a substitution operation. We have experienced the simplicity of this theory by implementing it in both Coq and HOL Light.

We have furthermore equipped the construction with an initial-algebra semantics, organising the models of any binding signature into a category, and proving that the term model is initial therein.

We hope that this will be useful, first because it provides an induction principle for reasoning on the term model, and also as a framework for operational and denotational semantics.

We have then studied this initial-algebra semantics in a bit more depth, in two directions:

- We have first established a formal link with the mainstream, presheaf-based approach [Fiore et al. 1999], proving that well-behaved models (in a suitable sense on each side of the correspondence) agree up to an equivalence of categories.
- We have then recast the whole initial-algebra semantics into two established, abstract frameworks for syntax with variable binding, one based on strengths [Fiore 2008; Fiore et al. 1999], the other on modules [Hirschowitz and Maggesi 2007, 2010].

Finally, we have shown that our theory extends easily to a simply-typed setting, and smoothly incorporates equations and transitions.

A PROOF SKETCH OF THEOREM 5.22

In this section, we provide a sketch of the proof that the categories Φ_S - $\mathbf{Mon}_{wb}$ and S - $\mathbf{DBAlg}_{wb}$ are equivalent, for any binding signature S.

We fix such a signature S for the whole section.

Let us start bydefining functors $\mathbf{DBMnd} \to [\mathbb{F}, \mathbf{Set}]$ and $[\mathbb{F}, \mathbf{Set}] \to \mathbf{Set}$.

Definition A.1. For any De Bruijn monad (X, s, v), let $X' \colon \mathbb{F} \to \mathbf{Set}$ map any n to the set of $x \in X$ with support n.

Definition A.2. Let $(-)(\mathbb{N}) \colon [\mathbb{F}, \mathbf{Set}] \to \mathbf{Set}$ map any $F \colon \mathbb{F} \to \mathbf{Set}$ to the coend $\int^n F(n) \times [n, \mathbb{N}]$, whose elements we write $x[\![\sigma]\!]$, for all $x \in F(n)$ and $\sigma \colon n \to \mathbb{N}$. These are equivalence classes of pairs (x, σ) under the equivalence relation generated by $(x, \sigma \circ f) \sim (f \cdot x, \sigma)$, for all $f \colon n \to p$, $x \in F(n)$, and $\sigma \colon p \to \mathbb{N}$.

Remark 16. *Through the equivalence* $[\mathbb{F}, \mathbf{Set}] \simeq [\mathbf{Set}, \mathbf{Set}]_f$, $X(\mathbb{N})$ *really corresponds to evaluation at* $\mathbb{N}$.

In the following, we use an alternative definition for $F(\mathbb{N})$.

Proposition A.3. *For all* $F \colon \mathbb{F} \to \mathbf{Set}$, *the set-cocone from*

$$F(0) \xhookrightarrow{\subseteq} F(1) \ldots \xhookrightarrow{\subseteq} F(n) \xhookrightarrow{\subseteq} F(n+1) \ldots$$

into $F(\mathbb{N})$ *is colimiting. Otherwise said* $F(\mathbb{N})$ *is isomorphic to the quotient of* $\sum_n F(n)$ *induced by renaming along inclusions, i.e.,* $(n, x) \sim (n+1, in_1 \cdot x)$.

Proof. The functor $\omega \to \mathbb{F}/\mathbb{N}$ mapping any n to the inclusion $n \subseteq \mathbb{N}$ is cofinal. □

Lemma A.4. *If* $F : \mathbb{F} \to \mathbf{Set}$ *is well-behaved, the canonical map* $F(n) \to F(\mathbb{N})$ *is injective for any* $n \in \mathbb{N}$.

Proof. This will follow from the fact that such a functor preserves injections, for then all maps in the set-cocone of Proposition A.3 are injections. If the domain of an injection $n \hookrightarrow m$ is not empty, then it has a retract and thus its image is automatically injective. Now, the image of an initial morphism $F(0) \to F(n)$ with $n > 0$ is also injective, as the pullback of $F(1) \hookrightarrow F(n+1)$ along $F(n) \hookrightarrow F(n+1)$, since F is well-behaved. □

The forgetful functor $[\mathbf{B}(\mathbb{N}), \mathbf{Set}] \to \mathbf{Set}$ is clearly monadic, hence an isofibration. By lifting along the bijection $F(\mathbb{N}) \cong \sum_n F(n)/\sim$ of Proposition A.3, we thus obtain an action of $\mathbf{B}(\mathbb{N})$ on $\sum_n F(n)/\sim$. Let us explicitly describe this.

Definition A.5. For any $f : \mathbb{N} \to \mathbb{N}$, let $n \xrightarrow{e_f} p \xrightarrow{\subseteq} \mathbb{N}$ denote the factorisation of the composite $n \xrightarrow{\subseteq} \mathbb{N} \xrightarrow{f} \mathbb{N}$ as an epi followed by an inclusion.

Proposition A.6. *The action of* $\mathbf{B}(\mathbb{N})$ *on* $\sum_n F(n)/\sim$ *induced by the bijection* $F(\mathbb{N}) \cong \sum_n F(n)/\sim$ *is such that, for any* $x \in F(n)$*, we have* $f \cdot [x] = [e_f \cdot x]$.

Proof. Denoting by $j_n : n \to \mathbb{N}$ the inclusion, the bijection maps any $x \in F(n)$ to $x(\!|j_n|\!)$. Thus, $f \cdot x$ must be the unique antecedent of

$$f \cdot (x(\!|j_n|\!)) = x(\!|f \circ j_n|\!) = x(\!|j_p \circ e_f|\!) = (e_f \cdot x)(\!|j_p|\!),$$

as desired. □

Our next step is to move, on the presheaf side, from monoids in $[\mathbb{F}, \mathbf{Set}]$ to the more convenient relative monads:

Notation 10. *Let* **RelMnd** *denote the category of monads relative to the canonical embedding* $\mathbb{F} \to \mathbf{Set}$*, thereafter merely called* relative monads*. Moreover, given a relative monad* $(F, \mathrm{bind}, \mathrm{ret})$*, we overload the notation* $-[-]$ *for* $\mathrm{bind} : F(n) \times F(m)^n \to F(m)$.

Proposition A.7. *The category* $\mathbf{Mon}[\mathbb{F}, \mathbf{Set}]$ *of monoids in* $[\mathbb{F}, \mathbf{Set}]$ *is isomorphic to the category* **RelMnd** *over* $[\mathbb{F}, \mathbf{Set}]$.

Proof. By [Altenkirch et al. 2015, §3.3]. □

Definition A.8. A relative monad is **well-behaved** when its underlying functor is.

Proposition A.9. *The functors* $X \mapsto X'$ *and* $F \mapsto F(\mathbb{N})$ *lift to functors*

$$\mathbf{DBMnd} \underset{(-)(\mathbb{N})}{\overset{(-)'}{\rightleftarrows}} \mathbf{RelMnd},$$

which restrict to an equivalence

$$\mathbf{DBMnd}_{wb} \underset{(-)(\mathbb{N})}{\overset{(-)'}{\rightleftarrows}} \mathbf{RelMnd}_{wb}$$

on well-behaved objects.

Lemma A.10. *Let* (X, s, v) *be any De Bruijn monad,* $e \in X'(p)$*, and* $\sigma : \mathbb{N} \to X$ *such that, for all* $i \in p$*,* $\sigma(i)$ *has support* N*. Then* $e[\sigma]$ *has support* N.

Proof. If $N = 0$, then we readily have $e[\sigma][f] = e[\sigma[f]] = e[\sigma]$ as desired. Otherwise, let $f : \mathbb{N} \to \mathbb{N}$ fix N, and let $g : \mathbb{N} \to \mathbb{N}$ be the identity on N and map all $i \geq N$ to 0. We thus have

$$e[\sigma] = e[g][\sigma] = e[\sigma'],$$

where σ', defined by $\sigma'(i < N) = \sigma(i)$ and $\sigma'(i \geq N) = \sigma(0)$, has the property that all $\sigma(i)$ have support N. It then follows that

$$e[\sigma][f] = e[\sigma'][f] = e[\sigma'[f]] = e[\sigma'] = e[\sigma],$$

as desired. □

Lemma A.11. *Let* (X, s, v) *by any De Bruijn monad and* $n \in \mathbb{N}$*. Then* $v(n)$ *has support* $n + 1$.

Proof. Let f fix $n + 1$. Then $v(n)[v \circ f] = v(f(n)) = v(n)$, as desired. □

Lemma A.12. *For any functor $F\colon \mathbb{F} \to \mathbf{Set}$, $p \in \mathbb{F}$, and map $(e_1, \dots, e_p) \in F(\mathbb{N})^p$, there exists n, $(\check{e}_1, \dots, \check{e}_p) \in F(n)^p$, and $f : n \to \mathbb{N}$ such that, $e_i \sim \check{e}_i$.*

Proof. Let each $e_i \in F(n_i)$. Taking n as the maximum among n_i, define $\check{e}_i$ as the weakening of e_i. □

Proof sketch for Proposition A.9. To see that $(-)'$ lifts to De Bruijn monads, consider any De Bruijn monad (X, s, v), and let $-[-]\colon X'(p) \times X'(n)^p \to X'(n)$ map any $x \in X'(p)$ and $\sigma\colon p \to X'(n)$ to $x[\bar{\sigma}]$, where $\bar{\sigma}(i) = \sigma(i)$ for $i \in p$, and $\bar{\sigma}(j) = v(0)$ for $j \notin p$. Similarly, let $\eta_n : n \to X'(n)$ map any $i \in n$ to $v(i)$. (This is well-defined by Lemmas A.10 and A.11.) This yields relative monad structure on X', by associativity and unitality, as desired.

Conversely, if T is a relative monad, we define a substitution map $T(\mathbb{N}) \times T(\mathbb{N})^{\mathbb{N}} \to T(\mathbb{N})$ as follows. For any $e \in T(\mathbb{N})$, say with $e \in T(p)$, and $\sigma\colon \mathbb{N} \to T(\mathbb{N})$, we consider the tuple $(\sigma_1, \dots, \sigma_p) \in T(q)^p$ (by Lemma A.12). We finally define $e[\sigma] := e[\sigma_i]_{i \in p}$, which we check is compatible with the quotient on $T(\mathbb{N})$.

This shows the desired lifting. For the equivalence, for any well-behaved De Bruijn monad (X, s, v), $X'(\mathbb{N})$ is the disjoint union of elements of X for each possible support $n \in \mathbb{N}$, quotiented by the inclusion relation. Now, if x has support n, it also has support m for any $m > n$, so x would appear multiple times in this disjoint union. Such redundancies are eliminated by the quotient, and we finally obtain a set isomorphic to X. It is straightforward to check that this isomorphism lifts to an isomorphism of De Bruijn monads.

Conversely, given a well-behaved relative monad $(F, \mathrm{bind}, \mathrm{ret})$, $F(\mathbb{N})'(n)$ is the subset of $F(\mathbb{N})$ consisting of elements $x \in F(m)$ such that given any $f\colon \mathbb{N} \to F(\mathbb{N})$ such that $f_i = \mathrm{ret}(i)$ for any $i < n$, $x[f_1, \dots, f_m] \sim x$, assuming $f_1, \dots, f_m \in F(p)$ by Lemma A.12.

Clearly, we can map any $x \in F(n)$ to itself, as an element of $F(\mathbb{N})'(n)$. This is in fact an injection, by Lemma A.4.

Let us show that this map is also surjective. Let $x \in F(\mathbb{N})'(n)$, as an element of $F(m)$ satisfying the above condition. Surjectivity will follow if we can construct a map $f\colon \mathbb{N} \to F(\mathbb{N})$ such that $f_1, \dots, f_m \in F(n)$ and $f_i = \mathrm{ret}(i)$ for any $i < n$; for then, by assumption, $x \sim x[f_1, \dots, f_{m-1}]$, with $x[f_1, \dots, f_{m-1}] \in F(n)$ the desired antecedent.

The construction of such a map f will be straightforward if we show that $F(n)$ is not empty, say with $u \in F(n)$, for then we may take $f(i) = \mathrm{ret}(i)$ for any $i < n$, and $f(i) = u$ for $i \geq n$. It thus suffices to show that $F(n)$ is not empty, which we do by case analysis on n.

- If $n > 0$, then at least $\mathrm{ret}(0) \in F(n)$.
- If $n = 0$, x belongs to the intersection of $F(m) \hookrightarrow F(m+m) \hookleftarrow F(m)$, since $x[in_l] \sim x \sim x[in_r]$ necessarily implies that $x[in_l] = x[in_r]$, as $F(m + m) \hookrightarrow F(\mathbb{N})$ is injective by Lemma A.4. Because F is well-behaved, we thus get an element in $F(0)$, as desired.

Finally, it can be shown that the pointwise isomorphism $F(\mathbb{N})'(n) \cong F(n)$ lifts to an isomorphism of relative monads. □

Definition A.13. A *relative module M over a relative monad* $(F, \mathrm{bind}_F, \mathrm{ret}_F)$ is a functor $M : \mathrm{Kl}(F) \to \mathbf{Set}$, where $\mathrm{Kl}(F)$ is the Kleisli category of F (see [Altenkirch et al. 2015]). We denote the action on morphisms $M(n) \times F(m)^n \to M(m)$ by $-[-]$.

Remark 17. Kl *extends to a functor* $\mathbf{RelMnd} \to \mathbf{CAT}$.

Notation 11. *We denote by* $\mathbf{RelMod}$ *the functor* $[\mathrm{Kl}, \mathbf{Set}] : \mathbf{RelMnd}^o \to \mathbf{CAT}$*, so that* $\mathbf{RelMod}(F)$ *is the category of relative modules over F. We denote by* $\int \mathbf{RelMod} \to \mathbf{RelMnd}$ *the induced fibration: an object of* $\int \mathbf{RelMod}$ *consists of pairs of a relative monad and a relative module over it.*

Similarly, we consider the fibration $\int \mathbf{DBMod} \to \mathbf{DBMnd}$ *of De Bruijn modules (defined, for example, as in Definition 7.1) over De Bruijn monads: an object of* $\int \mathbf{DBMod}$ *is a De Bruijn monad* X *together with an* X*-module.*

Both of these fibrations have a section, which we both denote by Θ *by abuse of notation, mapping a monad to itself, as a module over itself.*

We have the same result for modules.

Proposition A.14. *The functors* $X \mapsto X'$ *and* $F \mapsto F(\mathbb{N})$ *lift to functors*

$$\begin{array}{ccc} \int \mathbf{DBMod} & \underset{(-)(\mathbb{N})}{\overset{(-)'}{\rightleftarrows}} & \int \mathbf{RelMod} \\ \downarrow & & \downarrow \\ \mathbf{DBMnd} & \underset{(-)(\mathbb{N})}{\overset{(-)'}{\rightleftarrows}} & \mathbf{RelMnd}, \end{array}$$

which restrict to an equivalence

$$\begin{array}{ccc} \int \mathbf{DBMod}_{wb} & \underset{(-)(\mathbb{N})}{\overset{(-)'}{\rightleftarrows}} & \int \mathbf{RelMod}_{wb} \\ \downarrow & & \downarrow \\ \mathbf{DBMnd}_{wb} & \underset{(-)(\mathbb{N})}{\overset{(-)'}{\rightleftarrows}} & \mathbf{RelMnd}_{wb} \end{array}$$

on well-behaved objects.

Proof. Similar to Proposition A.9. □

Finally, let us show that this last equivalence lifts to an equivalence between models of S. Let us first recall that, for any functor $J\colon \mathbf{F} \to \mathbf{C}$, monads relative to J are the same as monoids in the induced skew monoidal category $[\mathbf{F}, \mathbf{C}]$.

Definition A.15. For any functor $J\colon \mathbf{F} \to \mathbf{C}$, any structurally strong endofunctor Σ on the induced skew monoidal category $[\mathbf{F}, \mathbf{C}]$, let $\bar{\Sigma}\colon \mathbf{Mon}[\mathbf{F}, \mathbf{C}] \to \mathbf{Mod}[\mathbf{F}, \mathbf{C}]$ denote the section of the forgetful functor mapping any relative monad T to $(T, \Sigma(T))$, with module structure given by the composite $\Sigma(T) \otimes T \to \Sigma(T \otimes T) \to \Sigma(T)$.

Proposition A.16. *For any functor* $J\colon \mathbf{F} \to \mathbf{C}$*, any structurally strong endofunctor* Σ *on* $[\mathbf{F}, \mathbf{C}]$*,* Σ *-* **Mon** *is the* ***inserter*** *[Kelly 1989] of the diagram*

$$\mathbf{Mon}[\mathbf{F}, \mathbf{C}] \underset{\Theta}{\overset{\bar{\Sigma}}{\rightrightarrows}} \mathbf{Mod}[\mathbf{F}, \mathbf{C}] \leftarrow \mathbf{Mon}[\mathbf{F}, \mathbf{C}]$$

in $\mathbf{CAT}/\mathbf{Mon}[\mathbf{F}, \mathbf{C}]$.

Proof. By mere definition unfolding. □

Lemma A.17. *For category* $\mathbf{C}$ *of one the forms* $[\mathbb{F}, \mathbf{Set}]$, $\mathbf{RelMod}(T)$, *and* $\mathbf{DBMod}(X)$ *with* T *and* X *well-behaved, the embedding* $\mathbf{C}_{wb} \hookrightarrow \mathbf{C}$ *creates coproducts, finite products, and derivatives.*

Proof. In each case, it suffices to show that the relevant object in $\mathbf{C}$ lies in $\mathbf{C}_{wb}$, by full faithfulness.

For $[\mathbb{F}, \mathbf{Set}]$, well-behaved objects are equivalently algebras for the monad induced by restriction and right Kan extension along $\mathbb{F}^\bullet \to \mathbb{F}$, where $\mathbb{F}^\bullet$ denotes $\mathbb{F}$ without $\emptyset$. Thus, first, the embedding creates all limits, hence in particular finite products. Since right Kan extension is computed by a connected limit, it commutes with coproducts, so the monad preserves coproducts, and the forgetful functor creates them. Finally, we need to show that F' is well-behaved, i.e., that $F'(\emptyset) \cong F'(1) \times_{F'(2)} F'(1)$, which in fact holds because $F'(1) \cong F'(2) \times_{F'(3)} F'(2)$ by Proposition 5.12.

For $\mathbf{RelMod}(T)$, the result follows from the fact that all of these objects are created by the forgetful functor to $[\mathbb{F}, \mathbf{Set}]$.

For $\mathbf{DBMod}(X)$, the result on coproducts and finite products follows straightforwardly from the fact that both are created by the forgetful functor to sets. For derivatives, we need to show that for any well-behaved X-module M, for all $a \in M$ and $\sigma \colon \mathbb{N} \to X$, $a[\Uparrow \sigma]$ has finite support. Let a have support n and each $\sigma(i)$ have support n_i. Then each $(\Uparrow \sigma)(i)$ has support $n_i + 1$, and the result follows by Lemma A.10. □

Proposition A.18. *The following square commutes serially up to natural isomorphism.*

$$\begin{array}{ccc} \mathbf{RelMnd}_{wb} & \xrightarrow{(-)(\mathbb{N})} & \mathbf{DBMnd}_{wb} \\ {\scriptstyle \overline{\Phi_S}}\downarrow\Theta & & {\scriptstyle \overline{\Sigma_S}}\downarrow\Theta \\ \int \mathbf{RelMod}_{wb} & \xrightarrow[(-)(\mathbb{N})]{} & \int \mathbf{DBMod}_{wb} \end{array}$$

Proof. Both $\overline{\Phi_S}$ and $\overline{\Sigma_S}$ map their argument to a coproduct of products of derivatives, which are all preserved by equivalences. □

Proof of Theorem 5.22. By Proposition A.16 and interchange of limits, S - $\mathbf{DBAlg}_{wb}$ is the inserter of a diagram in $\mathbf{CAT}/\mathbf{DBMnd}_{wb}$, which is mapped by the 2-equivalence $\mathbf{CAT}/\mathbf{DBMnd}_{wb} \simeq \mathbf{CAT}/\mathbf{RelMnd}_{wb}$ to a diagram whose inserter is Φ_S - $\mathbf{Mon}_{wb}$ by Proposition A.14. Finally, because 2-equivalences preserve 2-limits, we get S - $\mathbf{DBAlg} \simeq \Phi_S$ - $\mathbf{Mon}$ as desired. □

REFERENCES

Martín Abadi, Luca Cardelli, Pierre-Louis Curien, and Jean-Jacques Lévy. 1990. Explicit Substitutions. In *Proc. 17th International Symposium on Principles of Programming Languages*, Frances E. Allen (Ed.). ACM, 31–46. https://doi.org/10.1145/96709.96712

Jirí Adámek, Stefan Milius, Nathan J. Bowler, and Paul Blain Levy. 2012. Coproducts of Monads on Set. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*. IEEE Computer Society, 45–54. https://doi.org/10.1109/LICS.2012.16

J. Adámek and J. Rosicky. 1994. *Locally Presentable and Accessible Categories*. Cambridge University Press. https://doi.org/10.1017/CBO9780511600579

Benedikt Ahrens, André Hirschowitz, Ambroise Lafont, and Marco Maggesi. 2020. Reduction monads and their signatures. *PACMPL* 4, POPL (2020), 31:1–31:29. https://doi.org/10.1145/3371099

Guillaume Allais, Robert Atkey, James Chapman, Conor McBride, and James McKinna. 2018. A type and scope safe universe of syntaxes with binding: their semantics and proofs. *PACM on Programming Languages* 2, ICFP (2018), 90:1–90:30. https://doi.org/10.1145/3236785

Thorsten Altenkirch, James Chapman, and Tarmo Uustalu. 2015. Monads need not be endofunctors. *Logical Methods in Computer Science* 11, 1 (2015). https://doi.org/10.2168/LMCS-11(1:3)2015

Nathanael Arkor and Dylan McDermott. 2021. Abstract Clones for Abstract Syntax. In *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference) (LIPIcs, Vol. 195)*, Naoki Kobayashi (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 30:1–30:19. https://doi.org/10.4230/LIPIcs.FSCD.2021.30

Richard S. Bird and Ross Paterson. 1999. De Bruijn notation as a nested datatype. *Journal of Functional Programming* 9, 1 (1999), 77–91. https://doi.org/10.1017/S0956796899003366

Peio Borthelle, Tom Hirschowitz, and Ambroise Lafont. 2020. A Cellular Howe Theorem. In *Proc. 35th ACM/IEEE Symposium on Logic in Computer Science*, Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller (Eds.). ACM. https://doi.org/10.1145/3373718.3394738

N. G. De Bruijn. 1972. Lambda-Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem. *Indagationes Mathematicae* 34 (1972), 381–392.

Marcelo Fiore and Chung-Kil Hur. 2009. On the construction of free algebras for equational systems. *Theoretical Computer Science* 410 (2009), 1704–1729.

Marcelo P. Fiore. 2008. Second-Order and Dependently-Sorted Abstract Syntax. In *LICS*. IEEE, 57–68. https://doi.org/10.1109/LICS.2008.38

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

Marcelo P. Fiore, Gordon D. Plotkin, and Daniele Turi. 1999. Abstract Syntax and Variable Binding. In *Proc. 14th Symposium on Logic in Computer Science* IEEE.

Murdoch J. Gabbay and Andrew M. Pitts. 1999. A New Approach to Abstract Syntax Involving Binders. In *Proc. 14th Symposium on Logic in Computer Science* IEEE.

Lorenzo Gheri and Andrei Popescu. 2020. A Formalized General Theory of Syntax with Bindings: Extended Version. *J. Autom. Reason.* 64, 4 (2020), 641–675. https://doi.org/10.1007/s10817-019-09522-2

André Hirschowitz, Tom Hirschowitz, and Ambroise Lafont. 2020. Modules over Monads and Operational Semantics. In *5th International Conference on Formal Structures for Computation and Deduction, FSCD (LIPIcs, Vol. 167)*, Zena M. Ariola (Ed.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 12:1–12:23. https://doi.org/10.4230/LIPIcs.FSCD.2020.12

André Hirschowitz, Tom Hirschowitz, and Ambroise Lafont. 2021a. Modules over Monads and Operational Semantics. (2021). Submitted expanded version of [Hirschowitz et al. 2020].

André Hirschowitz, Tom Hirschowitz, Ambroise Lafont, and Marco Maggesi. 2021b. Variable binding and substitution for (nameless) dummies. (2021). https://amblafont.github.io/articles/debruijn-extended.pdf Preprint.

André Hirschowitz and Marco Maggesi. 2007. Modules over Monads and Linearity. In *Proc. 14th International Workshop on Logic, Language, Information and Computation (LNCS, Vol. 4576)*. Springer, 218–237. https://doi.org/10.1007/3-540-44802-0_3

André Hirschowitz and Marco Maggesi. 2010. Modules over monads and initial semantics. *Information and Computation* 208, 5 (2010), 545–564. https://doi.org/10.1016/j.ic.2009.07.003

André Hirschowitz and Marco Maggesi. 2012. Nested Abstract Syntax in Coq. *Journal of Automated Reasoning* 49, 3 (2012), 409–426. https://doi.org/10.1007/s10817-010-9207-9

Gérard Huet. 1994. Residual theory in λ-calculus: a formal development. *Journal of Functional Programming* 4, 3 (1994), 371–394. https://doi.org/10.1017/S0956796800001106

G. Maxwell Kelly. 1989. Elementary observations on 2-categorical limits. *Bulletin of the Australian Mathematical Society* 39 (1989), 301–317.

Stephen Lack and Ross Street. 2014. On monads and warpings. *Cahiers de Topologie et Géométrie Différentielle Catégoriques* LV, 4 (2014), 244–266.

Ambroise Lafont. 2021. Initial algebra semantics for de Bruijn monads in Coq. https://github.com/amblafont/binding-debruijn.

Saunders Mac Lane. 1998. *Categories for the Working Mathematician* (2nd ed.). Number 5 in Graduate Texts in Mathematics. Springer.

Marco Maggesi. 2021. Initial algebra semantics for de Bruijn monads in HOL Light. https://github.com/maggesi/dbmonad/tree/master/De_Bruijn.

Gordon Plotkin. 1990. An illative theory of relations. In *Situation Theory and its Applications (CSLI Lecture Notes, 22)*, R. Cooper et al. (Eds.). Stanford University, 133–146.

John Power. 2007. Abstract Syntax: Substitution and Binders: Invited Address. *Electronic Notes in Theoretical Computer Science* 173 (04 2007), 3–16. https://doi.org/10.1016/j.entcs.2007.02.024

Steven Schäfer, Gert Smolka, and Tobias Tebbi. 2015a. Completeness and Decidability of de Bruijn Substitution Algebra in Coq. In *Proc. 4th International Conference on Certified Programs and Proofs*, Xavier Leroy and Alwen Tiu (Eds.). ACM, 67–73. https://doi.org/10.1145/2676724.2693163

Steven Schäfer, Tobias Tebbi, and Gert Smolka. 2015b. Autosubst: Reasoning with de Bruijn Terms and Parallel Substitutions. In *Proc. 6th International Conference on Interactive Theorem Proving (LNCS, Vol. 9236)*, Christian Urban and Xingyuan Zhang (Eds.). Springer, 359–374. https://doi.org/10.1007/978-3-319-22102-1_24

Kathrin Stark, Steven Schäfer, and Jonas Kaiser. 2019. Autosubst 2: reasoning with multi-sorted de Bruijn terms and vector substitutions. In *Proc. 8th International Conference on Certified Programs and Proofs*, Assia Mahboubi and Magnus O.

Myreen (Eds.). ACM, 166–180. https://doi.org/10.1145/3293880.3294101

Kornel Szlachányi. 2012. Skew-monoidal categories and bialgebroids. *Advances in Mathematics* 231 (2012), 1694–1730. https://doi.org/10.1016/j.aim.2012.06.027

Věra Trnková. 1969. Some properties of set functors. *Commentationes Mathematicæ Universitatis Carolinæ* 10, 2 (1969), 323–352.