Symmetric List Objects

FIONA BLACKETT, University of Strathclyde, UK
VIKRAMAN CHOUDHURY, University of Strathclyde, UK
RIN LIU, University of Strathclyde, UK

1 Introduction

Lists are a fundamental data structure, important in computer science and mathematics. They are often defined in functional programming languages as cons-lists, generated inductively from a pair of constructors $\mathsf{nil} : L(A)$ and $\mathsf{cons} : A \to L(A) \to L(A)$, and whose eliminators allow for definitions by recursion and proofs by induction.

In category theory, these lists are understood as initial algebras $\mu X.1 + A \times X$ [7, 1]. This list structure can be axiomatised more generally in the setting of a monoidal category, replacing the cartesian product with a tensor, and removing the need for a coproduct by axiomatising the nil and cons maps directly with an iteration mechanism. These list objects generalise the notion of Natural Numbers Objects [10] and were introduced in Joyal's arithmetic universes. An internal language for them was studied by Maietti [11], and other variants were explored in [4] and [8].

In all of these settings, additional axioms on the underlying category are required to reconstruct traditional list operations, such as list append (+), from these list objects. However, in the minimal setting of a monoidal category, this notion of a list object can be strengthened to that of a *parametrised* list object which supports parametrised recursion. It is shown in [6] that this yields a construction of free monoids (see also [9]) in monoidal categories. We give a string diagrammatic proof of this result, showing that if a monoidal category C has all parametrised list objects, then the forgetful functor $\mathrm{Mon}(C) \to C$ is monadic.

Finite multisets are another important data structure that are the commutative counterpart to lists; and in fact can be axiomatised with only a simple modification to these list objects. In this talk, we will study the notion of a *symmetric* list object in a symmetric monoidal category. We will identify lists that differ by a transposition of the two frontmost elements using the symmetry of the underlying category. Using this, we show the analogous commutative results that symmetric list objects are free commutative monoids, and that having all symmetric list objects yields a similar monadic adjunction between $\mathrm{CMon}(C)$ and C.

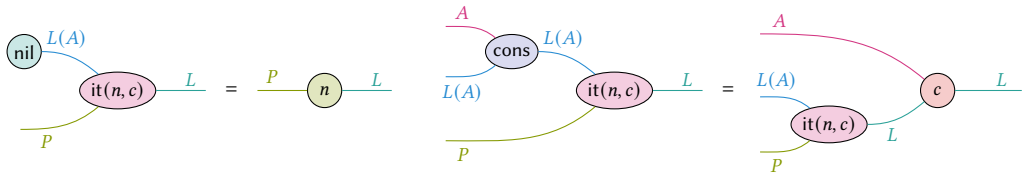
In type theory, finite multisets have been considered as setoids over lists, where the equivalence relation is bag equivalence [5, 2], or as higher inductive types of lists with an additional swap constructor to permute the two head elements [3]. These constructions use function types, but our setting only requires a symmetric monoidal structure without closure.

2 List objects

Let $C(1, \otimes)$ be a strict monoidal category.

Definition 1. A (*parametrised*) *list object* (hereafter, just *list object*) $L(A)$ on an object A consists of a pair of maps $\mathsf{nil} : 1 \to L(A)$ and $\mathsf{cons} : A \otimes L(A) \to L(A)$ such that for every object L equipped with maps $n : P \to L$ and $c : A \otimes L \to L$, there exists a unique *mediating map* or *iterator* $\mathsf{it}(n, c) : L(A) \otimes P \to L$ satisfying computation rules (a) and (b):

Authors' Contact Information: Fiona Blackett, fiona.blackett@strath.ac.uk, University of Strathclyde, UK; Vikraman Choudhury, vikraman.choudhury@strath.ac.uk, University of Strathclyde, UK; Rin Liu, rin.liu@strath.ac.uk, University of Strathclyde, UK.

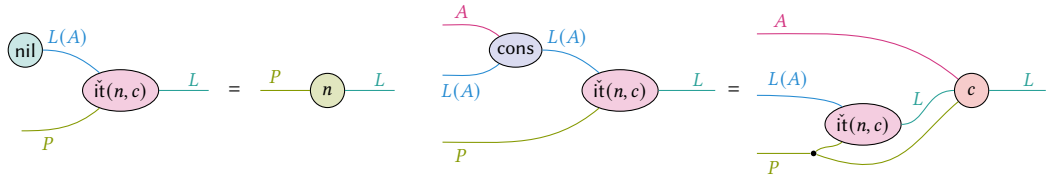


(a) nil-β (b) cons-β

Theorem 2. *The list object $L(A)$ is the free monoid on A.*

Theorem 3 (Lemma 3.4.2 [6]). *If C admits parametrised list objects for every A, then $\mathsf{Mon}(C)$ is monadic over C via the evident forgetful functor.*

Theorem 4. *If $(C, \times, 1)$ is cartesian monoidal and $L(A)$ is a list object, then $L(A)$ satisfies the following stronger universal property: for every object L equipped with maps $n : P \to L$ and $c : A \times L \times P \to L$, then there exists a doubly parametrised iterator $\check{\mathrm{it}}(n, c) : L(A) \times P \to L$, such that the following two computation rules hold:*



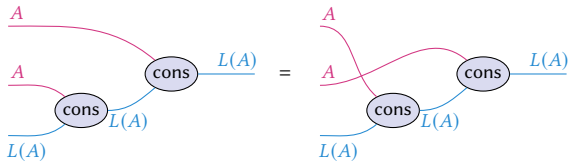
(a) nil-β (a) cons-β

Theorem 5 (Lemma 3.4.3 [6]). *If $(C, \times, 1)$ is cartesian monoidal, then the induced monad is strong.*

3 Symmetric List Objects

Hereafter, we assume C is symmetric strict monoidal, i.e. strict monoidal with a symmetric braiding.

Definition 6. A (*parametrised*) *symmetric list object* or *ilst* on A is a list object $L(A)$ with the following additional axiom and the corresponding universal property.



ilst axiom

Theorem 7. *Let $(L(A), \mathsf{nil}, \mathsf{cons})$ be a list object on an object A. Then, the following are equivalent:*

(1) $(L(A), \mathsf{nil}, \mathsf{cons})$ is a symmetric list object;
(2) $(L(A), \mathsf{nil}, +\!\!+)$ is a commutative monoid object;
(3) $(L(A), \mathsf{nil}, +\!\!+, \eta)$ is the free commutative monoid object on A;

Theorem 8. *If C admits parametrised symmetric list objects for every A, then $\mathsf{CMon}(C)$ is monadic over C via the evident forgetful functor.*

Theorem 9. *If $(C, \times, 1)$ is cartesian monoidal, then the induced monad $T : C \to C$ is strong, and further, commutative.*

Symmetric lists give an alternative way of constructing free commutative monoids in a symmetric monoidal category which is relevant to the study of the exponential modality in linear logic. In the future, we hope to compare this to other approaches, such as in [12].

References

[1] Jiří Adámek, Stefan Milius, and Lawrence S. Moss. 2025. *Initial Algebras and Terminal Coalgebras: The Theory of Fixed Points of Functors. Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge. ISBN: 978-1-108-83546-6. doi:10.1017/9781108884112.

[2] Thorsten Altenkirch, Thomas Anberrée, and Nuo Li. [n. d.] Definable Quotients in Type Theory.

[3] Vikraman Choudhury and Marcelo Fiore. 2023. Free Commutative Monoids in Homotopy Type Theory. *Electronic Notes in Theoretical Informatics and Computer Science*, Volume 1 - Proceedings of... (Feb. 22, 2023), 10492. doi:10.46298/entics.10492.

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

[5] Nils Anders Danielsson. 2012. Bag Equivalence via a Proof-Relevant Membership Relation. In *Interactive Theorem Proving* (Lecture Notes in Computer Science). Lennart Beringer and Amy Felty, (Eds.) Springer, Berlin, Heidelberg, 149–165. ISBN: 978-3-642-32347-8. doi:10.1007/978-3-642-32347-8_11.

[6] Marcelo Fiore and Philip Saville. 2017. List Objects with Algebraic Structure. *LIPIcs, Volume 84, FSCD 2017*, 84, 16:1–16:18. Dale Miller, (Ed.) ISBN: 9783959770477. doi:10.4230/LIPICS.FSCD.2017.16.

[7] Tatsuya Hagino. 1987. *A categorical programming language*. Ph.D. Dissertation. University of Edinburgh, UK. https://hdl.handle.net/1842/13976.

[8] C.B. Jay. 1997. Finite objects in a locos. *Journal of Pure and Applied Algebra*, 116, 1–3, (Mar. 1997), 169–183. doi:10.1016/S0022-4049(97)81630-1.

[9] G. M. Kelly. 1980. 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, (Aug. 1980), 1–83. doi:10.1017/S0004972700006353.

[10] F. William Lawvere. 1963. FUNCTORIAL SEMANTICS OF ALGEBRAIC THEORIES. *Proceedings of the National Academy of Sciences*, 50, 5, (Nov. 1963), 869–872. doi:10.1073/pnas.50.5.869.

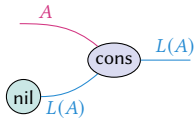
[11] Maria Emilia Maietti. 2010. Joyal's arithmetic universe as list-arithmetic pretopos. *Theory and Applications of Categories*, 24, 39–83. doi:10.70930/tac/qee78enb.

[12] Paul-André Melliès, Nicolas Tabareau, and Christine Tasson. 2018. An explicit formula for the free exponential modality of linear logic. *Mathematical Structures in Computer Science*, 28, 7, (Aug. 2018), 1253–1286. doi:10.1017/S0960129516000426.

A Proofs for Section 2 (List objects)

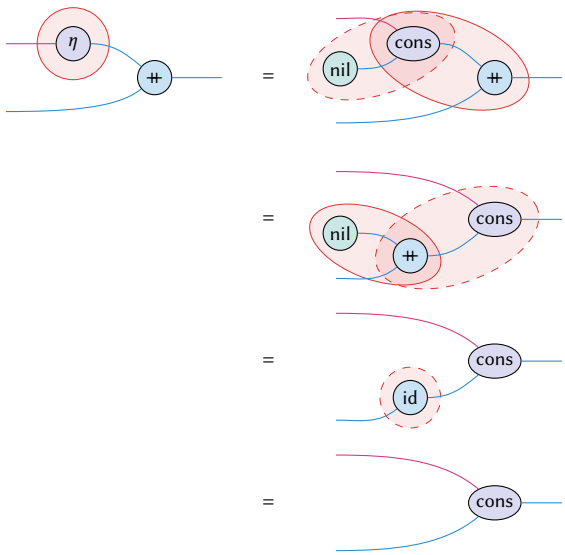
When rewriting a diagram, the affected portions will be outlined, and the results on the following diagram will be outlined with a dashed border. String labels will also be omitted from diagrams in proofs to reduce clutter, with colours indicating the types.

We define the append operation $+\!\!+ : L(A) \otimes L(A) \to L(A)$ as the iterator $\mathsf{it}(\mathsf{id}, \mathsf{cons})$, and the unit map $\eta : A \to L(A)$ as:



Lemma 10.

Proof.

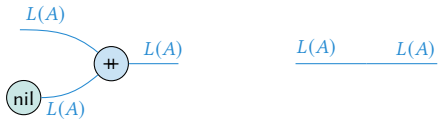


□

Theorem 2. *The list object $L(A)$ is the free monoid on A.*

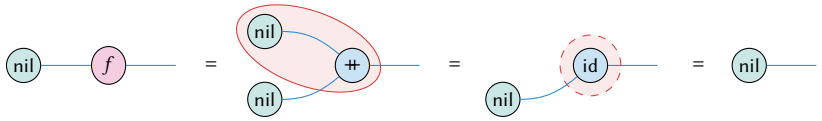
Proof. We claim that $(L(A), \mathsf{nil}, +\!\!+)$ is a free monoid structure on A.
The left unit law directly holds by $+\!\!+$-nil-β:

For the right unit law, consider the following maps $f, g : L(A) \to L(A)$ below:

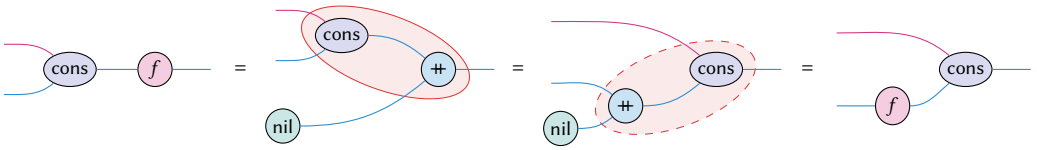


We claim that these maps are both the iterator $\mathsf{it}(\mathsf{nil}, \mathsf{cons})$. By the uniqueness of the mediating map, it suffices to show that the two maps satisfy the required computation rules.

The identity map trivially satisfies both rules, so we have $g = \mathsf{it}(\mathsf{nil}, \mathsf{cons})$. We have f-nil-β by applying $+\!\!+$-nil-β:



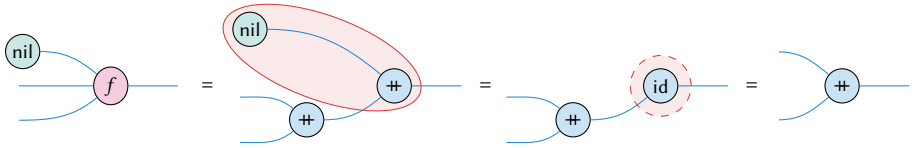
and similarly, f-cons-β follows from $+\!\!+$-cons-β:



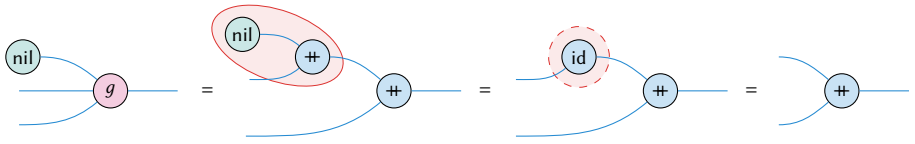
and hence $f = \mathsf{it}(\mathsf{nil}, \mathsf{cons}) = g$.

For associativity, let $f, g : L(A) \otimes L(A) \otimes L(A) \to L(A)$ be the two maps below:

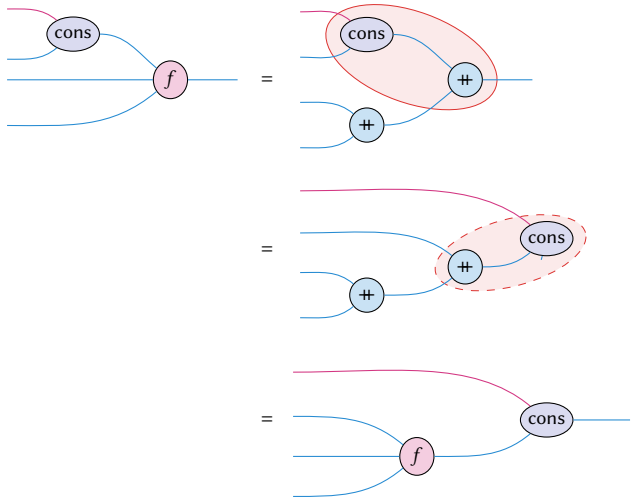
We claim that both maps are equal to $\mathsf{it}(+\!\!+, \mathsf{cons})$. We have nil-$\beta$ for both since $+\!\!+$ computes on nil in the left argument. In detail, for f:



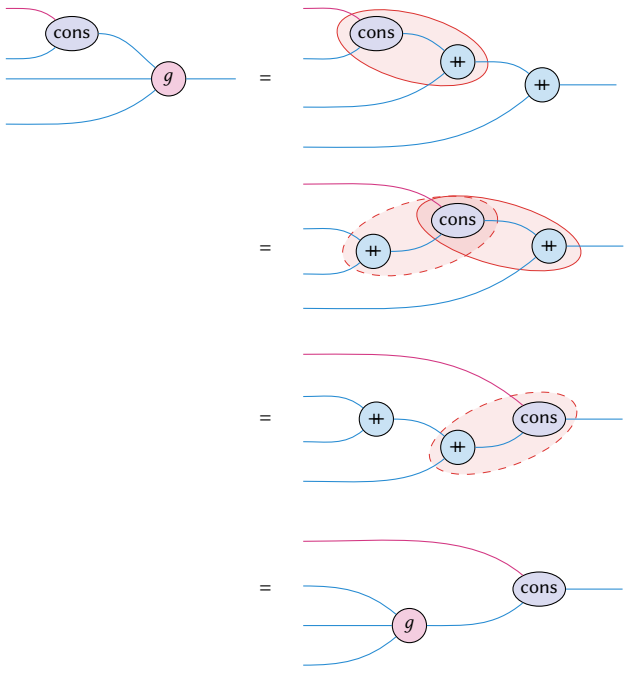
and for g:



Similarly, we have cons-β since $+\!\!+$ also computes on cons. For f:



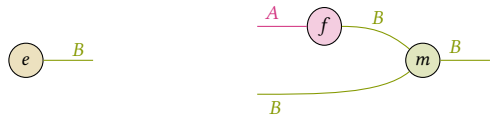
and for g, we push the cons through $+\!\!+$ with two computations:



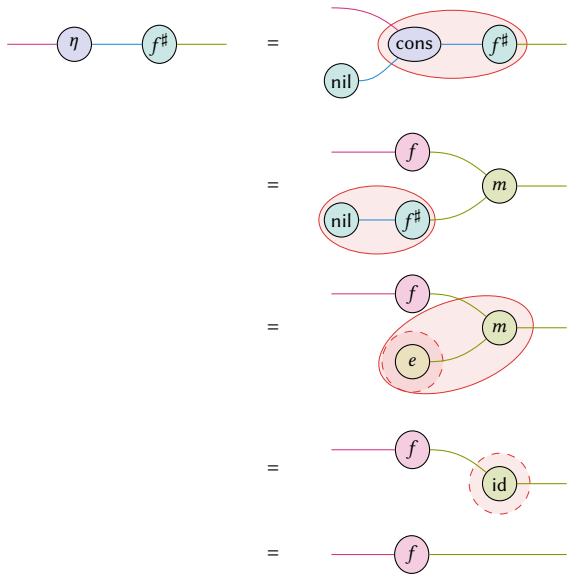
Hence $f = \mathsf{it}(+\!\!+, \mathsf{cons}) = g$.

It remains to show the unique extension property. Let $f : A \to B$ be a morphism, with (B, e, m) a monoid object. We define the extension $f^\sharp : L(A) \to B$ as the mediating map $\mathsf{it}(n, c)$, where

$n : 1 \longrightarrow B$ and $c : A \otimes B \longrightarrow B$ are:

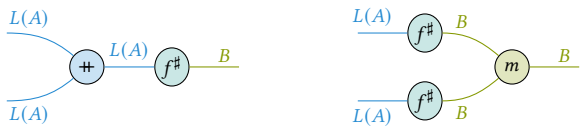


We verify that $f^\sharp \circ \eta = f$:

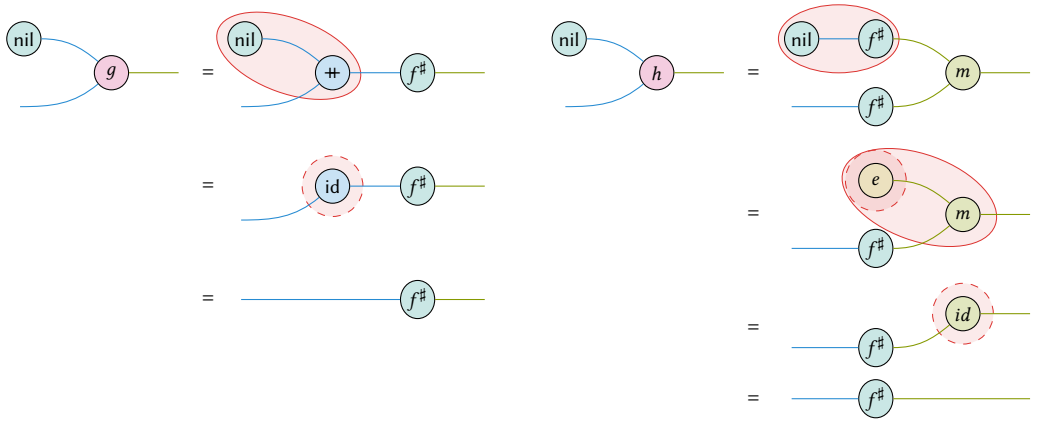


Next, we check that $f^\sharp$ is a monoid homomorphism.

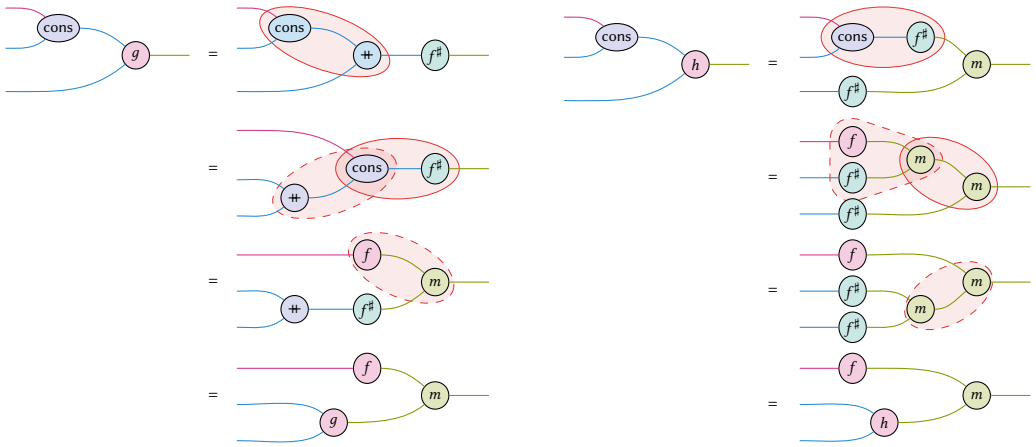
Since $f^\sharp = \mathsf{it}(e, \ldots)$, $f^\sharp$ computes on nil to e and hence preserves the monoid identity. For the homomorphism property, let $g, h : L(A) \otimes L(A) \longrightarrow B$ be the following maps:



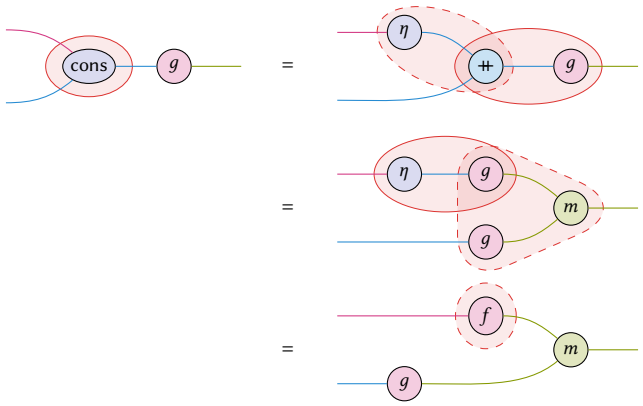
We show that these maps are both the iterator $\mathrm{it}(f^\sharp, c)$, where c is the same list algebra map as in the definition of $f^\sharp$ above. We have g-nil-β by $+\!\!+$-nil-β, and h-nil-β by $f^\sharp$-nil-β and left unit for B:



For cons-β we have the computations:

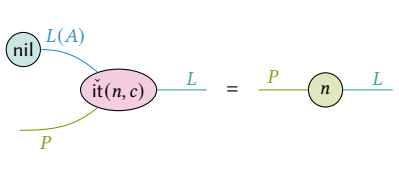


Finally, we show that the extension $f^\sharp$ is unique. Suppose there is another monoid homomorphism $g : L(A) \to B$ satisfying $g \circ \eta = f$. Then, g satisfies the same computation rules as $f^\sharp$. Indeed, nil-β holds since g preserves identities, and cons-β holds via the following computation:

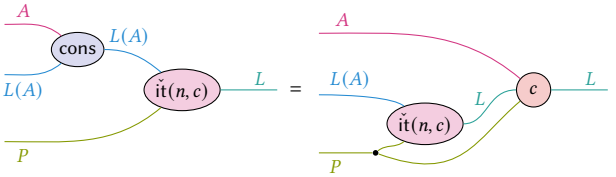


□

Theorem 4. *If $(C, \times, 1)$ is cartesian monoidal and $L(A)$ is a list object, then $L(A)$ satisfies the following stronger universal property: for every object L equipped with maps $n : P \to L$ and $c : A \times L \times P \to L$, then there exists a doubly parametrised iterator $\check{\mathrm{it}}(n, c) : L(A) \times P \to L$, such that the following two computation rules hold:*



(a) nil-β



(a) cons-β

Proof sketch. Consider the list algebra structure on $K = L \times P$ given by $\check{n} : P \xrightarrow{\Delta_P} P \times P \xrightarrow{n \times P} L \times P$ and $\check{c} : A \times L \times P \xrightarrow{A \times L \times \Delta_P} A \times L \times P \times P \xrightarrow{c \times P} L \times P$. By the ordinary universal property, we have an iterator $\mathrm{it}(\check{n}, \check{c}) : L(A) \times P \to L \times P$. Then, $\check{\mathrm{it}}(n, c)$ is given by $\pi_1 \circ \mathrm{it}(\check{n}, \check{c})$. □

Theorem 5 (Lemma 3.4.3 [6]). *If $(C, \times, 1)$ is cartesian monoidal, then the induced monad is strong.*

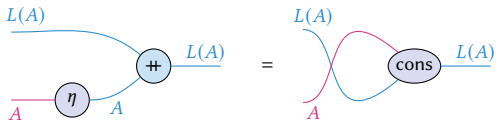
Proof sketch. Since the category is symmetric, we only need to define one of the strengths. A construction of the right strength $\tau_{A,B} : L(A) \times B \to L(A \times B)$ is as follows.

We can equip $L(A \times B)$ with a list algebra structure with the maps $n := B \xrightarrow{!} 1 \xrightarrow{\mathrm{nil}} L(A \times B)$ and $c := A \times L(A \times B) \xrightarrow{\cong} A \times B \times L(A \times B) \xrightarrow{\mathrm{cons}} L(A \times B)$, and from the strong universal property, we obtain a map $\check{\mathrm{it}}(n, c) : L(A) \times B \to L(A \times B)$. This iterator is the right strength map.

The required strength coherences then follow from the strong universal property. □

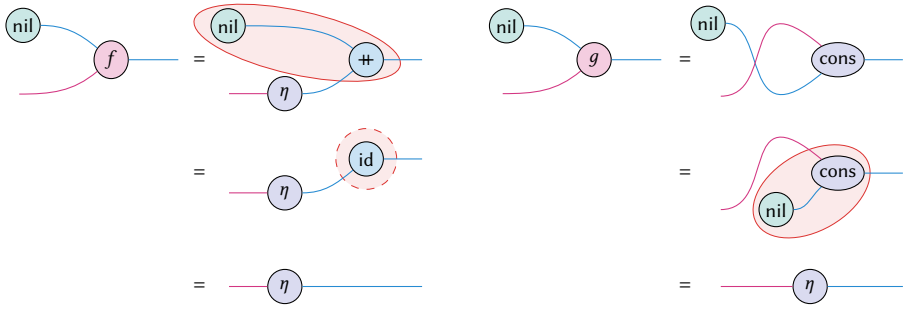
B Proofs for Section 3 (Symmetric List Objects)

Lemma 11. *For any symmetric list object $L(A)$:*

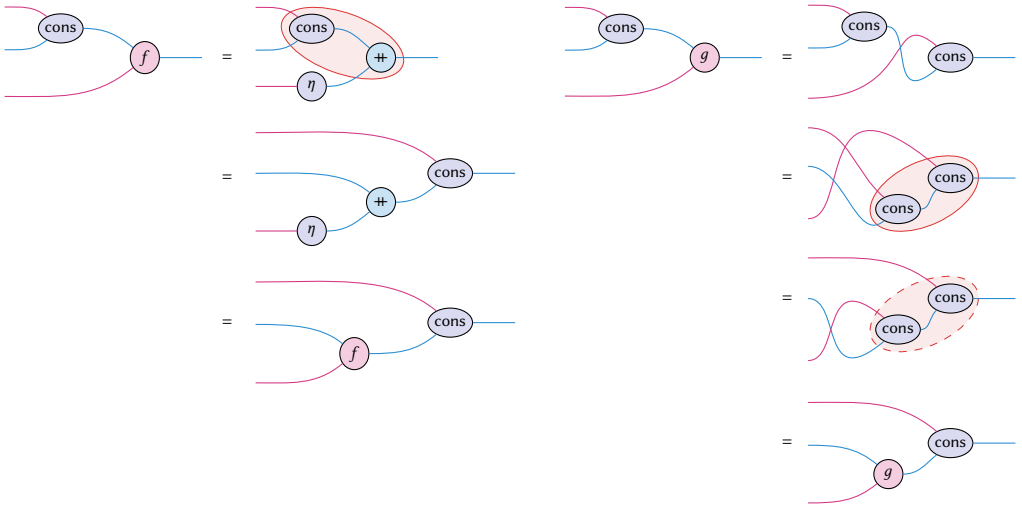


Proof. Let the two maps be f and g, respectively. We claim that both maps are the iterator $\mathrm{it}(\eta, \mathrm{cons})$.

nil-β:



cons-β:



□

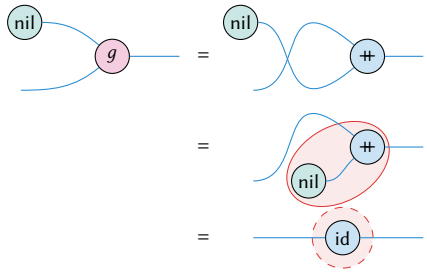
Theorem 7. *Let* $(L(A), \mathsf{nil}, \mathsf{cons})$ *be a list object on an object* A*. Then, the following are equivalent:*

(1) $(L(A), \mathsf{nil}, \mathsf{cons})$ *is a symmetric list object;*
(2) $(L(A), \mathsf{nil}, +\!\!+)$ *is a commutative monoid object;*
(3) $(L(A), \mathsf{nil}, +\!\!+, \eta)$ *is the free commutative monoid object on* A*;*

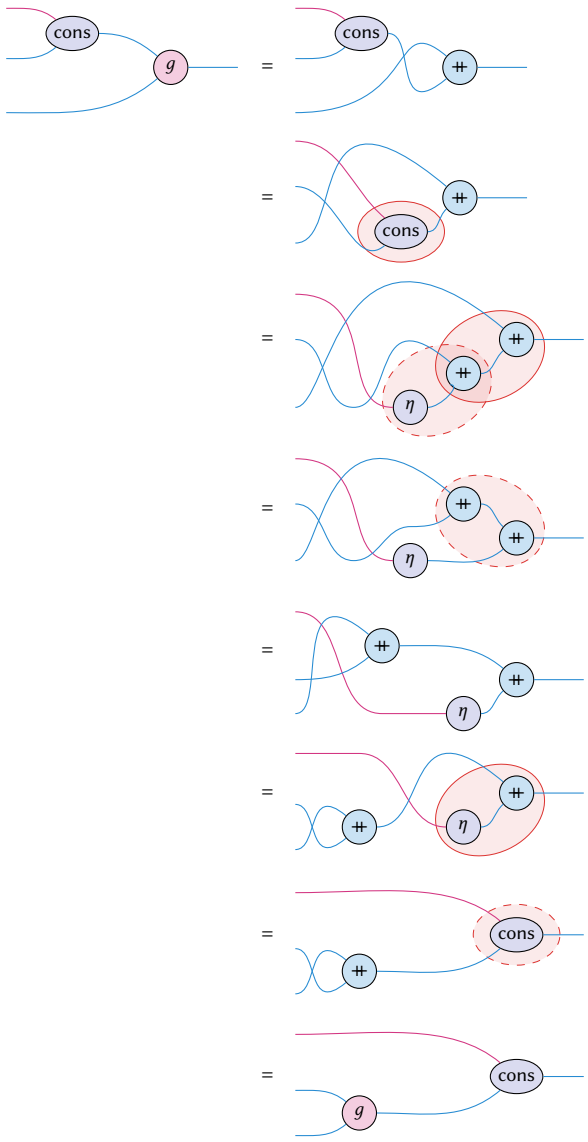
Proof. $(1 \Rightarrow 2)$ We have already shown that list objects are monoids, so we need only show that $+\!\!+$ is commutative:

Let the two maps above be f and g respectively. By definition, $f = +\!\!+ = \mathsf{it}(\mathsf{id}, \mathsf{cons})$, so we show that the braided version satisfies the same computation rules:

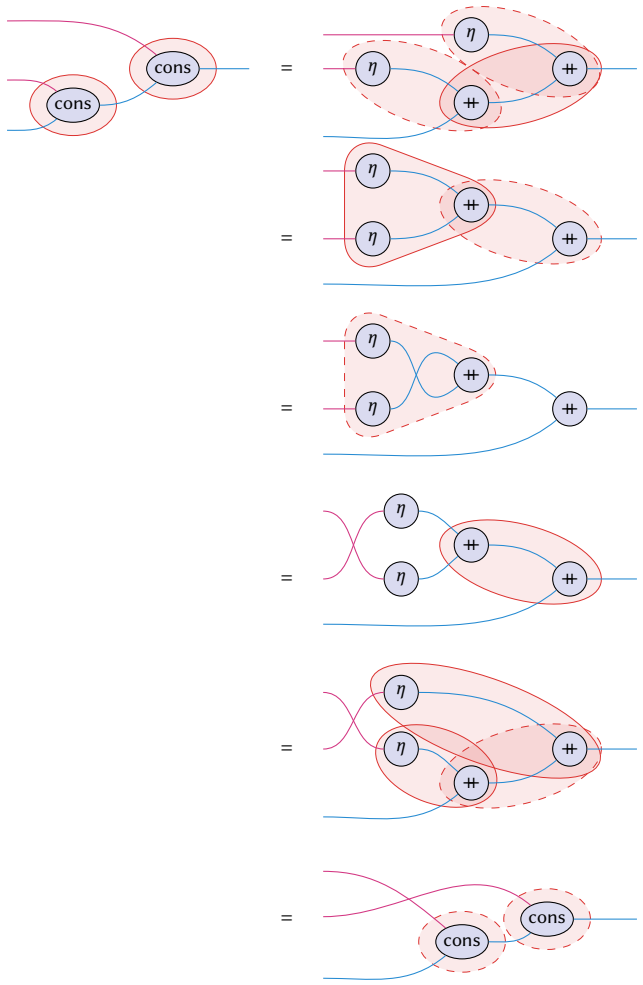
g-nil-β:



g-cons-β:



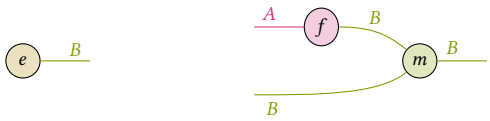
$(2 \Rightarrow 1)$ $L(A)$ is a list object by assumption, so it suffices to show the ilst axiom only:



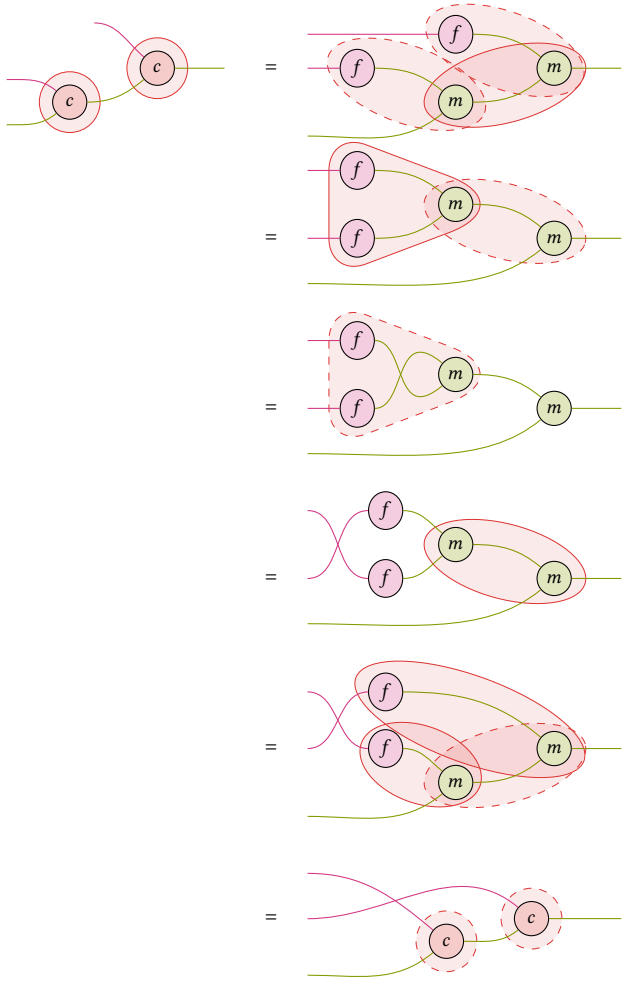
This establishes the equivalence of (1) and (2).

$(3 \Rightarrow 2)$ is clear, so it remains to prove the reverse implication.

$(1 \wedge 2 \Rightarrow 3)$ $(L(A), \mathsf{nil}, +\!\!+)$ is a commutative monoid object by assumption, so we need only show freeness. Let $f : A \to B$ be a morphism, with (B, e, m) a commutative monoid object. Since monoid homomorphisms are automatically commutative monoid homomorphisms, we may construct the extension $f^\sharp$ as $\mathrm{it}(n, c)$, where n and c are:



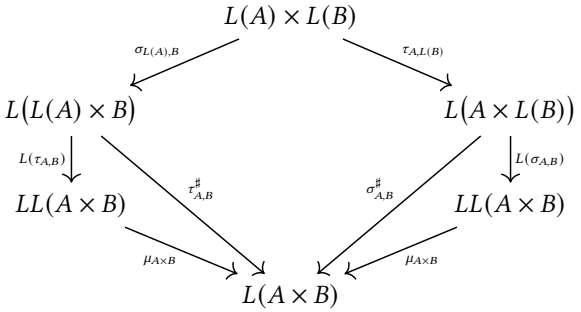
as in the non-commutative case, but we need to show that the list algebra structure on B satisfies the ilst axiom to use the iterator:


c
c
=
f
f
m
m
=
f
f
m
m
=
f
f
m
m
=
f
f
m
m
=
f
f
m
m
=
c
c

□

Theorem 9. *If $(C, \times, 1)$ is cartesian monoidal, then the induced monad $T : C \to C$ is strong, and further, commutative.*

Proof sketch. The construction of the strength map(s) is the same as in the non-symmetric case, but in this case it satisfies the following additional commutativity coherence:



The lower pairs of edges compute to τ and σ when precomposed with $\eta_{L(A)\times B}$ and $\eta_{A\times L(B)}$, respectively, and are thus extensions of the strength maps by the universal property of $(-)^{\sharp}$. The two sides of the remaining square can then be shown to be given by the same doubly parametrised iterator by verifying the computation rules. □