List Objects with Algebraic Structure

Marcelo Fiore[1] **and Philip Saville**[2]

1 **Computer Laboratory, University of Cambridge, 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK**
Marcelo.Fiore@cl.cam.ac.uk

2 **Computer Laboratory, University of Cambridge, 15 JJ Thomson Avenue, Cambridge CB3 0FD, UK**
Philip.Saville@cl.cam.ac.uk

Abstract

We introduce and study the notion of list object with algebraic structure. The first key aspect of our development is that the notion of list object is considered in the context of monoidal structure; the second key aspect is that we further equip list objects with algebraic structure in this setting. Within our framework, we observe that list objects give rise to free monoids and moreover show that this remains so in the presence of algebraic structure. In addition, we provide a basic theory explicitly describing as an inductively defined object such free monoids with suitably compatible algebraic structure in common practical situations. This theory is accompanied by the study of two technical themes that, besides being of interest in their own right, are important for establishing applications. These themes are: parametrised initiality, central to the universal property defining list objects; and approaches to algebraic structure, in particular in the context of monoidal theories. The latter leads naturally to a notion of nsr (or near semiring) category of independent interest. With the theoretical development in place, we touch upon a variety of applications, considering Natural Numbers Objects in domain theory, giving a universal property for the monadic list transformer, providing free instances of algebraic extensions of the Haskell Monad type class, elucidating the algebraic character of the construction of opetopes in higher-dimensional algebra, and considering free models of second-order algebraic theories.

1998 ACM Subject Classification D.1.1 Applicative (Functional) Programming; D.3.1 Formal Definitions and Theory; D.3.3 Language Constructs and Features; F.3.2 Semantics of Programming Languages F.3.3 Studies of Program Constructs.

Keywords and phrases list object; free monoid; strong monad; (cartesian, linear, and second-order) algebraic theory; near semiring; Haskell Monad type class; opetope.

Digital Object Identifier 10.4230/LIPIcs.FSCD.2017.16

1 Introduction

Lists are a basic and fundamental construction in computer science and mathematics.

In type theory [35] and programming theory [37], the polymorphic type of lists $\mathrm{L}(X)$ is an inductive type with constructors

$$nil : \mathrm{L}(X) \quad , \quad cons : X \times \mathrm{L}(X) \to \mathrm{L}(X)$$

and an eliminator supporting definitions by primitive recursion. Amongst the variety of possible primitive-recursion schemes, here we will be concerned with *pure iteration* [43]; by which from $n : P \to L$ and $c : X \times L \to L$ one may form the parametrised iterator

$\mathrm{it}(n,c) : \mathrm{L}(X) \times P \to L$ defined by

$$\mathrm{it}(n,c)(\mathit{nil},p) = n(p) \quad , \quad \mathrm{it}(n,c)(\mathit{cons}(x,l),p) = c\big(x, \mathrm{it}(n,c)(l,p)\big) \quad .$$

The type of Natural Numbers is then that of lists on the unit type.

In category theory, the list type is universally modelled by the notion of (parametrised) list object, directly generalising Lawvere's notion of NNO (Natural Numbers Object) [29]. A *list object* $\mathrm{L}(X)$ on an object X in a category with finite products $(1,\times)$, is an initial algebraic structure of the form

$$1 \xrightarrow{\mathit{nil}} \mathrm{L}(X) \xleftarrow{\mathit{cons}} X \times \mathrm{L}(X) \quad .$$

Such a structure is a *parametrised list object* whenever it is parametrised initial, in the sense that for every structure $(P \xrightarrow{n} L \xleftarrow{c} X \times L)$ there exists a unique mediating map $\mathrm{it}(n,c) : \mathrm{L}(X) \times P \to L$ such that

$$\begin{array}{ccccc} 1 \times P & \xrightarrow{\mathit{nil}\times P} & \mathrm{L}(X) \times P & \xleftarrow{\mathit{cons}\times P} & X \times \mathrm{L}(X) \times P \\ \cong \downarrow & & \downarrow{\scriptstyle \mathrm{it}(n,c)} & & \downarrow{\scriptstyle X\times\mathrm{it}(n,c)} \\ P & \xrightarrow[n]{} & L & \xleftarrow[c]{} & X \times L \end{array}$$

It is well-known that when the binary product $(\times)$ is closed these notions of list object coincide, and that in the further presence of binary coproducts $(+)$ one may explicitly describe list objects by means of initial algebras:

$$\mathrm{L}(X) = \mu A.\, 1 + X \times A$$

where we use the common notation $\mu A.\, F(A)$ for the initial F-algebra. However, there are important scenarios (*e.g.* pretoposes [21, 33]) where the product structure is not closed. For them the more general parametrised concept is the appropriate one, and we henceforth adopt it as primitive.

In this paper, we investigate two orthogonal generalisations of list objects. One generalisation is concerned with weakening the cartesian structure of list objects to a monoidal one (Definition 3.1). Whilst NNOs have been studied in cartesian [29] and monoidal [6, 38] settings, list objects have been mainly considered in the cartesian one. The other generalisation is novel in that we equip list objects with further algebraic structure with respect to which the iterator is an algebra homomorphism (Definition 5.1).

Section 2 provides the necessary categorical background for setting up the above definitions. List objects in monoidal categories are introduced in Section 3, where we observe that they give rise to free monoids (Lemma 3.4). A main example is the monad freely generated by an endofunctor (Example 3.5). The parametrisation aspect inherent to list objects is considered and studied in broader generality in Section 4. In particular, we present a theory that accounts for the parametrised initiality of list objects as iteratively-constructed initial algebras

$$\mathrm{L}(X) = \mu A.\, I + X \otimes A$$

in contexts where the monoidal structure $(I,\otimes)$ may not be left-closed (Corollary 4.9). This general theory applies in the further context of list objects with algebraic structure (specifically Lemma 4.8) and plays an essential role in the application to higher-dimensional algebra that we present in Section 7.2.2.

The central notion of list object with T-algebraic structure, as prescribed by a strong monad T, is introduced in Section 5 and referred to as T-list object (Definition 5.1). This is accompanied by a related notion of monoid with compatible T-algebraic structure, referred to as T-monoid (Definition 5.2). These two notions are related as follows: firstly, we observe

that T-list objects $\mathrm{M}(X)$ yield free T-monoids (Lemma 5.6); secondly, we give an explicit description of T-list objects as parametrised initial algebras (Lemma 5.7):

$$\mathrm{M}_T(X) = \mu A.\, T(I + X \otimes A)$$

that is available in common practical situations (Corollaries 5.10 and 5.11). Overall, thus, this yields a universal inductive description of free T-monoids by means of parametrised initial-algebras that is of wide applicability (Theorem 5.8). This result, which we had obtained independently, also features in the recent work of Piróg [39], who establishes it by different methods.

Throughout the above development, algebraic structure is considered abstractly as encapsulated by the notion of strong monad. Section 6 complements this from the perspective of algebraic theories, whilst Section 7 discusses applications to programming semantics and algebraic theories presented in the context of related work. The study of linear algebraic theories (Section 6.2) led us to the notion of nsr (or near semiring) category (Definition 6.2), which, besides being of independent interest, plays a role in the applications to functional programming and higher-dimensional algebra respectively presented in Sections 7.2.1 and 7.2.2. Finally, Section 8 concludes with a brief discussion on the significance of our results, and indications of ongoing work.

2 Algebraic structure in monoidal categories

We begin by outlining the basic categorical notions of algebraic and monoidal structure which are relevant to our development.

Strong monoidal structures. Throughout we work with *monoidal categories* $(\mathcal{C}, I, \otimes)$. To simplify notation, we replace the structural isomorphisms by an unlabelled $\cong$ in diagrams.

We will be considering *strong multiary endofunctors* (F, st) on monoidal categories $(\mathcal{C}, I, \otimes)$ consisting of a functor $F : \mathcal{C}^n \to \mathcal{C}$ $(n \in \mathbb{N})$ together with a *strength* st, which is given by a natural transformation $st_{(X_1,\ldots,X_n),Y} : F(X_1, \ldots, X_n) \otimes Y \to F(X_1 \otimes Y, \ldots X_n \otimes Y)$ subject to coherence conditions. Maps of strong endofunctors $(F, st) \to (F', st')$ are *strong natural transformations* given by natural transformations $F \to F'$ satisfying coherence conditions. (See *e.g.* [25] for details, or [10, Section I.1.2] for related, more general, notions.)

A *strong monad* $T = (T, st, \mu, \eta)$ is a strong functor (T, st) equipped with strong natural transformations $\eta : (\mathrm{Id}, \mathrm{id}) \to (T, st)$ and $\mu : (TT, T(st) \circ st) \to (T, st)$ subject to the monad laws. (See *e.g.* [32, Section VI.1].)

We will be specifically concerned with two kinds of algebraic structure on objects in a monoidal category: monoids and monad algebras. We recall both notions.

Monoids. A *monoid* (M, e, m) in a monoidal category $(\mathcal{C}, I, \otimes)$ consists of an object $M \in \mathcal{C}$ together with a *unit* map $e : I \to M$ and a *multiplication* map $m : M \otimes M \to M$, subject to identity and associativity laws. *Homomorphisms* of monoids are morphisms of the underlying objects that preserve units and commute with the multiplications. We write $Mon(\mathcal{C})$ for the corresponding category. (See *e.g.* [32, Section VII.3] for details.)

Monad algebras. An *algebra* (X, x) for a (strong) endofunctor (F, st) consists of an object X together with a structure map $x : FX \to X$. Algebra *homomorphisms* $(X, x) \to (Y, y)$ are maps $h : X \to Y$ such that $h \circ x = y \circ Fh$. We write $F\text{-}alg$ for the corresponding category.

An *algebra* for a (strong) monad (T, st, η, μ) is a T-algebra with structure map satisfying

identity and associativity laws with respect to η and μ. We write $T\text{-}Alg$ for the full subcategory of $T\text{-}alg$ so determined. (See *e.g.* [32, Section VI.2].)

Crucially, we will be interested in *left homomorphisms* [24] between algebras. These are given by maps $h : X \otimes P \to Y$ such that

$$\begin{array}{ccccc} F(X) \otimes P & \xrightarrow{st_{X,P}} & F(X \otimes P) & \xrightarrow{Fh} & FY \\ {\scriptstyle x\otimes P}\downarrow & & & & \downarrow{\scriptstyle y} \\ X \otimes P & & \xrightarrow[h]{} & & Y \end{array} \tag{1}$$

3 List objects

List objects have been mainly studied in the cartesian setting, see *e.g.* [21, 7, 33]. In the linear setting, the special case of Lawvere's NNO [29], which amounts to the list object on a unit object, has been considered in monoidal categories [38] (see also [6]) and in a syntactic calculus [2]. As for general list objects, as far as we know, they have only appeared indirectly under the guise of algebraically-free monoids (see Remark 3.2 below). The notion of list object is as expected and we make it explicit below.

▶ Remark. What we define under the name of *list object* is frequently referred to, in the cartesian setting, as a *parametrised list object*. We have adopted the shorter terminology for brevity and because we feel that it is the appropriate concept for the monoidal setting — especially in the extension to include algebraic structure put forward in Section 5.

▶ **Definition 3.1.** Let X be an object of a monoidal category $(I, \otimes)$.

1. A *list algebra* on X is an object L together with a pair of maps $\big(I \to L \leftarrow X \otimes L\big)$.
2. A *list object* on X is a list algebra $\big(I \xrightarrow{nil} \mathrm{L}(X) \xleftarrow{cons} X \otimes \mathrm{L}(X)\big)$ that is parametrised initial, in the sense that for every parametrised list algebra $(P \xrightarrow{n} L \xleftarrow{c} X \otimes L)$ there exists a unique mediating map $\mathrm{it}(n, c) : \mathrm{L}(X) \otimes P \to L$ such that

$$\begin{array}{ccccc} I \otimes P & \xrightarrow{nil\otimes P} & \mathrm{L}(X) \otimes P & \xleftarrow{cons\otimes P} & X \otimes \mathrm{L}(X) \otimes P \\ {\scriptstyle\cong}\downarrow & & {\scriptstyle \mathrm{it}(n,c)}\downarrow & & \downarrow{\scriptstyle X\otimes \mathrm{it}(n,c)} \\ P & \xrightarrow[n]{} & L & \xleftarrow[c]{} & X \otimes L \end{array}$$

Thus, list objects are universally characterised by their constructors and their support of definitions by pure iteration [43].

▶ Remark 3.2. In a monoidal category $(I, \otimes)$ with binary coproducts $(+)$ that are preserved by the endofunctor $(-)\otimes P$ for every object P, the notion of list object on an object X is equivalent to Kelly's notion of the algebraically-free monoid on the free pointed-object $(I \to I + X)$ on X, see [23, §23.1].

▶ **Example 3.3.** Our notion of list object on the unit object coincides with the notion of *LNNO (Left Natural Numbers Object)* of Paré and Román [38].

We summarise the algebraic structure of and provided by list objects.

▶ **Lemma 3.4.** *1. Every list object on X with carrier $\mathrm{L}(X)$ provides a monoid structure with carrier $\mathrm{L}(X)$ that is the free monoid on X.*

2. If the ambient category $\mathcal{C}$ has list objects, then the forgetful functor $Mon(\mathcal{C}) \to \mathcal{C}$ is monadic. This induces a list (or free monoid) monad L.

3. When the monoidal structure is cartesian the monad L is strong.

▶ **Example 3.5.** The above explains the well-known construction of the free monad on an endofunctor by means of free algebras. Indeed, for an endofunctor F, a choice of free F-algebra $\big(X \xrightarrow{\eta_X} T_F X \xleftarrow{\varphi_X} F(T_F X)\big)$ on each object X determines a list object $(\mathrm{Id} \xrightarrow{\eta} T_F \xleftarrow{\varphi} F \circ T_F)$ on F in the category of endofunctors with the composition monoidal structure, inducing the free monad T_F on F. The case $F = \mathrm{Id}$ is precisely Burroni's notion of *NNO functor* that underlies his notion of Peano-Lawvere category [6] (consult also [38]).

▶ Remark. In general, the notions of list object and of free monoid do not agree. The former may not be available, as in the category of sets with the coproduct monoidal structure.

4 Parametrised initial algebras

As natural as the definition of list object is, it is also important to analyse its universal property through the theory of parametrised initial algebras. This section presents the relevant aspects of this theory.

▶ **Definition 4.1.** An *F-algebra* for a functor $F : \mathcal{D} \times \mathcal{C} \to \mathcal{C}$ is defined as a pair $\big((D, C), \varphi\big)$ with $\varphi : F(D, C) \to C$ in $\mathcal{C}$. F-algebras form a category *F-alg* in which a *homomorphism* $(D, C) \to (D', C')$ is a pair $(g : D \to D', f : C \to C')$ such that $f \circ \varphi = \varphi' \circ F(g, f)$.

▶ **Definition 4.2.** For a functor $F : \mathcal{D} \times \mathcal{C} \to \mathcal{C}$, the *initial-algebra functor* $\mu F : \mathcal{D} \to \mathcal{C}$ is defined, whenever possible, by choosing an initial $F(X, -)$-algebra $\big(\mu A.\, F(X, A), \varphi_X\big)$ and setting $\mu F(X) := \mu A.\, F(X, A)$. On morphisms, the action of μF on $f : X \to Y$ is uniquely determined by the fact that $\big(f, \mu F(f)\big)$ is an F-algebra homomorphism from $(\mu F(X), \varphi_X)$ to $(\mu F(Y), \varphi_Y)$.

▶ **Example 4.3.** Consider the functor $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C} : (X, A) \mapsto I + X \otimes A$ in a monoidal category $(\mathcal{C}, I, \otimes)$ with binary coproducts $(+)$. In the presence of list objects $\mathrm{L}(X)$ on every $X \in \mathcal{C}$, the endofunctor μF is available and canonically isomorphic to the list-object endofunctor L.

We recall the basic iterative construction for building initial-algebra functors.

▶ **Lemma 4.4** ([1, 30])**.** *For every functor $F : \mathcal{D} \times \mathcal{C} \to \mathcal{C}$ where $\mathcal{C}$ has an initial object (0) and ω-colimits and $F_D := F(D, -)$ is ω-cocontinuous for every $D \in \mathcal{D}$, we have a functor $\mu F : \mathcal{D} \to \mathcal{C}$ defined by setting $\mu F(D)$ to be a chosen colimit of the ω-chain $\langle {F_D}^n(0 \to F_D 0)\rangle_{n \in \mathbb{N}}$. Moreover, if F is ω-cocontinuous then so is μF.*

We turn our attention to parametrised initiality. The general theory requires the framework of monoidal actions, see [10, Section I.1]. We do not dwell on this here; rather, we restrict attention to the special case relevant to the present development.

▶ **Definition 4.5.** For a monoidal category $(\mathcal{C}, I, \otimes)$, a strong functor $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, and an object $Z \in \mathcal{C}$, an initial $F(Z, -)$-algebra $(\mu A.\, F(Z, A), \varphi_Z)$ is said to be *parametrised initial* whenever for every object $P \in \mathcal{C}$ and algebra structure $\gamma : F(Z \otimes P, C) \to C$ on an object $C \in \mathcal{C}$ there is a unique map $u : \big(\mu A.\, F(Z, A)\big) \otimes P \to C$ making the following diagram commute

$$\begin{array}{ccccc} F\big(Z, \mu A.\, F(Z, A)\big) \otimes P & \xrightarrow{st} & F\big(Z \otimes P, \big(\mu A.\, F(Z, A)\big) \otimes P\big) & \xrightarrow{F(Z \otimes P, u)} & F(Z \otimes P, C) \\ \downarrow{\scriptstyle \varphi_Z \otimes P} & & & & \downarrow{\scriptstyle \gamma} \\ \big(\mu A.\, F(Z, A)\big) \otimes P & & \xrightarrow[u]{\quad\quad\quad\quad} & & C \end{array}$$

▶ Remark. As is well-known, for a left-closed monoidal category (*i.e.*, one in which $(-)\otimes P$ has a right adjoint for all objects P), the notions of initiality and of parametrised initiality coincide.

▶ **Example 4.6.** Consider a monoidal category $(\mathcal{C}, I, \otimes)$ with binary coproducts $(+)$ that are preserved by every endofunctor $(-)\otimes P$ for $P \in \mathcal{C}$. For every object $X \in \mathcal{C}$, the functor $F_X : \mathcal{C}\times\mathcal{C} \to \mathcal{C} : (Z, A) \mapsto Z + X \otimes A$ is canonically strong and the notion of parametrised initial $F(I,-)$-algebra coincides with that of list object on X.

We now provide a conceptual framework for parametrised iteratively-constructed initial algebras. Our main technical tool is the following *lax-uniformity* property of initial algebra functors. (For uniformity in a related 2-dimensional setting see [9, Section 7.3.2].)

▶ **Theorem 4.7.** *Consider a diagram of categories, functors, and a natural transformation as on the left below*

$$\begin{array}{ccc} \mathcal{D}\times\mathcal{C} & \xrightarrow{F} & \mathcal{C} \\ {\scriptstyle K\times J}\downarrow & \overset{h}{\Leftarrow} & \downarrow{\scriptstyle J} \\ \mathcal{B}\times\mathcal{A} & \xrightarrow[G]{} & \mathcal{A} \end{array} \qquad\qquad \begin{array}{ccc} \mathcal{D} & \xrightarrow{\mu F} & \mathcal{C} \\ {\scriptstyle K}\downarrow & \overset{\mu h}{\Leftarrow} & \downarrow{\scriptstyle J} \\ \mathcal{B} & \xrightarrow[\mu G]{} & \mathcal{A} \end{array} \tag{2}$$

Assume that the categories $\mathcal{C}, \mathcal{A}$ both have an initial object and ω-colimits, and that the functors $F(D,-), G(B,-), J$ are ω-cocontinuous for all $D \in \mathcal{D}$ and $B \in \mathcal{B}$. If J also preserves initial objects, then for all algebras $\alpha : G(KD, A) \to A$ there exists a unique map $\alpha^\star : J(\mu F(D)) \to A$ such that

$$\begin{array}{ccccc} JF(D,\mu F(D)) & \xrightarrow{h_{D,\mu F(D)}} & G\big(KD, J(\mu F(D))\big) & \xrightarrow{G(KD,\alpha^\star)} & G(KD,A) \\ {\scriptstyle J\cong}\downarrow & & & & \downarrow{\scriptstyle \alpha} \\ J(\mu F(D)) & & \xrightarrow[\alpha^\star]{} & & A \end{array}$$

and we have the situation as on the right in (2) above with

$$\mu h_D := \big(G(KD, \mu G(KD)) \xrightarrow{\cong} \mu G(KD)\big)^\star \ .$$

The lemma below follows by applying the theorem above to the diagram

$$\begin{array}{ccc} \mathcal{C}\times\mathcal{C} & \xrightarrow{F} & \mathcal{C} \\ {\scriptstyle \mathrm{Id}\times(-\otimes P)}\downarrow & \overset{st}{\Longleftarrow} & \downarrow{\scriptstyle -\otimes P} \\ \mathcal{C}\times\mathcal{C} & \xrightarrow[F(-\otimes P,=)]{} & \mathcal{C} \end{array}$$

for each $P \in \mathcal{C}$.

▶ **Lemma 4.8.** *Let $(\mathcal{C}, I, \otimes)$ be a monoidal category with an initial object and ω-colimits, both preserved by the endofunctor $(-)\otimes P$ for each $P \in \mathcal{C}$. For an ω-cocontinuous strong endofunctor $(F, st) : \mathcal{C}\times\mathcal{C} \to \mathcal{C}$, every initial $F(Z,-)$-algebra for $Z \in \mathcal{C}$ is parametrised initial.*

From this lemma and Example 4.6, we attain conditions for (not necessarily left-closed) monoidal categories under which list objects arise as iteratively-constructed initial algebras.

▶ **Corollary 4.9.** *Let $(\mathcal{C}, I, \otimes)$ be a monoidal category with finite coproducts $(0, +)$ and ω-colimits, both preserved by the endofunctor $(-)\otimes P$ for all $P \in \mathcal{C}$. For $X \in \mathcal{C}$, if the endofunctor $X \otimes (-)$ is ω-cocontinuous, then the initial $(I + X \otimes -)$-algebra gives a list-object structure on X.*

5 List objects and monoids with algebraic structure

We have observed that list objects provide free monoids (Lemma 3.4). In this section, we generalise this fact to encompass notions of list object and monoid that are equipped with algebraic structure.

List objects with algebraic structure. We equip list objects with a monadic algebra structure. Crucially, we use this algebraic structure to refine their defining universal property. This is the key concept of the paper.

▶ **Definition 5.1.** Let (T, st) be a strong monad on a monoidal category $(I, \otimes)$.

1. A *T-list algebra* on an object X consists of an object A together with a list algebra $\big(I \to A \leftarrow X \otimes A\big)$ and a T-algebra structure $TA \to A$.
2. A *T-list object* on an object X is a parametrised initial T-list algebra. Explicitly, it is a T-list algebra as on the left below

$$\begin{array}{ccccc} & & T(\mathrm{M}(X)) & & \\ & & \downarrow{\scriptstyle \tau} & & \\ I & \xrightarrow[nil]{} & \mathrm{M}(X) & \xleftarrow[cons]{} & X \otimes \mathrm{M}(X) \end{array} \qquad \begin{array}{ccccc} & & TA & & \\ & & \downarrow{\scriptstyle \alpha} & & \\ P & \xrightarrow[n]{} & A & \xleftarrow[c]{} & X \otimes A \end{array}$$

such that for every parametrised T-list algebra as on the right above there exists a unique mediating map $\mathrm{it}(n, c, \alpha) : \mathrm{M}(X) \otimes P \to A$ such that

$$\begin{array}{ccccc} I \otimes P & \xrightarrow{nil \otimes P} & \mathrm{M}(X) \otimes P & \xleftarrow{cons \otimes P} & X \otimes \mathrm{M}(X) \otimes P \\ {\scriptstyle \cong}\downarrow & & \vdots{\scriptstyle \mathrm{it}(n,c,\alpha)} & & \downarrow{\scriptstyle X \otimes \mathrm{it}(n,c,\alpha)} \\ P & \xrightarrow[p]{} & A & \xleftarrow[c]{} & X \otimes A \end{array} \qquad \begin{array}{ccccc} T\big(\mathrm{M}(X)\big) \otimes P & \xrightarrow{st_{\mathrm{M}(X),P}} & T(\mathrm{M}(X) \otimes P) & \xrightarrow{T(\mathrm{it}(n,c,\alpha))} & TA \\ {\scriptstyle \tau \otimes P}\downarrow & & & & \downarrow{\scriptstyle \alpha} \\ \mathrm{M}(X) \otimes P & & \xrightarrow[\mathrm{it}(n,c,\alpha)]{} & & A \end{array}$$

Thus, definitions by pure iteration in this setting are required to be left algebra homomorphisms (recall (1)).

Monoids with algebraic structure. Generalising from [15], we define the notion of monoid equipped with compatible monadic algebraic structure. This has also recently been considered by Piróg [39] under the terminology of Eilenberg-Moore monoid.

▶ **Definition 5.2.** Let (T, st) be a strong monad on a monoidal category $(\mathcal{C}, I, \otimes)$. A *$T$-monoid* is an object C equipped with a monoid structure $(I \xrightarrow{e} C \xleftarrow{m} C \otimes C)$ and a T-algebra structure $(c : TC \to C)$ compatible in the sense that the monoid multiplication is a left homomorphism; that is, the following diagram commutes:

$$\begin{array}{ccccc} T(C) \otimes C & \xrightarrow{st_{C,C}} & T(C \otimes C) & \xrightarrow{Tm} & TC \\ {\scriptstyle c \otimes C}\downarrow & & & & \downarrow{\scriptstyle c} \\ C \otimes C & & \xrightarrow[m]{} & & C \end{array}$$

Homomorphisms of T-monoids are morphisms of the underlying objects that are both monoid and T-algebra homomorphisms. We write $T\text{-}Mon(\mathcal{C})$ for the category of T-monoids and their homomorphisms.

▶ Remark. The construction $T\text{-}Mon(\mathcal{C})$ generalises both the category of algebras for a monad $T\text{-}Alg$ and the category of monoids $Mon(\mathcal{C})$. The former is available for any category $\mathcal{C}$ with finite coproducts by taking these as the monoidal structure; the latter arises by restricting attention to identity monads.

The notion of T-monoid is essentially an extension of the notion of F-monoid for a strong

endofunctor F on a monoidal category $\mathcal{C}$, introduced in [15, Section 4], to incorporate further algebraic structure. The original notion is typically subsumed by the notion of T_F-monoid for T_F the monad on $\mathcal{C}$ induced by a left adjoint to the forgetful functor $F\text{-}alg \to \mathcal{C}$. Indeed, consider the following proposition.

▶ **Proposition 5.3.** *Let $(\mathcal{C}, I, \otimes)$ be a monoidal category and F a strong endofunctor on it.*

1. *If the forgetful functor $F\text{-}alg \to \mathcal{C}$ has a left adjoint then it is monadic.*
2. *If $\otimes$ is left-closed and the forgetful functor $F\text{-}alg \to \mathcal{C}$ has a left adjoint then the induced monad T_F is canonically strong.*
3. *Assuming that $\mathcal{C}$ has finite coproducts and ω-colimits that are preserved by $(-) \otimes P$ for all $P \in \mathcal{C}$, if F is ω-cocontinuous then the forgetful functor $F\text{-}alg \to \mathcal{C}$ has a left adjoint and the induced monad T_F is canonically strong.*
4. *In either of the situations (2) and (3) above, the notions of T_F-monoid and of F-monoid coincide.*

Wolff [48] noticed that, for T a strong monad on a monoidal category $(I, \otimes)$, TI is a monoid. The notion of T-monoid allows us to sharpen this result.

▶ **Lemma 5.4.** *For every strong monad (T, st, η, μ) on a monoidal category $(\mathcal{C}, I, \otimes)$, the free T-algebra (TI, μ_I) equipped with unit $\eta_I : I \to TI$ and multiplication*

$$(TI) \otimes (TI) \xrightarrow{st_{I,TI}} T(I \otimes TI) \cong T(TI) \xrightarrow{\mu_I} TI$$

is a T-monoid. In fact, it is initial in $T\text{-}Mon(\mathcal{C})$.

The original example in which the lemma above was applied essentially considered the monad freely generated by a binding-signature endofunctor. In that context, the initiality property yields initial-algebra semantics for abstract syntax with variable binding and substitution. See [15, 10] for details.

For further examples of T-monoids consider the following.

▶ **Proposition 5.5.** *For every monoid M in a monoidal category $(\mathcal{C}, I, \otimes)$, we have a strong monad $M^{\otimes}$ $\big($with underlying endofunctor $M \otimes (-)\big)$ for which $M^{\otimes}\text{-}Alg$ is the category of left M-actions, and $M^{\otimes}\text{-}Mon(\mathcal{C})$ is isomorphic to the slice category $M/Mon(\mathcal{C})$.*

T-list objects induce free T-monoids. For a strong ω-cocontinuous monad T on a finitely and ω-chain cocomplete monoidal category $(I, \otimes)$ in which $\otimes$ is ω-cocontinuous, free T-monoids can be shown to exist (for instance, by applying the theory of [13]). However, no explicit description for this construction is available. The main purpose of this section is to establish one under practical hypotheses.

First we observe that the situation for list objects generalises in the expected way (compare Lemma 3.4).

▶ **Lemma 5.6.** *Let T be a strong monad on a monoidal category $\mathcal{C}$.*

1. *T-list objects induce free T-monoids.*
2. *If $\mathcal{C}$ has T-list objects then the forgetful functor $T\text{-}Mon(\mathcal{C}) \to \mathcal{C}$ is monadic inducing a free T-monoid (or T-list) monad M_T.*
3. *When the monoidal structrure is cartesian closed, the monad M_T is strong.*

The proof that list objects induce free monoids relies on parametrised initiality to define the multiplication map by iteration and on the uniqueness property of mediating maps to show that the monoid laws and universal property hold. The proof of Lemma 5.6(1) follows the same approach, except one needs to check the relevant maps are left homomorphisms.

In the light of the lemma, the problem of constructing free T-monoids can be reduced to

that of constructing T-list objects. For an object X in a monoidal category $(\mathcal{C}, I, \otimes)$ with binary coproducts $(+)$, we introduce the functor

$$F(P, C) := T(P + X \otimes C) \ ,$$

consider a parameterised initial $F(I, -)$-algebra

$$\mathrm{M}X := \mu A.\, T(I + X \otimes A) \ ,$$

and proceed to equip it with a T-list object structure on X. We construct the list algebra $(I \xrightarrow{nil} \mathrm{M}X \xleftarrow{cons} X \otimes \mathrm{M}X)$ as

$$nil := \big(I \xrightarrow{\mathrm{inl}} I + X \otimes \mathrm{M}X \xrightarrow{\eta} T(I + X \otimes \mathrm{M}X) \xrightarrow{\varphi} \mathrm{M}X\big)$$

and

$$cons := \big(X \otimes \mathrm{M}X \xrightarrow{\mathrm{inr}} I + X \otimes \mathrm{M}X \xrightarrow{\eta} T(I + X \otimes \mathrm{M}X) \xrightarrow{\varphi} \mathrm{M}X\big)$$

where $\varphi : T(I + X \otimes \mathrm{M}X) \xrightarrow{\cong} \mathrm{M}X$ is the structure map for $\mathrm{M}X$. This has a T-algebra structure, given by

$$\widetilde{\mu} := \big(\ T(\mathrm{M}X) \xrightarrow[\cong]{T\varphi^{-1}} TT(I + X \otimes \mathrm{M}X) \xrightarrow{\mu} T(I + X \otimes \mathrm{M}X) \xrightarrow[\cong]{\varphi} \mathrm{M}X\ \big) \ .$$

▶ **Lemma 5.7.** *For a strong monad (T, st) on a monoidal category $(I, \otimes)$ with binary coproducts $(+)$ preserved by the endofunctor $(-) \otimes P$ for every object P, the structure*

$$\begin{array}{ccccc} & & T(\mathrm{M}X) & & \\ & & \downarrow{\scriptstyle \widetilde{\mu}} & & \\ I & \xrightarrow[nil]{} & \mathrm{M}X & \xleftarrow[cons]{} & X \otimes \mathrm{M}X \end{array} \tag{3}$$

is a T-list object.

In particular, the mediating map $\mathrm{it}(n, c, \alpha) : (\mathrm{M}X) \otimes P \to A$ to a parametrised T-list algebra $(P \xrightarrow{n} A \xleftarrow{c} X \otimes A, TA \xrightarrow{\alpha} A)$ arises by parametrised initiality (recall Definition 4.5) as follows

$$\begin{array}{ccccc} T(I + X \otimes \mathrm{M}X) \otimes P & \xrightarrow{T(\delta^+) \circ st} & T(I \otimes P + X \otimes (\mathrm{M}X) \otimes P) & \xrightarrow{T(I \otimes P + X \otimes \mathrm{it}(n,c,\alpha))} & T(I \otimes P + X \otimes A) \\ \downarrow{\scriptstyle \varphi \otimes P} & & & & \downarrow{\scriptstyle T[n \circ \cong, c]} \\ & & & & TA \\ & & & & \downarrow{\scriptstyle \alpha} \\ (\mathrm{M}X) \otimes P & & \xrightarrow[\mathrm{it}(n,c,\alpha)]{} & & A \end{array}$$

where δ^+ denotes the inverse of the canonical map $(- \otimes P) + (= \otimes P) \to (- + =) \otimes P$.

We thus have the following result.

▶ **Theorem 5.8.** *Let T be a strong monad on a monoidal category $(\mathcal{C}, I, \otimes)$ with binary coproducts $(+)$. If*

1. *for every $P \in \mathcal{C}$, the endofunctor $(-) \otimes P$ preserves binary coproducts, and*
2. *every functor $F_X(-, =) := T(- + X \otimes =)$ for $X \in \mathcal{C}$ has a parametrised initial $F_X(I, -)$-algebra*

then $\mathcal{C}$ has T-list objects as in (3) above and, thereby, the free T-monoid monad M_T.

▶ Remark. We have that $\mu A.\, T(I + X \otimes A) \cong T\big(\mu A.\, (I + X \otimes TA)\big)$ by the Rolling Rule [3, Theorem 4], which gives the description of free T-monoids of Piróg [39].

▶ **Example 5.9.** **1.** The free Id-monoid monad is the list monad.

2. One may apply Theorem 5.8 in the situation of Proposition 5.3 to describe the free F-monoid for a strong endofunctor F. This is given by

$$\mu A.\, T_F(I + X \otimes A) = \mu A.\, \mu B.\, I + X \otimes A + F(B)$$

or, by the Diagonal Rule [3, Theorem 16], equivalently by

$$\mathrm{M}_{T_F}(X) := \mu C.\, I + X \otimes C + F(C) \ .$$

Thus, we recover a slightly restricted version of a corresponding result for endofunctors with *pointed strength* of Fiore [10, Theorems 2 and 3] (see also Section 7.3 below). In the context of the motivating application of [10], this yields initial-algebra semantics for abstract syntax with variable binding and parameterised metavariables, with their associated operations of capture-avoiding substitution and of meta-substitution. See [17, 10] for details.

For our applications, we are interested in using Theorem 5.8 in two distinct concrete scenarios, encapsulated in the corollaries below.

▶ **Corollary 5.10.** *Let T be a strong monad on a left-closed monoidal category $(\mathcal{C}, I, \otimes)$ with binary coproducts $(+)$. If every endofunctor $T(I + X \otimes -)$ for $X \in \mathcal{C}$ has an initial algebra then $\mathcal{C}$ has T-list objects and, thereby, the free T-monoid monad M_T.*

▶ **Corollary 5.11.** *Let T be a strong monad on a monoidal category $(\mathcal{C}, I, \otimes)$ with finite coproducts $(0, +)$ and ω-colimits. If*

1. *for every $P \in \mathcal{C}$, the endofunctor $(-) \otimes P$ preserves finite coproducts, and*

2. *the functors $\otimes$ and T are ω-cocontinuous*

then $\mathcal{C}$ has T-list objects and, thereby, the free T-monoid monad M_T.

▶ **Example 5.12.** In view of Proposition 5.5 and Corollary 5.10, for every monoid M in a left-closed monoidal category $(\mathcal{C}, I, \otimes)$ with binary coproducts $(+)$ the free M-pointed object in $Mon(\mathcal{C})$ on an object $X \in \mathcal{C}$ is given by $\mu A.\, M \otimes (I + X \otimes A)$.

6 Strong algebraic structure

Our discussion so far has considered algebraic structure abstractly as encapsulated by the notion of monad. Our main interest is in monads that are strong, and this section examines these as they arise from algebraic theories. Our aim here is not to treat these in full generality (as *e.g.* in the enriched setting [41]) but to focus attention on two traditionally important contexts: cartesian and monoidal. By way of introduction, Section 6.1 recalls the setting of (cartesian) algebraic theories as studied in universal algebra [28, 8, 40]. Then, in Section 6.2, we develop the operadic [5, 36] or linear [27] setting. Our analysis unveils the notion of nsr (or near semiring) category (Definition 6.2), a categorification of the algebraic structure of near semiring [47], as the basic categorical framework in which to consider monoidal (or linear) algebraic models.

6.1 Algebraic theories

A *Lawvere theory* [28] is seen in universal algebra as an *abstract clone* of operations (see *e.g.* [8, 40]). This is a family of sets $O = \{O_n\}_{n \in \mathbb{N}}$ together with *variables* $x_i^{(n)} \in O_n$ $(1 \leq i \leq n \in \mathbb{N})$ and *substitution* operations $s_{m,n} : O_m \times {O_n}^m \to O_n$ $(m, n \in \mathbb{N})$ subject to identity and associativity laws. A *map* $f : O \to O'$ of abstract clones is a family of functions $f_n : O_n \to O'_n$ $(n \in \mathbb{N})$ preserving variables and commuting with substitution.

A main example of an abstract clone is the *concrete clone* of operations (or clone of endomorphisms) on an object in a cartesian category $\mathcal{C}$. For an object $X \in \mathcal{C}$ this is defined as $E(X) = \{\, \mathcal{C}(X^n, X) \,\}_{n\in\mathbb{N}}$ with variables given by projections and substitution by composition. An *O-algebra* (or model) of an abstract clone O is an object X together with a structure map $O \to E(X)$ of abstract clones. An algebra *homomorphism* $h : (X, x) \to (Y, y)$ is a map $h : X \to Y$ such that $h \circ x_o = y_o \circ h^n : X^n \to Y$ for all $n \in \mathbb{N}$ and $o \in O_n$. We write *O-Alg* for the corresponding category. If $\mathcal{C}$ is cocomplete and the product $(\times)$ is separately cocontinuous in each argument, the forgetful functor *O-Alg* $\to \mathcal{C}$ is monadic and the induced monad T_O is given on an object $X \in \mathcal{C}$ by the coequaliser

$$\coprod_{m,n\in\mathbb{N}} \coprod_{\rho\in\mathbb{F}([m],[n])} O_m \cdot X^n \underset{[\iota_n \circ (O_\rho \cdot X^n)]_{\rho:[m]\to[n]}}{\overset{[\iota_m \circ (O_m \cdot X^\rho)]_{\rho:[m]\to[n]}}{\rightrightarrows}} \coprod_{\ell\in\mathbb{F}} O_\ell \cdot X^\ell \twoheadrightarrow T_O(X)$$

where we write $\cdot$ for the functor $\mathbf{Set} \times \mathcal{C} \to \mathcal{C} : (I, C) \mapsto \coprod_{i\in I} C$, and where we let $\mathbb{F}$ denote the category of finite cardinals $[n] := \{1, \ldots, n\}$ $(n \in \mathbb{N})$ and functions between them and set $O_\rho := s_{m,n}\big(-, (x^{(n)}_{\rho 1}, \ldots, x^{(n)}_{\rho m})\big) : O_m \to O_n$ for all $\rho \in \mathbb{F}([m], [n])$. Intuitively, thus, the coequaliser identifies $\iota_o(-_{\rho 1}, \ldots, -_{\rho m})$ and $\iota_{O_\rho(o)}(-_1, \ldots, -_n)$ for all $m, n \in \mathbb{N}$, $\rho \in \mathbb{F}([m], [n])$, and $o \in O_m$.

Important to our concerns here, when $\mathcal{C}$ is furthermore monoidal and the tensor product $(\otimes)$ is left cocontinuous, the free monad T_O is canonically strong, with strength components given by the unique map making the following diagram commute

$$\begin{array}{ccc} \coprod_{\ell\in\mathbb{F}} O_\ell \cdot (X^\ell \otimes P) & \twoheadrightarrow & T_O(X) \otimes P \\ \Big\downarrow{\scriptstyle \coprod_{\ell\in\mathbb{F}} O_\ell \cdot \langle \pi_1 \otimes P, \ldots, \pi_\ell \otimes P\rangle} & & \Big\downarrow \\ \coprod_{\ell\in\mathbb{F}} O_\ell \cdot (X \otimes P)^\ell & \twoheadrightarrow & T_O(X \otimes P) \end{array}$$

We can therefore consider T_O-monoids. At any rate, one can also give an equivalent direct definition for abstract clones (or for their equational presentations).

▶ **Definition 6.1.** An O-monoid, for an abstract clone O in a monoidal category with finite products, is an object C equipped with a monoid structure $(I \xrightarrow{e} C \xleftarrow{m} C \otimes C)$ and an O-algebra structure $\gamma : O \to E(C)$ compatible in the sense that the monoid multiplication is a left algebra homomorphism; that is, the diagram

$$\begin{array}{ccccc} C^n \otimes C & \xrightarrow{\langle \pi_1 \otimes C, \ldots, \pi_n \otimes C\rangle} & (C \otimes C)^n & \xrightarrow{m^n} & C^n \\ \Big\downarrow{\scriptstyle \gamma_o \otimes C} & & & & \Big\downarrow{\scriptstyle \gamma_o} \\ C \otimes C & & \xrightarrow[m]{\qquad\qquad\qquad} & & C \end{array}$$

commutes for all $n \in \mathbb{N}$ and $o \in O_n$.

In the next section we shall see that this construction falls within a more general framework.

6.2 Linear algebraic theories

Operads are a well-established formalism for investigating monoidal or linear algebraic structure, see *e.g.* [34]. In their basic form, they come in two guises: plain and symmetric. Here, for simplicity, we limit the discussion to the plain case; the development can be suitably adapted to the symmetric case. A *plain operad* is a family of sets $O = \{\, O_n \,\}_{n\in\mathbb{N}}$ together with

a *variable* $x \in O_1$ and *linear substitution* operations $s_\ell^{(n_1,\dots,n_\ell)} : O_\ell \times \prod_{i\in[\ell]} O_{n_i} \to O_{\Sigma_{i\in[\ell]} n_i}$ $(\ell, n_1, \dots, n_\ell \in \mathbb{N})$ subject to identity and associative laws. A map $f : O \to O'$ of plain operads is a family of functions $f_n : O_n \to O'_n$ $(n \in \mathbb{N})$ preserving the variable and commuting with substitution. (See *e.g.* [31, Chapter 2]).

A main example of a plain operad is the *endomorphism operad* on an object in a monoidal category. For an object X in a monoidal category $(\mathcal{C}, J, *)$ this is defined as $E(X) = \{\, \mathcal{C}(X^{*n}, X) \,\}_{n\in\mathbb{N}}$ with variable given by the identity and substitution by multi-composition. The category of algebras for an operad O is defined analogously to that for abstract clones, and equally denoted *O-Alg*. Without going into details, the theory of [13] may be applied to show that when $\mathcal{C}$ is cocomplete and $*$ is ω-cocontinuous, the forgetful functor $O\text{-}Alg \to \mathcal{C}$ is monadic. We are interested in establishing a framework in which the induced free O-algebra monads are strong. This is provided by the following notion.

▶ **Definition 6.2.** An *nsr-category* is a category equipped with two monoidal structures, $(I, \otimes)$ and $(J, *, \lambda, \rho, \alpha)$, together with a *distributive structure* of the former over the latter specified by $\otimes$-strengths for J and $*$

$$\delta_P^{(0)} : J \otimes P \to J \ , \quad \delta_{A,B,P}^{(2)} : (A * B) \otimes P \to (A \otimes P) * (B \otimes P)$$

such that λ, ρ, α are $\otimes$-strong.

▶ Remark. As in [42], the above terminology is motivated by the algebraic structure known as a *near semiring* [47], consisting of two monoid structures on the same carrier that are subject to one-sided annihilation and distributivity laws of one structure over the other.

▶ **Example 6.3.** **1.** For a monoidal category $(\mathcal{C}, I, \otimes)$ with finite coproducts $(0, +)$, the opposite category $\mathcal{C}^{\mathrm{op}}$ has a canonical distributive structure

$$0 \otimes P \leftarrow 0 \ , \quad (A + B) \otimes P \leftarrow (A \otimes P) + (B \otimes P)$$

of $\otimes$ over $+$. Hence, whenever the above maps are invertible, we also have a distributive structure of $\otimes$ over $+$ in $\mathcal{C}$.

2. Cartesian monoidal categories have a distributive structure of the monoidal structure over the cartesian structure. This was implicitly used in the previous section.

3. The category of endofunctors of a monoidal category has a distributive structure of the composition monoidal structure over the pointwise monoidal structure.

4. Joyal's category of combinatorial species of structures [20] forms an nsr-category with the substitution and multiplication monoidal structures. This was implicitly used in [11].

The natural ambient universe for monoids with linear algebraic structure (either given by an operad or their equational presentations) is that of nsr-categories.

▶ **Definition 6.4.** An O-monoid, for an operad O in an nsr-category $(I, \otimes, J, *, \delta)$, is an object C equipped with a $\otimes$-monoid structure $(I \xrightarrow{e} C \xleftarrow{m} C \otimes C)$ and an O-algebra $*$-structure $\gamma : O \to E(C)$, compatible in the sense that the monoid multiplication is a left algebra homomorphism; that is, the diagram

$$\begin{array}{ccccc} C^{*n} \otimes C & \xrightarrow{\delta^{(n)}} & (C \otimes C)^{*n} & \xrightarrow{m^{*n}} & C^{*n} \\ {\scriptstyle \gamma_o \otimes C}\downarrow & & & & \downarrow{\scriptstyle \gamma_o} \\ C \otimes C & & \xrightarrow[m]{\qquad\qquad} & & C \end{array}$$

commutes for all $n \in \mathbb{N}$ and $o \in O_n$, where $\delta^{(n)}_{C_1,\dots,C_n,P}$ stands for the iterated strength $(C_1 * \cdots * C_n) \otimes P \to (C_1 \otimes P) * \cdots * (C_n \otimes P)$.

▶ Remark 6.5. In well-behaved settings, one may typically rephrase O-monoids in terms of T-monoids. Specifically, in a cocomplete nsr-category $(\mathcal{C}, \otimes, *)$ with $\otimes$ left-cocontinuous and $*$ ω-cocontinuous, the forgetful functor $O\text{-}Alg \to \mathcal{C}$ is monadic and the induced free O-algebra monad T_O is $\otimes$-strong. Furthermore, the notions of T_O-monoid and of O-monoid coincide. For instance, this is the case in the example featured in [11, Example 5.13(3)].

We conclude the section by discussing an important concrete example.

▶ **Lemma 6.6.** *Let $(\mathcal{C}, \otimes, *)$ be an nsr-category.*

1. *Assuming that $\mathcal{C}$ has $*$-list objects and that $\otimes$ is left-closed, the $*$-list monad L_* (arising from Lemma 3.4) is $\otimes$-strong.*
2. *Assuming that $\mathcal{C}$ has finite coproducts and ω-colimits, that $\otimes$ is left ω-cocontinuous, that $*$ is ω-cocontinuous, and that $(-) * P$ preserves finite coproducts for all $P \in \mathcal{C}$, the $*$-list monad L_* (arising from Corollary 4.9 and Lemma 3.4) is $\otimes$-strong.*

▶ **Example 6.7.** The notion of O-monoid for O the theory of monoids in an nsr-category $(\mathcal{C}, I, \otimes, J, *, \delta)$ amounts to an object C equipped with two monoid structures $(I \xrightarrow{1} C \xleftarrow{\times} C \otimes C)$ and $(J \xrightarrow{0} C \xleftarrow{+} C * C)$ such that

$$\begin{array}{ccc} J \otimes C & \xrightarrow{\delta^{(0)}_C} & J \\ {\scriptstyle 0\otimes C}\downarrow & & \downarrow{\scriptstyle 0} \\ C \otimes C & \xrightarrow[\times]{} & C \end{array} \qquad \begin{array}{ccccc} (C * C) \otimes C & \xrightarrow{\delta^{(2)}_{C,C,C}} & (C \otimes C) * (C \otimes C) & \xrightarrow{\times * \times} & C * C \\ {\scriptstyle +\otimes C}\downarrow & & & & \downarrow{\scriptstyle +} \\ C \otimes C & & \xrightarrow[\times]{} & & C \end{array}$$

These two diagrams respectively express one-sided annihilation and distributivity laws. As such, we refer to these structures as *nsr (or near-semiring) objects*. Nsr-objects in nsr-categories have already been considered by Rivas *et al.* [42].

For a $\otimes$-strong $*$-list monad L_* (arising, for instance, as in Lemma 6.6), L_*-monoids are precisely nsr-objects. In the context of Theorem 5.8 we therefore have that the free nsr-object on an object X is given by

$$\mathrm{Nsr}(X) := \mu A.\, \mathrm{L}_*(I + X \otimes A) \ .$$

▶ **Example 6.8.** For a strong monad T on a monoidal category $(\mathcal{C}, I, \otimes)$, we have a strong monad T° (with underlying endofunctor $T \circ (-)$) on the monoidal category $(\mathrm{Endo}(\mathcal{C}), I, \otimes)$ of endofunctors with the pointwise monoidal structure, and $\mathrm{M}_{T^\circ}(\mathrm{Id}) = \mathrm{M}_T$; that is, the free T°-monoid on the identity is the free T-monoid monad. Furthermore, it is an nsr-object in the nsr-category $(\mathrm{Endo}(\mathcal{C}), \mathrm{Id}, \circ, I, \otimes)$ (recall Example 6.3(3)).

7 Applications and related work

Structures akin to the ones studied in this paper have been considered in a variety of contexts. We have already mentioned the algebraic approach to abstract syntax and variable binding of [15, 10] (see also [18]); we also touch upon this below. Before that, we will discuss applications to programming theory, in semantics and functional programming, and to higher-dimensional algebra, in the opetopic approach [4, 45]. Our work connects these developments and opens up possibilities for interaction between them.

7.1 Semantics and programming

Let us uncover some examples of T-list objects in semantics and programming.

Natural Numbers Objects. Following Lawvere [29], we refer to the T-list object on the unit object as a T-NNO. This general notion provides a uniform sense in which the various domains of natural numbers that are needed in denotational semantics are Natural Numbers Objects for suitable algebraic notions of primitive recursion. Indeed, Corollary 5.11 has the following consequences in the category of cpos and continuous functions: the NNO for the cartesian monoidal structure is the *natural numbers* $\mu A.\, 1 + A$ and the lifting-NNO for the cartesian monoidal structure is the *lazy natural numbers* $\mu A.\, (1 + A)_\perp$. Moreover, the lifting-NNO for the cocartesian monoidal structure is the *strict natural numbers* $\mu A.\, A_\perp$.

List transformer. Corollary 5.10 gives a universal property for the monadic *list transformer* of Jaskelioff [19, Example 4.7] (see also [16]): it is a T-list object construction and the free T-monoid monad in the bicartesian closed setting. Indeed, assuming the needed initial algebras exist, the list transformer $\mathrm{Lt}(T)$ of a strong monad T on a cartesian closed category with binary coproducts is defined as $\mathrm{Lt}(T)\, X := \mu A.\, T(1 + X \times A)$. This result was first established by Kammar [22] using a type-theoretic internal language. It follows from Example 6.8 that the list transformer produces instances of the Haskell MonadPlus type class also discussed below.

7.2 Free monoids with linear algebraic structure in nsr-categories

The examples of this section arise from the linear algebraic framework of Section 6.2, where instances of free T-monoids have been considered in functional programming and higher-dimensional algebra.

7.2.1 Functional programming

We consider an nsr-category $(\mathcal{C}, I, \otimes, J, *)$ with left-closed tensor products and binary coproducts, placing ourselves in the setting of Corollary 5.10. This is the setting assumed in the context of functional programming, say in Haskell.

For nsr-categories with structure as in Example 6.3(2) (*i.e.* with $(J, *) = (1, \times)$), Rivas *et al.* [42] constructed free nsr-objects as in Example 6.7. Specialising this construction to the nsr-category of endofunctors with structure $(\mathrm{Id}, \circ, 1, \times)$ (recall Example 6.3(3)) over a cartesian category $(1, \times)$ they obtained, and programmed with, instances of the Haskell MonadPlus type class freely generated from an endofunctor by means of the recursive type $\mathrm{Mp}(F)\, X := \mu A.\, \mathrm{List}(X + FA)$. Further examples in this direction may be found in [46].

For the purpose of combinatorial search, Spivey [44] proposed extending the Haskell MonadPlus type class to a Bunch type class by adding a unary *wrap* operation subject to an equation. In our linear algebraic framework of Section 6.2, this precisely amounts to extending the theory of monoids with an extra unary operation. The free algebras for this theory are given by $T(X) := \mu A.\, J + (X + A) * A$ with induced free T-monoids $\mathrm{Bun}(X) := \mu A.\, J + (I + X \otimes A + A) * A$. Specialising these constructions to the monadic setting of functional programming, one obtains freely generated instances of the Haskell Bunch type class by means of the recursive type $\mathrm{Bun}(F)\, X := \mu A.\, 1 + X \times A + F(A) \times A + A \times A$.

7.2.2 Higher-dimensional algebra

Let $(\mathcal{C}, I, \otimes, J, *)$ be a cocomplete nsr-category (Definition 6.2) where $\otimes$ is left cocontinuous and right ω-cocontinuous and $*$ is ω-cocontinuous. Then, the free O-algebra monad T_O (Remark 6.5) is ω-cocontinuous and Corollary 5.11 applies.

Restricting attention to the theory of monoids, it is enough to require that both tensor products be ω-cocontinuous and that they preserve finite coproducts in their first argument for free monoids to be iteratively constructed as list objects. In fact, this is precisely the context in which the work of Szawiel and Zawadowski [45] takes place. In particular, they define the *web monoid* as $\mathrm{L}_*(I)$ and establish a universal property for it. Our results sharpen and generalise theirs: by establishing that the web monoid is the initial nsr-object $\mathrm{M}_{\mathrm{L}_*}(0)$, and by providing a general construction of free nsr-objects $\mathrm{M}_{\mathrm{L}_*}(X)$.

7.3 Second-order algebraic theories

The application of the theory of monoids with algebraic structure for an endofunctor or a monad in the study of abstract syntax with variable binding [15] and with parametrised metavariables [17, 10] crucially requires a slightly more general theory than the one expounded upon here (recall Example 5.9(2)). Indeed, because of the presence of binding and the need for capture-avoidance in substitution, the semantic endofunctors and monads freely generated from the syntax only admit a *pointed strength* (see [10] and [14]); that is, a strength of the form

$$st_{X,I\to P} : T(X) \otimes P \to T(X \otimes P)$$

only available for parameters P equipped with a point $(I \to P)$. The results of the present work generalise not only to this setting but, in fact, to a more comprehensive one involving monoidal actions (see [10]). As such, they will be presented elsewhere.

8 Concluding remarks

We have introduced the notions of T-list object and T-monoid, developing some of their related basic theory and discussing applications. In particular, in Section 5, we have established that T-list objects yield free T-monoids and have provided an explicit description for them:

$$\mathrm{M}_T(X) := \mu A.\, T(I + X \otimes A)$$

that we have shown to be available in common practical scenarios. Our theory captures all the examples of this construction that we are aware of. In this respect, the study of parametrised initiality of Section 4 is crucial for cases where the tensor product is not left-closed. Considering compatible algebraic and monoid structures from the perspective of T-monoids leads naturally to the notion of nsr-category. These categories provide a common framework for applications in a diverse range of settings, some of which we have presented in Sections 6 and 7.

Our work opens up intriguing possibilities for interaction between a priori separate research areas. For instance, the web monoid [45], mentioned in Section 7.2.2, is used to recast a version of Baez and Dolan's construction of opetopes [4] (see also [26]). The full generality of our linear algebraic framework (Section 6.2) may be useful in porting this approach to new applications. Our perspective on the structure of opetopes (Section 7.2.2) provides an inductively defined construction for them that may be amenable to formalisation in proof assistants. At any rate, it has already led to a unification of the algebraic aspects of the theory of abstract syntax and the theory of opetopes [12].

Acknowledgements. We are grateful to Zawadowski for bringing his work with Szawiel [45] to our attention, and both of them for discussions about it. We are also grateful to Kammar for his note [22] and to Rivas for comments. We would like to thank the anonymous reviewers for their helpful suggestions.

References

1 J. Adámek. Free algebras and automata realizations in the language of categories. *Commentationes Mathematicae Universitatis Carolinae*, 015(4):589–602, 1974.

2 S. Alves, M. Fernández, M. Florido, and I. Mackie. Linear recursive functions. In *Rewriting, Computation and Proof: Essays Dedicated to Jean-Pierre Jouannaud on the Occasion of His 60th Birthday*, pages 182–195. Springer, 2007.

3 R. Backhouse, M. Bijsterveld, R. van Geldrop, and J. van der Woude. Categorical fixed point calculus. In *Category Theory and Computer Science, CTCS 1995*, volume 953 of *Lecture Notes in Computer Science*, pages 159–179. Springer, 1995.

4 J. C. Baez and J. Dolan. Higher-dimensional algebra III. n-categories and the algebra of opetopes. *Advances in Mathematics*, 135(2):145–206, 1998. doi:10.1006/aima.1997.1695.

5 J. M. Boardman and R. M. Vogt. Homotopy-everything H-spaces. *Bull. Amer. Math. Soc.*, 74(6):1117–1122, 11 1968.

6 A. Burroni. Récursivité graphique (1^e partie) : catégorie des fonctions récursives primitives formelles. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 27(1):49–79, 1986.

7 J.R.B. Cockett. List-arithmetic distributive categories: Locoi. *Journal of Pure and Applied Algebra*, 66(1):1–29, 1990. doi:10.1016/0022-4049(90)90121-W.

8 P.M. Cohn. *Universal algebra*. Reidel, 2nd edition, 1981.

9 M. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. Cambridge University Press, 1996.

10 M. Fiore. Second-order and dependently-sorted abstract syntax. In *Proceedings of the 2008 23rd Annual IEEE Symposium on Logic in Computer Science*, LICS '08, pages 57–68. IEEE Computer Society, 2008. doi:10.1109/LICS.2008.38.

11 M. Fiore. An equational metalogic for monadic equational systems. *Theory and Applications of Categories*, 27(18):464–492, 2013.

12 M. Fiore. An algebraic combinatorial approach to opetopic structure. Notes and talk, *Seminar on Higher Structures*, Program on Higher Structures in Geometry and Physics, Max Planck Institute for Mathematics, February–March 2016.

13 M. Fiore and C.-K. Hur. On the construction of free algebras for equational systems. *Theoretical Computer Science*, 410(18):1704–1729, 2009. doi:10.1016/j.tcs.2008.12.052.

14 M. Fiore and C.-K. Hur. Second-order equational logic. In *Computer Science Logic, CSL 2010*, volume 6247 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2010.

15 M. Fiore, G. Plotkin, and D. Turi. Abstract syntax and variable binding. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, LICS'99. IEEE Computer Society, 1999.

16 G. Gonzalez. The list-transformer package, 2016. URL: https://hackage.haskell.org/package/list-transformer.

17 M. Hamana. Term rewriting with variable binding: An initial algebra approach. In *Proceedings of the 5th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming*, PPDP'03, pages 148–159. ACM, 2003.

18 A. Hirschowitz and M. Maggesi. Modules over monads and linearity. In *WoLLIC 2007*, number 4576 in Lecture Notes in Computer Science, pages 218–237. Springer, 2005.

19 M. Jaskelioff. *Lifting of Operations in Modular Monadic Semantics*. PhD thesis, University of Nottingham, 2009.

20 A. Joyal. Foncteurs analytiques et espèces de structures. In *Combinatoire Énumérative*, volume 1234 of *Lecture Notes in Mathematics*, pages 126–159. Springer, 1986.

21 A. Joyal. The Gödel incompleteness theorem, a categorical approach (abstract). *Cah. Top. Géom. Diff. Cat.*, 16(3), 2005. Short abstract of the talk given at the International conference *Charles Ehresmann: 100 ans*, Amiens, 7–9 October, 2005.

22 O. Kammar. Algebraic construction of the list transformer. Private communication, 2014.

23 G. M. Kelly. A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on. *Bulletin of the Australian Mathematical Society*, 22(1):1–83, 1980. doi:10.1017/S0004972700006353.

24 A. Kock. Bilinearity and cartesian closed monads. *Mathematica Scandinavica*, 29(2):161–174, 1971.

25 A. Kock. Strong functors and monoidal monads. *Archiv der Mathematik*, 23(1):113–120, 1972. doi:10.1007/BF01304852.

26 J. Kock, A. Joyal, M. Batanin, and J.-F. Mascari. Polynomial functors and opetopes. *Advances in Mathematics*, 224(6):2690–2737, 2010. doi:10.1016/j.aim.2010.02.012.

27 J. Lambek. Multicategories revisited. In *Categories in Computer Science and Logic: Proc. of the Joint Summer Research Conference*, pages 217–239. American Mathematical Society, 1989.

28 F. W. Lawvere. Functorial semantics of algebraic theories. *Proceedings of the National Academy of Sciences of the United States of America*, 50(5):869–872, 1963.

29 F. W. Lawvere. An elementary theory of the category of sets. *Proceedings of the National Academy of Sciences of the United States of America*, 52(6):1506–1511, 1964.

30 D. J. Lehmann and M. B. Smyth. Algebraic specification of data types: A synthetic approach. *Mathematical Systems Theory*, 14(1):97–139, 1981. doi:10.1007/BF01752392.

31 T. Leinster. *Higher Operads, Higher Categories*. Number 298 in London Mathematical Society Lecture Note Series. Cambridge University Press, 2004.

32 S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 1998.

33 M.E. Maietti. Joyal's arithmetic universe as list-arithmetic pretopos. *Theory and Applications of Categories*, 24(3):39–83, 2010.

34 M. Markl. Operads and PROPs. In *Handbook of Algebra, Volume 5*, pages 87–140. Elsevier, 2008.

35 P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, 1984.

36 J. P. May. *The Geometry of Iterated Loop Spaces*, volume 271 of *Lecture Notes in Mathematics*. Springer, 1972.

37 B. Nordström, K. Peterson, and J. Smith. *Programming in Martin Löf's Type Theory*. Clarendon Press, 1990.

38 R. Paré and L. Román. Monoidal categories with Natural Numbers Object. *Studia Logica*, 48(3):361–376, Sep 1989. doi:10.1007/BF00370829.

39 Maciej Piróg. Eilenberg-Moore monoids and backtracking monad transformers. In *Proceedings 6th Workshop on Mathematically Structured Functional Programming, MSFP@ETAPS 2016*, pages 23–56, 2016. doi:10.4204/EPTCS.207.2.

40 B. Plotkin. *Universal Algebra, Algebraic Logic, and Databases*. Springer, 1994.

41 J. Power. Enriched Lawvere theories. *Theory and Applications of Categories*, 6:83–93, 1999.

42 E. Rivas, M. Jaskelioff, and T. Schrijvers. From monoids to near-semirings: The essence of MonadPlus and Alternative. In *Proceedings of the 17th International Symposium on Principles and Practice of Declarative Programming*. ACM, 2015.

43 R. M. Robinson. Primitive recursive functions. *Bull. Amer. Math. Soc.*, 53(10):925–942, 10 1947.

44 J. M. Spivey. Algebras for combinatorial search. *J. Funct. Program.*, 19(3-4):469–487, 2009. doi:10.1017/S0956796809007321.

45 S. Szawiel and M. Zawadowski. The web monoid and opetopic sets. *Journal of Pure and Applied Algebra*, 217:1105–1140, 2013. doi:10.1016/j.jpaa.2012.09.030.

46 T. Uustalu. A divertimento on MonadPlus and nondeterminism. *Journal of Logical and Algebraic Methods in Programming*, 85(5):1086 – 1094, 2016. doi:10.1016/j.jlamp.2016.06.004.

47 W. G. Van Hoorn and B. Van Rootselaar. Fundamental notions in the theory of seminearrings. *Compositio Mathematica*, 18(1-2):65–78, 1967.

48 H. Wolff. Monads and monoids on symmetric monoidal closed categories. *Archiv der Mathematik*, 24(1):113–120, 1973. doi:10.1007/BF01228184.