HAL open science

Variable binding and substitution for (nameless) dummies

André Hirschowitz, Tom Hirschowitz, Ambroise Lafont, Marco Maggesi

To cite this version:

André Hirschowitz, Tom Hirschowitz, Ambroise Lafont, Marco Maggesi. Variable binding and substitution for (nameless) dummies. 25th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2022), Apr 2022, Munich, Germany. hal-03547002

HAL Id: hal-03547002

https://hal.science/hal-03547002v1

Submitted on 28 Jan 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Variable binding and substitution for (nameless) dummies

André Hirschowitz[1], Tom Hirschowitz[2]✉, Ambroise Lafont[3], and Marco Maggesi[4]

[1] Univ. Côte d'Azur, CNRS, LJAD, 06103, Nice, France
[2] Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, 73000, Chambéry, France
[3] University of New South Wales, Sydney, Australia
[4] Università degli Studi di Firenze, Italy

Abstract. 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.

Keywords: 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 $(n_1, \ldots, n_p)$ of natural numbers, whose meaning is that the considered operation has p arguments, with n_i variables bound in the ith argument, for all $i \in \{1, \ldots, p\}$.

Example 1.

- λ-abstraction has binding arity (1) (one argument, with one bound variable);
- application has binding arity $(0, 0)$ (two arguments, with no bound variable);
- 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).

There are several possible representations of the syntax specified by a binding signature, most of them benefiting from good semantical understanding. The traditional, nominal representation has been nicely framed within nominal sets [12]. The representation by De Bruijn levels, a.k.a. nested datatypes [5,1], is well-understood thanks to presheaf models [11], as is higher-order abstract syntax [19]. However, one of the oldest representations, using De Bruijn's idea of modelling variables with nameless dummies, does not benefit from any semantical framework. This may be related to the fact that it is often perceived

as low-level and error-prone [4]. Our goal in this paper is to equip De Bruijn's representation with a suitable semantical framework.

Let us start by stressing some of the features of this representation, for some fixed binding signature S.

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 $(n_1, \ldots, n_p)$, a map $o_{\mathrm{DB}_S} : \mathrm{DB}_S^p \to \mathrm{DB}_S$.

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 $(n_1, \ldots, n_p)$, $e_1, \ldots, e_p \in \mathrm{DB}_S$, and $f\colon \mathbb{N} \to \mathrm{DB}_S$,

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

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 (§2) 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[5] 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 (§3) 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.

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

We thus propose a theory for syntax with substitution, which is an alternative to the mainstream initial-algebra semantics of Fiore et al.'s [11]. 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 [11], 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 $1 \to \mathbf{Set}$ selecting $\mathbb{N}$, while Fiore et al.'s 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 3, §4) 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 §5, we show how De Bruijn S-algebras can be defined by resorting to (a slight generalisation [6] of) *pointed strong endofunctors*, in the spirit of [11].
- Their framework accomodates simple types and equations [7]; we also provide such extensions of our theory in §6 and §7.

Related work

Abstract frameworks for variable binding One of the mainstream such frameworks is [11]. This has been our main reference and in §5 we establish a strong link between this framework and our proposal. This link could probably be extended to variants such as [17,18,3].

In a more recent work, Allais et al. [1] introduce a universe of syntaxes, which essentially corresponds to a simply-typed version of binding signatures. Their framework is designed to facilitate the definition of 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 [13], 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 categories of well-behaved objects obtained in §4 are technically very close to nominal sets [12]: finite supports appear in the action-based presentation of nominal sets, while pullback preservation appears in their sheaf-based presentation. And indeed, any well-behaved presheaf yields a nominal set, and so does any well-behaved De Bruijn monad. However, these links are not entirely satisfactory, because they do not account for substitution. The reason is that the only categorical theory of substitution that we know of for nominal sets, by Power [24], is operadic rather than monadic, so we do not immediately see how to extend the correspondence.

Proof assistant libraries Allais et al. [1] mechanise their approach in Agda. In the same spirit, the presheaf-based approach was recently formalised [9].

De Bruijn representation benefits from well-developed proof assistant libraries, in particular Autosubst [26,27]. 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, for any endofunctor F, F-$\mathbf{alg}$ denotes 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 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 21. *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*

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

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

Definition 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. 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 3. 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 5, as an application of Theorem 2, 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 3. *Consider any set M, equipped with maps $s\colon M \times M^{\mathbb{N}} \to M$ and $v\colon \mathbb{N} \to M$. For any assignment $\sigma\colon \mathbb{N} \to M$, we define the assignment $\Uparrow \sigma\colon \mathbb{N} \to M$ by*

$$(\Uparrow \sigma)(0) = v(0)$$
$$(\Uparrow \sigma)(n+1) = \sigma(n)[\uparrow],$$

where $\uparrow\colon \mathbb{N} \to X$ maps any n to $v(n+1)$.

Remark 1. Both $\Uparrow$ and $\uparrow$ depend on M and (part of) (s, v). Here, and in other similar situations below, we abuse notation and omit such dependencies for readability.

Of course we may iterate lifting:

Definition 4. *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 5.

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

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

Definition 6. *Let $a = (n_1, \ldots, n_p)$ be any binding arity, M be any set, $s\colon M \times M^{\mathbb{N}} \to M$, and $v\colon \mathbb{N} \to M$ be any maps. 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):*

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

Remark 2. 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 7. *A* ***first-order signature*** *consists of a set O of* ***operations****, equipped with an* ***arity*** *map $ar\colon O \to \mathbb{N}$.*

Definition 8. *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 $o_X\colon X^{ar(o)} \to X$.*

Let us now generalise this to binding signatures.

Definition 9.

- *A* ***binding signature*** *[23] consists of a set O of* ***operations****, equipped with an* ***arity*** *map $ar\colon O \to \mathbb{N}^*$. 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 $|ar|\colon O \to \mathbb{N}$ maps any $o \in O$ to $|ar(o)|$.*

Example 4. 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 10. *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 11. *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 3. The induced endofunctor just depends on the underlying first-order signature.

Remark 4. 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 1. *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 map $s\colon \mathrm{DB}_S \times {\mathrm{DB}_S}^{\mathbb{N}} \to \mathrm{DB}_S$ 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 [22] and Coq [20].

Remark 5. 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.

Remark 6. We hope that our mechanisations [22,20] may be useful for future developments based on De Bruijn representation, to automatically generate the correct syntax and substitution from a suitable signature. This will have the advantage of reducing what needs to be read to make sure that the development actually does what is claimed. Normally, this part includes the whole definition of syntax and substitution, while our framework reduces it to only the binding signature. Our mechanisations may in fact be used for this purpose on existing developments, to certify the syntax and substitution, leaving only the binding signature for the reader to check.

Example 5. For the binding signature of λ-calculus (Example 4), the carrier of the initial model is $\mu Z.\mathbb{N} + Z + Z^2$, and substitution is defined inductively by:

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

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

Remark 7. More explicitly, the first axiom says: $f(s(x,\sigma)) = t(f(x), f \circ \sigma)$.

Notation 31. *De Bruijn monads and morphisms between them form a category, which we denote by* **DBMnd**.

Let us conclude this subsection by briefly mentioning a categorical point of view on the category of De Bruijn monads for the categorically-minded reader, in terms of **relative** monads [2].

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

Remark 8. Canonicity here means that the isomorphism lies over the canonical isomorphism $[1, \mathbf{Set}] \cong \mathbf{Set}$.

According to the theory of [2], this yields:

Corollary 1. *The tensor product $X \otimes Y := X \times Y^{\mathbb{N}}$ induces a skew monoidal [28] 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 $\mathrm{Lan}_{\mathbb{N}}(X)\colon \mathbf{Set} \to \mathbf{Set}$. By the standard formula for left Kan extensions [21], we have $\mathrm{Lan}_{\mathbb{N}}(X)(Y) \cong X \times Y^{\mathbb{N}} = X \otimes Y$. The result thus follows by [2, Theorems 4 and 5].

3.2 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 13. *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 14. *For any binding signature S, 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.*

Theorem 2. *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 1 on DB_S makes it initial in S-**DBAlg**.*

Proof. We have proved the result in both HOL Light [22] and Coq [20].

4 Relation to presheaf-based models

The classical initial-algebra semantics introduced in [11] 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.

4.1 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.

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. As is well-known, this category is equivalent to the category $[\mathbf{Set}, \mathbf{Set}]_f$ of finitary endofunctors on sets, and inherits from it a **substitution** monoidal structure. By construction, monoids for this monoidal structure are 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, satisfying a suitable compatibility condition.

The definition of Φ_S relies on an operation called derivation:

Definition 15 (Endofunctor associated to a binding signature).

- *Let the **derivative** X' of any functor $X : \mathbb{F} \to \mathbf{Set}$ 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 2. *Through the equivalence with finitary functors, derivation becomes $F'(A) = F(A+1)$, for any finitary $F : \mathbf{Set} \to \mathbf{Set}$ and $A \in \mathbf{Set}$.*

Example 6. For the binding signature S_λ of Example 4 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 [11,10] for details.

Definition 16. *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 I, satisfying two coherence conditions.*

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

Definition 17. *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 2 ([11,10]). *For all binding signatures S, Φ_S is pointed strong.*

At last, we arrive at the definition of models.

Definition 18. *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.*

$$\begin{tikzcd}
F(X) \otimes X \arrow[r, "st_{X,(X,v)}"] \arrow[d, "a \otimes X"'] & F(X \otimes X) \arrow[r, "F(s)"] & F(X) \arrow[d, "a"] \\
X \otimes X \arrow[rr, "s"'] & & X
\end{tikzcd}$$

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-$\mathbf{Mon}$ denote the category of F-monoids and morphisms between them.

Example 7. For the binding signature S_λ of Example 4, 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.

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 $T\colon \mathbb{F} \to \mathbf{Set}$ 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

$$\begin{tikzcd} \emptyset \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & n \arrow[d] \\ m \arrow[r] & m+n. \end{tikzcd}$$

Remark 9. The reader might wonder about other, i.e., non-empty pullbacks. But these are automatically preserved, by [29, Proposition 2.1].

Definition 19.

- *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 $\Phi_S\text{-}\mathbf{Mon}$ is **well-behaved** iff the underlying functor is. Let $\Phi_S\text{-}\mathbf{Mon}_{wb}$ denote the full subcategory spanned by well-behaved objects.*

Example 8. 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 3. *The subcategory $\Phi_S\text{-}\mathbf{Mon}_{wb}$ includes the initial object.*

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

Remark 10. In most natural situations, all models are in fact well-behaved [16, Proposition 5.17].

4.2 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 20. *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.*

Example 9. By Proposition 4 below, the initial S-algebra is finitary, for any binding signature S. For a counterexample, consider the greatest fixed point $\nu A.\mathbb{N}+\Sigma_S(A)$, for any S with at least one operation with more than one argument. E.g., if S has an operation of binding arity $(0,0)$, like application in λ-calculus, then the term $v(0)\ (v(1)\ (v(2)\ \ldots))$ does not have finite support.

Definition 21. *For any binding signature S, let $S\text{-}\mathbf{DBAlg}_{wb}$ denote the full subcategory spanning De Bruijn S-algebras whose underlying De Bruijn monad is finitary.*

Proposition 4. *The subcategory $S\text{-}\mathbf{DBAlg}_{wb}$ includes the initial object.*

4.3 Bridging the gap

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

Theorem 3. *Consider any binding signature S. The subcategories Φ_S - $\mathbf{Mon}_{wb}$ and S - $\mathbf{DBAlg}_{wb}$ are equivalent.*

Proof. See [16, Appendix A].

Remark 11. 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.

Restricting attention to well-behaved objects, we may thus benefit from the strengths of both approaches. Typically, in De Bruijn monads, free variables need to be computed explicitly, while presheaves come with intrinsic scoping, as terms are indexed by sets of potential free variables. Conversely, in some settings, observational equivalence may relate programs with different sets of free variables [25]. In such cases, it is useful to have all terms collected in one single set. This needs to be computed (and involves non-trivial quotienting) in presheaves, while it is direct in De Bruijn monads.

5 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 Φ_S-monoids [11]. In this section, we establish a different kind of link, by showing that, for any binding signature S, both categories S - **DBAlg** and Φ_S - **Mon** are instances of a common categorical construction. We have seen that the standard category Φ_S - **Mon** is constructed from the pointed strong endofunctor Φ_S, so we would like a similar construction of S - **DBAlg**. However, pointed strong endofunctors live on monoidal categories [11,10], while we have seen in Corollary §1 that $\mathbb{N}$ and the tensor product only equip **Set** with skew monoidal structure. In order to bridge this gap, we resort to a generalisation of pointed strengths to skew monoidal categories proposed by Borthelle et al. [6].

We give a condensed account: the interested reader is referred to [16, §6].

The starting point is that the endofunctor Σ_S associated to any given binding signature S may be equipped with a family of maps

$$\mathbf{dbs}_S \colon \Sigma_S(X) \otimes Y \to \Sigma_S(X \otimes Y).$$

However, in order for such a map to be well-defined, we need to assume that Y features variables and renaming, i.e., that it is a **pointed $\mathbb{N}$-module**, as we now introduce:

Definition 22.

- *An* $\mathbb{N}$***-module*** *is a set* X *equipped with an action* $X \times \mathbb{N}^{\mathbb{N}} = X \otimes \mathbb{N} \to X$*. of the monoid* $\mathbb{N}^{\mathbb{N}}$*.*
- *For such an action* $r : X \times \mathbb{N}^{\mathbb{N}} = X \otimes \mathbb{N} \to X$*, we generally denote* $r(x, f)$ *by* $x[f]_r$*, or merely* $x[f]$ *when clear from context.*
- *A morphism of* $\mathbb{N}$*-modules is a map between underlying sets, commuting with action in the obvious sense.*
- *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 10. Any De Bruijn monad (X, s, v) (in particular $\mathbb{N}$ itself) has a canonical structure of pointed $\mathbb{N}$-module given by v and $r(x, f) = x[v \circ f]$.

We may now define the map $\mathbf{dbs}_S$. Lifting of assignments (Definition 3) straightforwardly generalises to pointed $\mathbb{N}$-modules. Recalling the definition

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

where $ar(o) = (n_1^o, \ldots, n_{p_o}^o)$ for all $o \in O$, we thus simply have:

Definition 23. *For any binding signature* $S = (O, ar)$*, the* ***De Bruijn strength*** $\mathbf{dbs}_S$ *of the induced endofunctor* Σ_S *is defined by*

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

for all sets X *and pointed* $\mathbb{N}$*-modules* Y*, with again* $ar(o) = (n_1^o, \ldots, n_{p_o}^o)$*.*

The fact that any De Bruijn monad is in particular a pointed $\mathbb{N}$-module by Example 10 enables the definition of models in the strength-based approach:

Definition 24. *For any binding signature* S*, a* Σ_S***-monoid*** *is an object* X*, equipped with monoid and* Σ_S*-algebra structure, say* $s \colon X \otimes X \to X$*,* $v \colon \mathbb{N} \to X$*, and* $a \colon \Sigma_S(X) \to X$*, making the following pentagon commute.*

$$\begin{tikzcd} \Sigma_S(X) \otimes X \arrow[r, "\mathbf{dbs}_{S,X,X}"] \arrow[d, "a \otimes X"'] & \Sigma_S(X \otimes X) \arrow[r, "\Sigma_S(s)"] & \Sigma_S(X) \arrow[d, "a"] \\ X \otimes X \arrow[rr, "s"'] & & X \end{tikzcd} \tag{3}$$

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

Let Σ_S - $\mathbf{Mon}$ *denote the category of* Σ_S*-monoids and morphisms between them.*

Remark 12. In [16], this definition is framed in a more general context, notably emphasising the fact that $\mathbf{dbs}_S$ is in fact a **structural strength** on the endofunctor Σ_S.

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

Proposition 5. *For any binding signature $S = (O, ar)$ and De Bruijn monad (M, s, v) equipped with a map $o_M : M^p \to M$ for all $o \in O$ with $ar(o) = (n_1, \ldots, n_p)$, the following are equivalent:*

(i) *each map $o_M : M^p \to M$ satisfies the **a-binding condition** w.r.t. (s, v);*
(ii) *the corresponding map $\Sigma_S M \to M$ renders the pentagon (3) commutative.*

Corollary 3. *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 1 and 2.

Corollary 4. *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 substitution map $s : \mathrm{DB}_S \otimes \mathrm{DB}_S \to \mathrm{DB}_S$ 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 (3) (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 1. This correspondence translates one statement into the other.

Remark 13. This result hints at a potential push-button proof of Theorems 1 and 2 (and Corollary 4). Indeed, it is almost an instance of [6, 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.

6 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 sets with families.

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.

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

Remark 14. This provides a countable set of variables at each type, which may not quite be what the reader would have called "typed De Bruijn representation". An inconvenience of this representation is that an "erasure" map from typed to untyped terms appears to need to rely on a bijection $\mathbb{T} \times \mathbb{N} \cong \mathbb{N}$ for "renaming" variables. In particular, not all indices can be preserved by such a map.

Definition 26. *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.*

The analogue of the tensor product $X \otimes Y = X \times Y^{\mathbb{N}}$ will be played by $[\mathbf{N}, Y] \cdot X$, i.e., the iterated self-coproduct of X, with one copy per Y-assignment.

Notation 61. *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* $f : \mathbf{N} \to Y$*.*

Furthermore, Notation 21 straightforwardly adapts to the typed case.

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

Definition 27. *A* ***De Bruijn*** $\mathbb{T}$***-monad*** *is a* $\mathbb{T}$*-set* X*, equipped with*

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

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

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

Example 11. 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.

Morphisms generalise straightforwardly, and we get:

Proposition 6. *De Bruijn* $\mathbb{T}$*-monads and morphisms between them form a category* $\mathbf{DBMnd}(\mathbb{T})$*.*

6.2 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 28. *Let (X, s, v) denote any De Bruijn $\mathbb{T}$-monad. We first define a typed analogue $\uparrow^\tau$ of the $\uparrow$ of Definition 3, as below left, and then the* ***lifting*** *of any assignment $\sigma\colon \mathbf{N} \to X$ as below right.*

$$(\uparrow^\tau)_\tau(n) = v_\tau(n+1) \qquad (\uparrow^\tau)_{\tau'}(n) = v_{\tau'}(n) \quad (\text{if } \tau \neq \tau')$$

$$(\Uparrow^\tau \sigma)_\tau(0) = v_\tau(0) \qquad (\Uparrow^\tau \sigma)_\tau(n+1) = \sigma_\tau(n)[\uparrow^\tau] \qquad (\Uparrow^\tau \sigma)_{\tau'}(n) = \sigma_{\tau'}(n)[\uparrow^\tau] \ (\text{if } \tau \neq \tau').$$

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.

We then generalise first-order and binding arities. The main point is:

Definition 29. *A* ***binding arity*** *is an element of $(\mathbb{T}^* \times \mathbb{T})^* \times \mathbb{T}$, i.e., a tuple $(((\gamma_1, \tau_1), \ldots, (\gamma_p, \tau_p)), \tau)$, where each $\gamma_i \in \mathbb{T}^*$ is a list of types, and each τ_i, as well as τ, are types, thought of as an inference rule* $\dfrac{\gamma_1 \vdash \tau_1 \quad \ldots \quad \gamma_p \vdash \tau_p}{\vdash \tau}$.

Example 12. The binding signature for simply-typed λ-calculus has two operations $\mathrm{lam}_{\tau,\tau'}$ and $\mathrm{app}_{\tau,\tau'}$ for all types τ and τ', of respective arities

$$\frac{\tau \vdash \tau'}{\vdash \tau \to \tau'} \qquad \text{and} \qquad \frac{\vdash \tau \to \tau' \quad \vdash \tau}{\vdash \tau'} .$$

This allows us to generalise binding conditions, as follows.

Definition 30. *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):*

$$\begin{aligned} &\forall \sigma\colon \mathbf{N} \to M, x_1, \ldots, x_p \in M(\tau_1) \times \ldots \times M(\tau_p), \\ &o(x_1, \ldots, x_p)[\sigma] = o(x_1[\Uparrow^{\gamma_1} \sigma], \ldots, x_p[\Uparrow^{\gamma_p} \sigma]). \end{aligned} \tag{4}$$

We may now generalise signatures and their models.

Definition 31. *A* ***$\mathbb{T}$-binding signature*** *consists of a set O of* ***operations****, equipped with an arity map $O \to (\mathbb{T}^* \times \mathbb{T})^* \times \mathbb{T}$.*

Definition 32. *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 algebra structure on X for the underlying first-order signature $|S|$, in the obvious sense, such that for all $o \in O$ with arity $ar(o) = (((\gamma_1, \tau_1), \ldots, (\gamma_p, \tau_p)), \tau)$, the structural map $o_X\colon X(\tau_1) \times \ldots \times X(\tau_p) \to X(\tau)$ satisfies the $ar(o)$-binding condition w.r.t. (s, v).*

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

Finally, following the untyped case, we may associate to each signature an endofunctor Σ_S, and we have the following typed extension of the initiality theorem.

Theorem 4. *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 $o_{\mathrm{DB}_S}\colon \mathrm{DB}_S(\tau_1) \times \ldots \times \mathrm{DB}_S(\tau_p) \to \mathrm{DB}_S(\tau)$ for all $o \in O$ with $ar(o) = (((\gamma_1,\tau_1),\ldots,(\gamma_p,\tau_p)),\tau)$. 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 $f : \mathbf{N} \to \mathrm{DB}_S$, $s_\tau(v_\tau(n), f) = f_\tau(n)$, 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**.*

Example 13. While we saw in Example 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.) It is relatively straightforward to design a binding signature for this De Bruijn monad, following [15].

7 Equations

In this section, we introduce a notion of equational theory for specifying (typed) De Bruijn monads, following ideas from [8].

Definition 33. *A **De Bruijn equational theory** consists of*

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

$$\begin{tikzcd} S\text{-}\mathbf{DBAlg} \arrow[rr, "L,R"] \arrow[dr, "U^S"'] & & T\text{-}\mathbf{DBAlg} \arrow[dl, "U^T"] \\ & \mathbf{DBMnd}(\mathbb{T}) & \end{tikzcd}$$

Example 14. Recalling the binding signature S_Λ for λ-calculus from Example 4, 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 $(e_1,e_2) \mapsto \mathrm{app}(\mathrm{lam}(e_1),e_2)$ while
- $R(X)$ has as structure map $(e_1,e_2) \mapsto e_1[e_2 \cdot \mathrm{id}]$.
  (Here $e_2 \cdot \mathrm{id}$ denotes the assignment $0 \mapsto e_2$, $n+1 \mapsto v(n)$.)

Definition 34. *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 15. 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 35. *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,*

$$\frac{}{o'_{L(\mathrm{DB}_S)}(e_1, \ldots, e_p) \sim_E o'_{R(\mathrm{DB}_S)}(e_1, \ldots, e_p)}$$

$$\frac{e_1 \sim_E e'_1 \qquad \ldots \qquad e_q \sim_E e'_q}{o_{\mathrm{DB}_S}(e_1, \ldots, e_q) \sim_E o_{\mathrm{DB}_S}(e'_1, \ldots, e'_q)}$$

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

Example 15. For the equational theory of Example 14, the first rule instantiates precisely to the β-rule, while the second enforces congruence.

Theorem 5. *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 [20] and HOL [22].

Example 16. The initial model for the equational theory of Example 14 is the quotient of λ-terms in De Bruijn representation by β-equivalence.

Remark 16. In [16, §9], we mention an equivalent way of defining De Bruijn equational theories in terms of modules.

8 Conclusion

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 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 [11], 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 the mainstream, abstract framework of [11,10]. Finally, we have shown that our theory extends easily to a simply-typed setting, and smoothly incorporates equations.

References

1. Allais, G., Atkey, R., Chapman, J., McBride, C., McKinna, J.: A type and scope safe universe of syntaxes with binding: their semantics and proofs. Proceedings of the ACM on Programming Languages **2**(ICFP), 90:1–90:30 (2018). https://doi.org/10.1145/3236785
2. Altenkirch, T., Chapman, J., Uustalu, T.: Monads need not be endofunctors. Logical Methods in Computer Science **11**(1) (2015). https://doi.org/10.2168/LMCS-11(1:3)2015
3. Arkor, N., McDermott, D.: Abstract clones for abstract syntax. In: Kobayashi, N. (ed.) Proc. 6th International Conference on Formal Structures for Computation and Deduction. Leibniz International Proceedings in Informatics (LIPIcs), vol. 195, pp. 30:1–30:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). https://doi.org/10.4230/LIPIcs.FSCD.2021.30
4. Berghofer, S., Urban, C.: A head-to-head comparison of de bruijn indices and names. Electronic Notes in Theoretical Computer Science **174**(5), 53–67 (2007). https://doi.org/10.1016/j.entcs.2007.01.018
5. Bird, R.S., Paterson, R.: De bruijn notation as a nested datatype. Journal of Functional Programming **9**(1), 77–91 (1999). https://doi.org/10.1017/S0956796899003366
6. Borthelle, P., Hirschowitz, T., Lafont, A.: A cellular Howe theorem. In: Hermanns, H., Zhang, L., Kobayashi, N., Miller, D. (eds.) Proc. 35th ACM/IEEE Symposium on Logic in Computer Science ACM (2020). https://doi.org/10.1145/3373718.3394738
7. Fiore, M.P., Hur, C.K.: Second-order equational logic. In: Proceedings of the 19th EACSL Annual Conference on Computer Science Logic (CSL 2010) (2010)
8. Fiore, M., Hur, C.K.: On the construction of free algebras for equational systems. Theoretical Computer Science **410**, 1704–1729 (2009)
9. Fiore, M., Szamozvancev, D.: Formal metatheory of second-order abstract syntax. Proceedings of the ACM on Programming Languages **6**(POPL) (2022). https://doi.org/10.1145/3498715
10. Fiore, M.P.: Second-order and dependently-sorted abstract syntax. In: LICS. pp. 57–68. IEEE (2008). https://doi.org/10.1109/LICS.2008.38
11. Fiore, M.P., Plotkin, G.D., Turi, D.: Abstract syntax and variable binding. In: Proc. 14th Symposium on Logic in Computer Science IEEE (1999)
12. Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax involving binders. In: Proc. 14th Symposium on Logic in Computer Science IEEE (1999)
13. Gheri, L., Popescu, A.: A formalized general theory of syntax with bindings: Extended version. Journal of Automated Reasoning **64**(4), 641–675 (2020). https://doi.org/10.1007/s10817-019-09522-2
14. Hirschowitz, A., Hirschowitz, T., Lafont, A.: Modules over monads and operational semantics. In: Ariola, Z.M. (ed.) Proc. 5th International Conference on Formal Structures for Computation and Deduction. Leibniz International Proceedings in Informatics (LIPIcs), vol. 167, pp. 12:1–12:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020). https://doi.org/10.4230/LIPIcs.FSCD.2020.12
15. Hirschowitz, A., Hirschowitz, T., Lafont, A.: Modules over monads and operational semantics (2021), submitted expanded version of [14]
16. Hirschowitz, A., Hirschowitz, T., Lafont, A., Maggesi, M.: Variable binding and substitution for (nameless) dummies (2021), https://amblafont.github.io/articles/debruijn-extended.pdf, preprint

17. Hirschowitz, A., Maggesi, M.: Modules over monads and linearity. In: Proc. 14th International Workshop on Logic, Language, Information and Computation LNCS, vol. 4576, pp. 218–237. Springer (2007). https://doi.org/10.1007/3-540-44802-0_3
18. Hirschowitz, A., Maggesi, M.: Modules over monads and initial semantics. Information and Computation **208**(5), 545–564 (2010). https://doi.org/10.1016/j.ic.2009.07.003
19. Hofmann, M.: Semantical analysis of higher-order abstract syntax. In: Proc. 14th Symposium on Logic in Computer Science IEEE (1999)
20. Lafont, A.: Initial algebra semantics for de Bruijn monads in Coq. https://github.com/amblafont/binding-debruijn (2021)
21. Mac Lane, S.: Categories for the Working Mathematician. No. 5 in Graduate Texts in Mathematics, Springer, 2nd edn. (1998)
22. Maggesi, M.: Initial algebra semantics for de Bruijn monads in HOL Light. https://github.com/maggesi/dbmonad/tree/master/De_Bruijn (2021)
23. Plotkin, G.: An illative theory of relations. In: Cooper, R., et al. (eds.) Situation Theory and its Applications. p. 133–146. No. 22 in CSLI Lecture Notes, Stanford University (1990)
24. Power, J.: Abstract syntax: Substitution and binders: Invited address. Electronic Notes in Theoretical Computer Science **173**, 3–16 (04 2007). https://doi.org/10.1016/j.entcs.2007.02.024
25. Sangiorgi, D., Walker, D.: The π-calculus – A Theory of Mobile Processes. Cambridge University Press (2001)
26. Schäfer, S., Tebbi, T., Smolka, G.: Autosubst: Reasoning with de Bruijn terms and parallel substitutions. In: Urban, C., Zhang, X. (eds.) Proc. 6th International Conference on Interactive Theorem Proving. LNCS, vol. 9236, pp. 359–374. Springer (2015). https://doi.org/10.1007/978-3-319-22102-1_24
27. Stark, K., Schäfer, S., Kaiser, J.: Autosubst 2: reasoning with multi-sorted de bruijn terms and vector substitutions. In: Mahboubi, A., Myreen, M.O. (eds.) Proc. 8th International Conference on Certified Programs and Proofs. pp. 166–180. ACM (2019). https://doi.org/10.1145/3293880.3294101
28. Szlachányi, K.: Skew-monoidal categories and bialgebroids. Advances in Mathematics **231**, 1694–1730 (2012). https://doi.org/10.1016/j.aim.2012.06.027
29. Trnková, V.: Some properties of set functors. Commentationes Mathematicæ Universitatis Carolinæ **10**(2), 323–352 (1969)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.