Symmetric List Objects

Rin Liu

joint work with
Fiona Blackett & Vikraman Choudhury

MSP, University of Strathclyde, Glasgow, UK

MSFP @ FLoC 2026, Lisbon, Portugal

July 18, 2026

Lists

In scheme:

```
(a b c) ; a list of three elements
(a . (b . (c . ()))) ; the same list
() ; the empty list
(car '(a b c)) ; a
(cdr '(a b c)) ; (b c)
```

In Haskell:

```
data [a] = [] | a : [a]
```

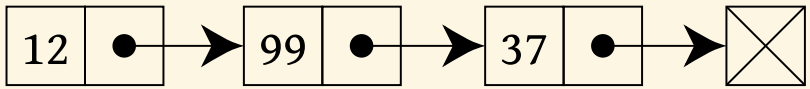
In Agda:

```
data ℒist (A : Set) : Set where
  [] : ℒist A
  _::_ : A → ℒist A → ℒist A
```

In Rocq:

```
Inductive list (A : Type) : Type :=
  | nil : list A
  | cons : A -> list A -> list A.
```

Linked lists:



Constructions

In categories with finite products and countable coproducts, we can construct lists as:

$$\coprod_{n \geq 0} A^n$$

Constructions

In categories with finite products and countable coproducts, we can construct lists as:

$$\coprod_{n \geq 0} A^n$$

Alternatively, if we have a natural numbers object, sigma types, and functions out of Fin_n, we can construct lists as:

$$\sum_{n:\mathbb{N}} A^{\mathsf{Fin}_n}$$

Constructions

In categories with finite products and countable coproducts, we can construct lists as:

$$\coprod_{n \geq 0} A^n$$

Alternatively, if we have a natural numbers object, sigma types, and functions out of Fin_n, we can construct lists as:

$$\sum_{n:\mathbb{N}} A^{\mathsf{Fin}_n}$$

We can also construct lists as initial algebras of the following functor:

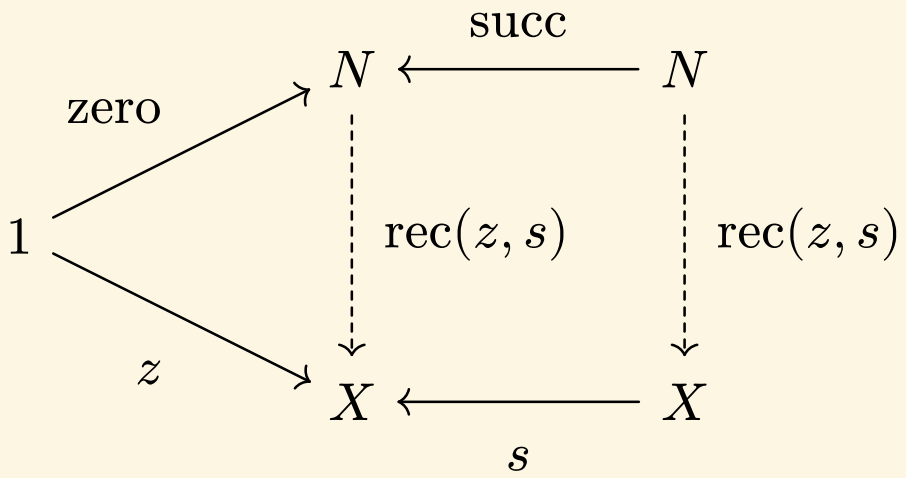
$$F(X) = 1 + A \times X$$

but this requires an initial object, and for the colimit of the following ω-chain to exist:

$$0 \to F(0) \to FF(0) \to FFF(0) \to \ldots$$

List Objects

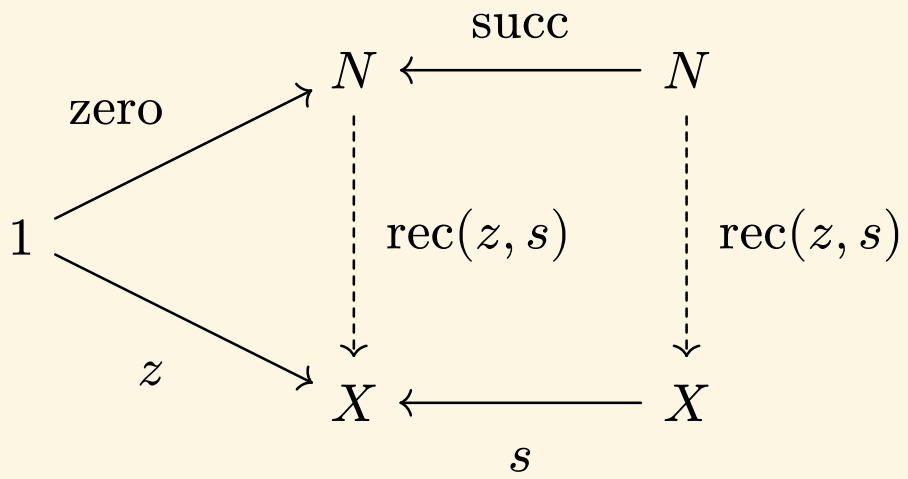
Lawvere introduced[1] natural numbers objects in his axiomatisation of set theory (ETCS):



[1] F. W. Lawvere, "An Elementary Theory of the Category of Sets*," *Proceedings of the National Academy of Sciences*, vol. 52, no. 6, pp. 1506–1511, Dec. 1964, doi: 10.1073/pnas.52.6.1506.

List Objects

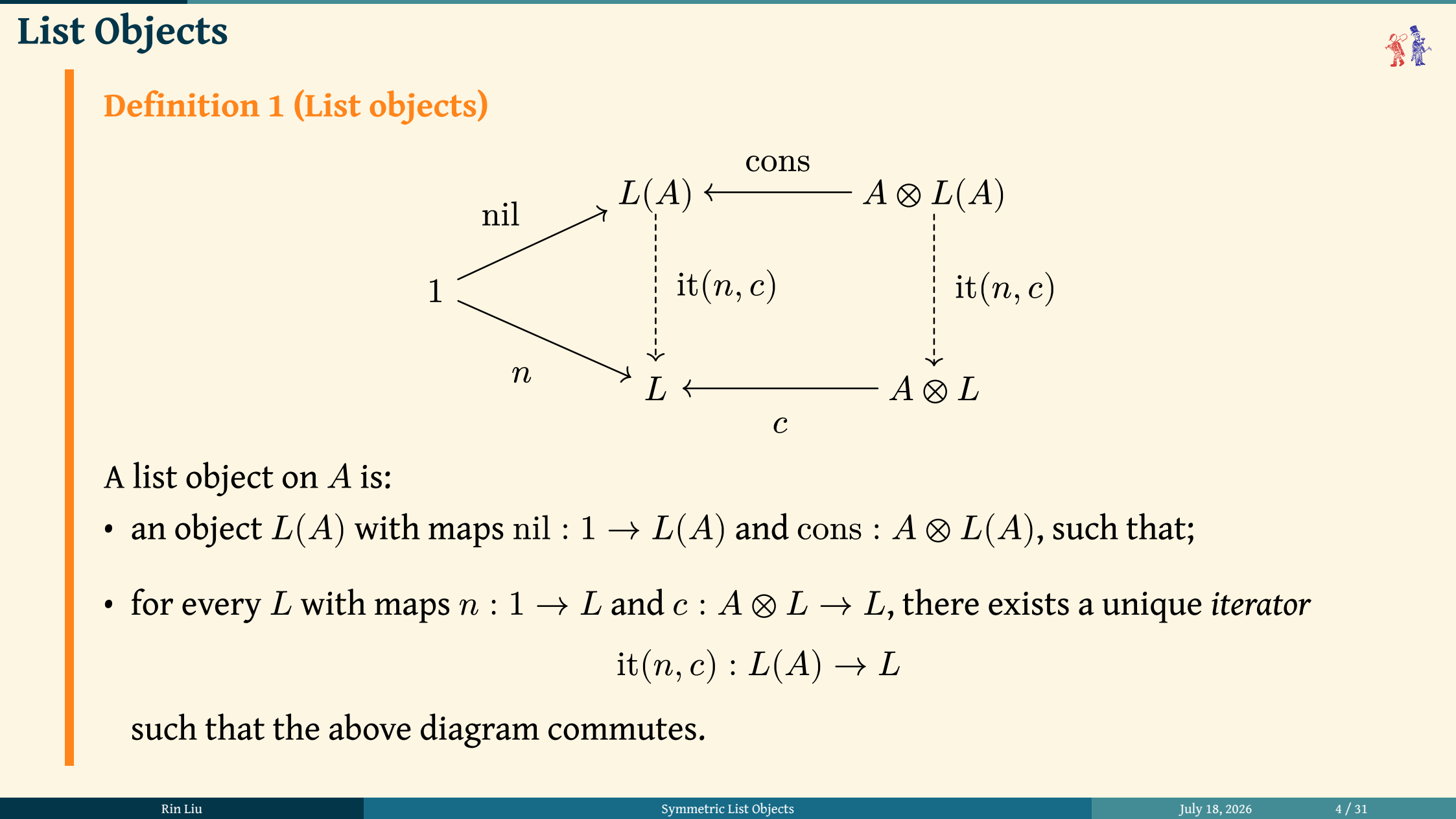
Lawvere introduced[1] natural numbers objects in his axiomatisation of set theory (ETCS):



We can adapt this universal property to axiomatise list objects in general monoidal categories.

[1] F. W. Lawvere, "An Elementary Theory of the Category of Sets*," *Proceedings of the National Academy of Sciences*, vol. 52, no. 6, pp. 1506–1511, Dec. 1964, doi: 10.1073/pnas.52.6.1506.

List Objects



Definition 1 (List objects)

A list object on A is:

- an object $L(A)$ with maps $\mathrm{nil} : 1 \to L(A)$ and $\mathrm{cons} : A \otimes L(A)$, such that;
- for every L with maps $n : 1 \to L$ and $c : A \otimes L \to L$, there exists a unique *iterator*

$$\mathrm{it}(n, c) : L(A) \to L$$

such that the above diagram commutes.

Previous work

- Joyal introduced arithmetic universes with list objects
- Maietti[1] [2] studies them in type theory
- Paré and Roman[3] study monoidal categories with parametrized NNOs
- Cockett[4] and Jay[5] study pretoposes with list objects, called locoi
- Fiore and Saville[6] study parametrised list objects

---

[1] M. E. Maietti, "Joyal's Arithmetic Universes via Type Theory," *Electronic Notes in Theoretical Computer Science*, vol. 69, pp. 272–286, Feb. 2003, doi: 10.1016/S1571-0661(04)80569-3.

[2] M. E. Maietti, "Joyal's Arithmetic Universe as List-Arithmetic Pretopos," *Theory and Applications of Categories*, vol. 24, pp. 39–83, 2010, doi: 10.70930/tac/qee78enb.

[3] R. Paré and L. Román, "Monoidal Categories with Natural Numbers Object," *Studia Logica*, vol. 48, no. 3, pp. 361–376, Sep. 1989, doi: 10.1007/BF00370829.

[4] J. Cockett, "List-Arithmetic Distributive Categories: Locoi," *Journal of Pure and Applied Algebra*, vol. 66, no. 1, pp. 1–29, Oct. 1990, doi: 10.1016/0022-4049(90)90121-W.

[5] C. Jay, "Finite Objects in a Locos," *Journal of Pure and Applied Algebra*, vol. 116, no. 1–3, pp. 169–183, Mar. 1997, doi: 10.1016/S0022-4049(97)81630-1.

[6] M. Fiore and P. Saville, "List Objects with Algebraic Structure," *LIPIcs, Volume 84, FSCD 2017*, vol. 84, p. 16:1–16:18, 2017, doi: 10.4230/LIPICS.FSCD.2017.16.

Examples of list objects in nature

- In $\mathsf{Set}(1, \times)$ (or any Grothendieck topos over Set), list objects are cons-lists:

$$1 \xrightarrow{\text{nil}} \mathcal{L}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{L}_f(A)$$

Examples of list objects in nature

- In $\mathsf{Set}(1, \times)$ (or any Grothendieck topos over Set), list objects are cons-lists:

$$1 \xrightarrow{\text{nil}} \mathcal{L}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{L}_f(A)$$

Examples of list objects in nature

- In $\mathsf{Set}(1, \times)$ (or any Grothendieck topos over Set), list objects are cons-lists:

$$1 \xrightarrow{\text{nil}} \mathcal{L}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{L}_f(A)$$

- In $[\mathcal{C}, \mathcal{C}]_{\text{fin}}(\text{Id}, \circ)$ (finitary endofunctors), list objects are the free monad construction:

$$\text{Id} \xrightarrow{\text{nil}} F^*(F) \xleftarrow{\text{cons}} F \circ F^*(F)$$

Examples of list objects in nature

- In $\mathsf{Set}(1, \times)$ (or any Grothendieck topos over Set), list objects are cons-lists:

$$1 \xrightarrow{\text{nil}} \mathcal{L}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{L}_f(A)$$

- In $[\mathcal{C}, \mathcal{C}]_{\text{fin}}(\text{Id}, \circ)$ (finitary endofunctors), list objects are the free monad construction:

$$\text{Id} \xrightarrow{\text{nil}} F^*(F) \xleftarrow{\text{cons}} F \circ F^*(F)$$

- In R-$\mathsf{Mod}(R, \otimes_R)$ (modules over a commutative ring R), list objects are tensor algebras:

$$R \xrightarrow{\text{nil}} \bigoplus_n V^{\otimes^n} \xleftarrow{\text{cons}} V \otimes_R \bigoplus_n V^{\otimes^n}$$

Examples of list objects in nature

- In $\mathsf{Set}(1, \times)$ (or any Grothendieck topos over Set), list objects are cons-lists:

$$1 \xrightarrow{\text{nil}} \mathcal{L}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{L}_f(A)$$

- In $[\mathcal{C}, \mathcal{C}]_{\text{fin}}(\text{Id}, \circ)$ (finitary endofunctors), list objects are the free monad construction:

$$\text{Id} \xrightarrow{\text{nil}} F^*(F) \xleftarrow{\text{cons}} F \circ F^*(F)$$

- In $R\text{-}\mathsf{Mod}(R, \otimes_R)$ (modules over a commutative ring R), list objects are tensor algebras:

$$R \xrightarrow{\text{nil}} \bigoplus_n V^{\otimes^n} \xleftarrow{\text{cons}} V \otimes_R \bigoplus_n V^{\otimes^n}$$

- In $\mathsf{Top}_\bullet(S^0, \wedge)$ (pointed spaces under smash product), list objects on connected spaces correspond to the James construction:

$$1 \xrightarrow{\text{nil}} J(A) \xleftarrow{\text{cons}} A \wedge J(A)$$

Limitations

1. We can only eliminate out of lone list objects.

Limitations

1. We can only eliminate out of lone list objects.
   - To define the list append function

$$+\!+ : L(A) \otimes L(A) \to L(A)$$

   we would need to curry the second argument to isolate the $L(A)$ on the left:

$$+\!+ : L(A) \to [L(A), L(A)]$$

   but this requires internal homs.

Limitations

1. We can only eliminate out of lone list objects.
   - To define the list append function

     $$+\!\!+ : L(A) \otimes L(A) \to L(A)$$

     we would need to curry the second argument to isolate the $L(A)$ on the left:

     $$+\!\!+ : L(A) \to [L(A), L(A)]$$

     but this requires internal homs.

2. These list objects also do not necessarily coincide with free monoids.

Limitations

1. We can only eliminate out of lone list objects.
   - To define the list append function

     $$+\!\!+ : L(A) \otimes L(A) \to L(A)$$

     we would need to curry the second argument to isolate the $L(A)$ on the left:

     $$+\!\!+ : L(A) \to [L(A), L(A)]$$

     but this requires internal homs.

2. These list objects also do not necessarily coincide with free monoids.
   - For example, every object of $\mathrm{Set}(0, +)$ is the free monoid on itself, but list objects on inhabited sets are always infinite:

     $$\mu X.0 + (A + X) \quad \cong \quad \mu X.A + X$$

Parametrised List Objects

To solve the first problem, we strengthen the universal property as follows:

Definition 2 (Parametrised list objects)

$$1 \xrightarrow{\text{nil}} L(A) \xleftarrow{\text{cons}} A \otimes L(A)$$

Parametrised List Objects

To solve the first problem, we strengthen the universal property as follows:

Definition 2 (Parametrised list objects)

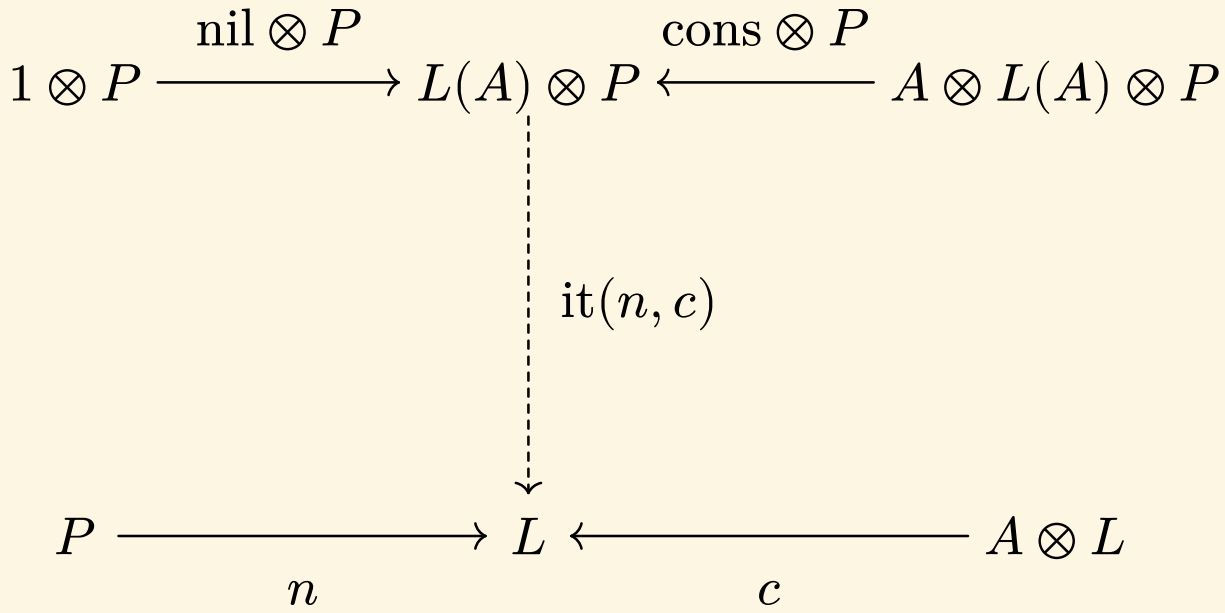
$$1 \xrightarrow{\text{nil}} L(A) \xleftarrow{\text{cons}} A \otimes L(A)$$

$$P \xrightarrow[n]{} L \xleftarrow[c]{} A \otimes L$$

Parametrised List Objects

To solve the first problem, we strengthen the universal property as follows:

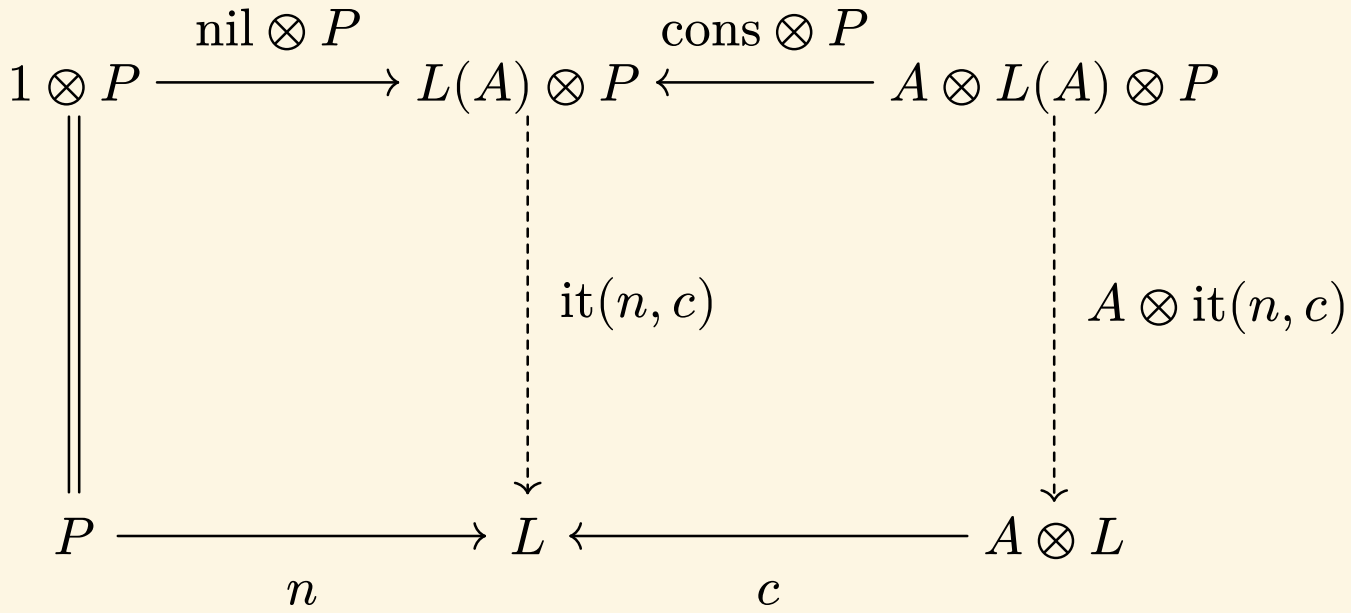
Definition 2 (Parametrised list objects)



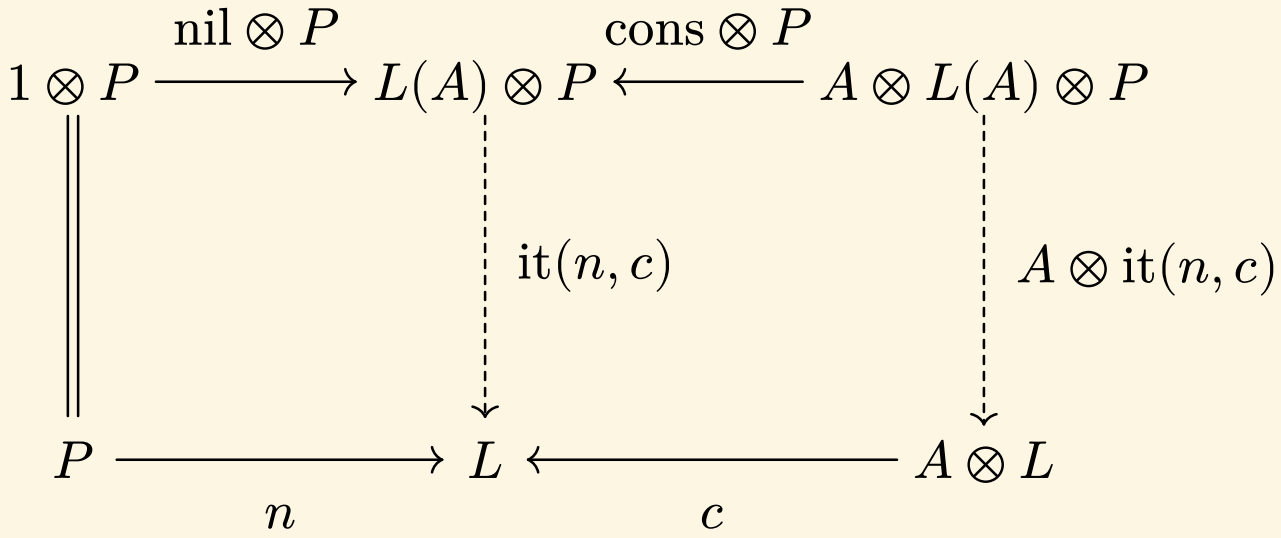
Parametrised List Objects

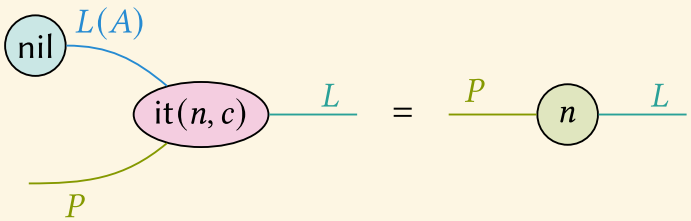
To solve the first problem, we strengthen the universal property as follows:

Definition 2 (Parametrised list objects)

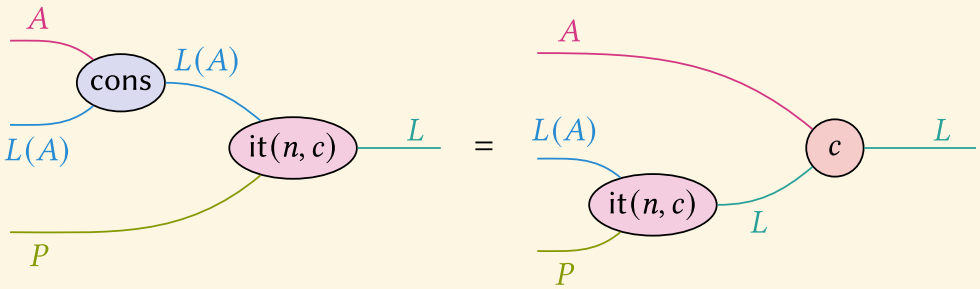


String Diagrams





(a) nil-β



(b) cons-β

Using the iterator

We can now use this stronger universal property to define $+\!\!+$ as an iterator.

Using the iterator

We can now use this stronger universal property to define $+\!+$ as an iterator.

$$\mathrm{it}(n, c) : L(A) \otimes P \to L$$

- Since $+\!+ : L(A) \otimes L(A) \to L(A)$, we have $L(A)$ as both the parameter P and the target L.
- We now only need maps $n : L(A) \to L(A)$ and $c : A \otimes L(A) \to L(A)$.

Using the iterator

We can now use this stronger universal property to define $+\!\!+$ as an iterator.

$$\mathrm{it}(n, c) : L(A) \otimes P \to L$$

- Since $+\!\!+ : L(A) \otimes L(A) \to L(A)$, we have $L(A)$ as both the parameter P and the target L.
- We now only need maps $n : L(A) \to L(A)$ and $c : A \otimes L(A) \to L(A)$.

```
_++_ : List A → List A → List A
[] ++ ys = ys
(x :: xs) ++ ys = x :: (xs ++ ys)
```

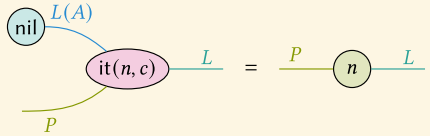
Using the iterator

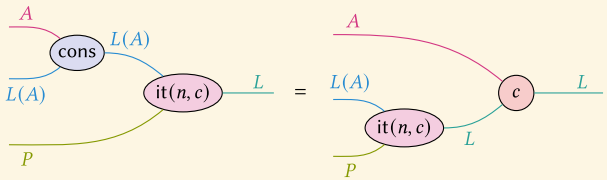
We can now use this stronger universal property to define $+\!\!+$ as an iterator.

$$\mathrm{it}(n, c) : L(A) \otimes P \to L$$

- Since $+\!\!+ : L(A) \otimes L(A) \to L(A)$, we have $L(A)$ as both the parameter P and the target L.
- We now only need maps $n : L(A) \to L(A)$ and $c : A \otimes L(A) \to L(A)$.

```
_++_ : List A → List A → List A
[] ++ ys = ys
(x :: xs) ++ ys = x :: (xs ++ ys)
```





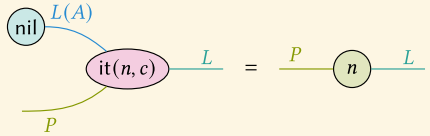
Using the iterator

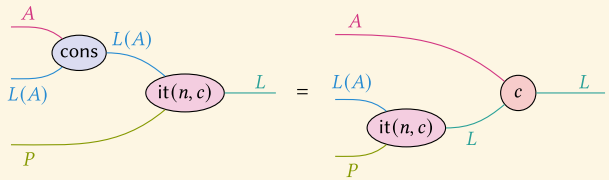
We can now use this stronger universal property to define $+\!+$ as an iterator.

$$\mathrm{it}(n, c) : L(A) \otimes P \to L$$

- Since $+\!+ : L(A) \otimes L(A) \to L(A)$, we have $L(A)$ as both the parameter P and the target L.
- We now only need maps $n : L(A) \to L(A)$ and $c : A \otimes L(A) \to L(A)$.

```
_++_ : List A → List A → List A
[] ++ ys = ys
(x :: xs) ++ ys = x :: (xs ++ ys)
```





so we pick $n = \mathrm{id}$ and $c = \mathrm{cons}$:

$$+\!+ := \mathrm{it}(\mathrm{id}, \mathrm{cons}).$$

Monoids

The stronger universal property resolves the first limitation, allowing us more flexibility when eliminating from lists.

Monoids

The stronger universal property resolves the first limitation, allowing us more flexibility when eliminating from lists. For the second limitation:

Theorem 1

$(L(A), \mathrm{nil}, +\!\!+)$ is the free monoid on A.

Monoids

The stronger universal property resolves the first limitation, allowing us more flexibility when eliminating from lists. For the second limitation:

Theorem 1

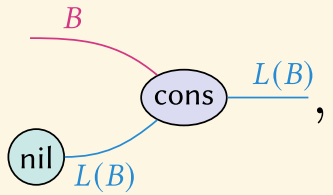
$(L(A), \mathrm{nil}, +\!\!+)$ is the free monoid on A.

Lemma 2

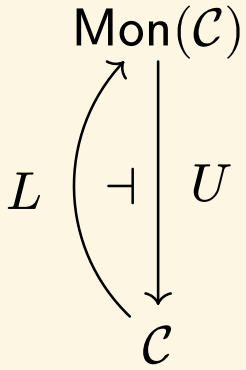
If $\mathcal{C}$ admits parametrised list objects for every object, then $L(-)$ is a functor.

$$f : X \to Y \quad \mapsto \quad \left(X \xrightarrow{f} Y \xrightarrow{\eta_Y} L(Y)\right)^\sharp : L(X) \to L(Y)$$

$$\eta := \quad , \qquad \left(A \xrightarrow{g} (B, e, m)\right)^\sharp := \mathrm{it}(e, m \circ (g \otimes B))$$



Monoids



Theorem 3 (M. Fiore and P. Saville[1])

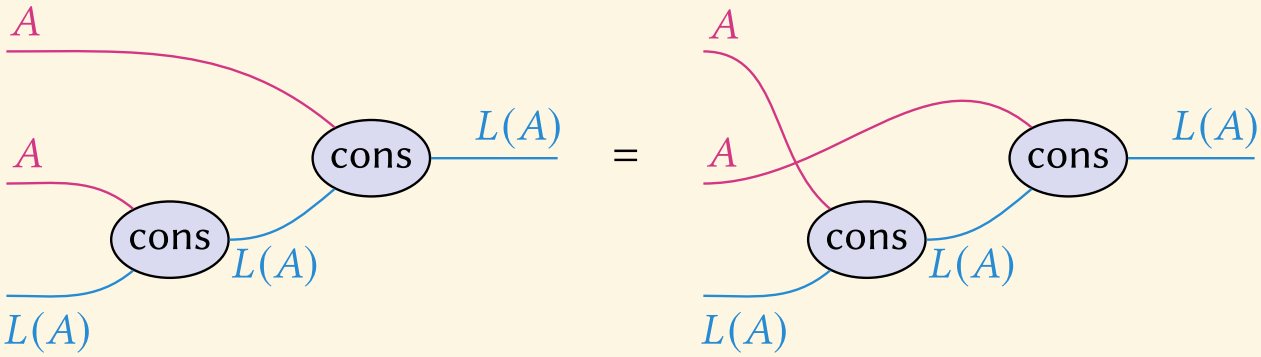
If $(\mathcal{C}, \times, 1)$ is cartesian monoidal, then the induced monad is strong.

[1] M. Fiore and P. Saville, "List Objects with Algebraic Structure," *LIPIcs, Volume 84, FSCD 2017*, vol. 84, p. 16:1–16:18, 2017, doi: 10.4230/LIPICS.FSCD.2017.16.

Symmetric Lists

Hereafter, the ambient category is symmetric strict monoidal.

Definition 3 (Symmetric list objects)



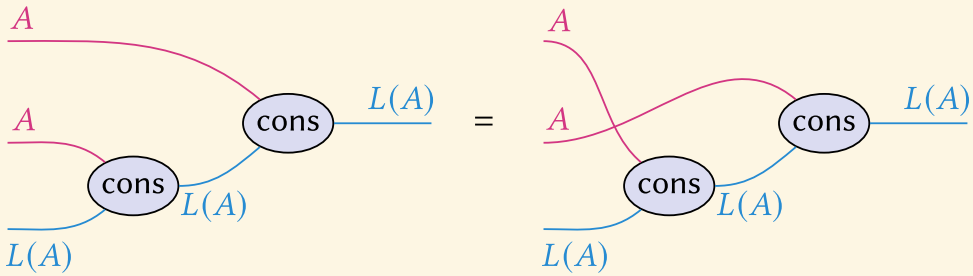
ilst axiom

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

Symmetric Lists

Hereafter, the ambient category is symmetric strict monoidal.

Definition 3 (Symmetric list objects)



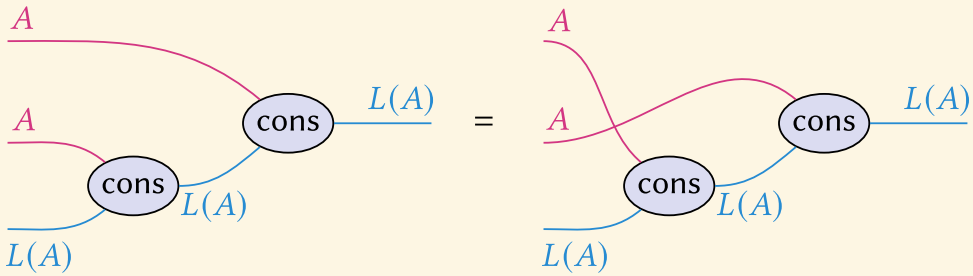

```
data ℐlst (A : Set) : Set where
  [] : ℐlst A
  _::_ : A → ℐlst A → ℐlst A
  swap : (a b : A) (xs : ℐlst A)
     → a :: b :: xs ≡ b :: a :: xs
```

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

Symmetric Lists

Hereafter, the ambient category is symmetric strict monoidal.

Definition 3 (Symmetric list objects)




```
data ℐlst (A : Set) : Set where
  [] : ℐlst A
  _::_ : A → ℐlst A → ℐlst A
  swap : (a b : A) (xs : ℐlst A)
     → a :: b :: xs ≡ b :: a :: xs
```

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

$$\sum_{n \geq 0} A^n / S_n$$

Examples of symmetric list objects in nature

- In $\mathrm{Set}(1, \times)$ (or any Grothendieck topos over Set), symm. list objects are finite multisets:

$$1 \xrightarrow{\text{nil}} \mathcal{M}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{M}_f(A)$$

Examples of symmetric list objects in nature

- In $\mathsf{Set}(1, \times)$ (or any Grothendieck topos over Set), symm. list objects are finite multisets:

$$1 \xrightarrow{\text{nil}} \mathcal{M}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{M}_f(A)$$

- In $\mathsf{R\text{-}Mod}(R, \otimes_R)$, symmetric list objects are symmetric algebras:

$$1 \xrightarrow{\text{nil}} \bigoplus_n V^{\otimes^n} \xleftarrow{\text{cons}} V \otimes \bigoplus_n V^{\otimes^n}$$

Examples of symmetric list objects in nature

- In $\mathsf{Set}(1, \times)$ (or any Grothendieck topos over Set), symm. list objects are finite multisets:

$$1 \xrightarrow{\text{nil}} \mathcal{M}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{M}_f(A)$$

- In $\mathsf{R\text{-}Mod}(R, \otimes_R)$, symmetric list objects are symmetric algebras:

$$1 \xrightarrow{\text{nil}} \bigoplus_n V^{\otimes^n} \xleftarrow{\text{cons}} V \otimes \bigoplus_n V^{\otimes^n}$$

- In $\mathsf{Top}_\bullet(1, \times)$ (pointed topological spaces with cartesian product) symmetric list objects are the infinite symmetric product construction:

$$1 \xrightarrow{\text{nil}} \mathrm{SP}^\infty(X) \xleftarrow{\text{cons}} X \times \mathrm{SP}^\infty(X)$$

Examples of symmetric list objects in nature

- In $\mathsf{Set}(1, \times)$ (or any Grothendieck topos over Set), symm. list objects are finite multisets:

$$1 \xrightarrow{\text{nil}} \mathcal{M}_f(A) \xleftarrow{\text{cons}} A \times \mathcal{M}_f(A)$$

- In $\mathsf{R\text{-}Mod}(R, \otimes_R)$, symmetric list objects are symmetric algebras:

$$1 \xrightarrow{\text{nil}} \bigoplus_n V^{\otimes^n} \xleftarrow{\text{cons}} V \otimes \bigoplus_n V^{\otimes^n}$$

- In $\mathsf{Top}_\bullet(1, \times)$ (pointed topological spaces with cartesian product) symmetric list objects are the infinite symmetric product construction:

$$1 \xrightarrow{\text{nil}} \mathrm{SP}^\infty(X) \xleftarrow{\text{cons}} X \times \mathrm{SP}^\infty(X)$$

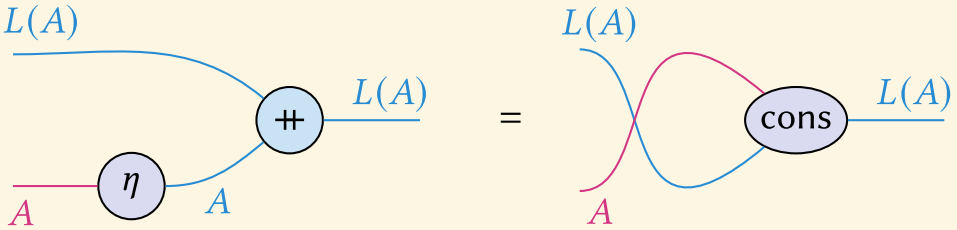
- In $\mathsf{Rel}(1, \otimes)$ (binary relations with $\otimes$), symmetric list objects are finite multisets:

$$1 \xrightarrow{\text{nil}} \mathcal{M}_f(A) \xleftarrow{\text{cons}} A \otimes \mathcal{M}_f(A)$$

Symmetric Lists

Theorem 4

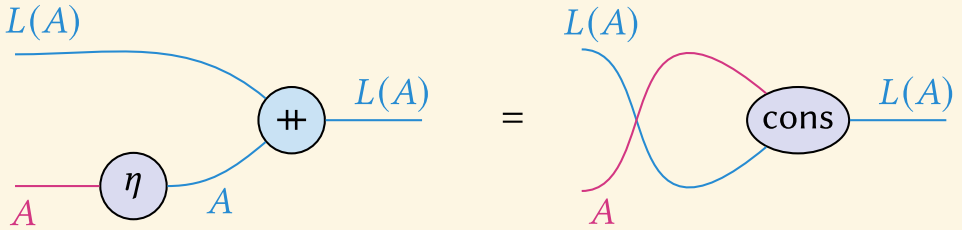
For any symmetric list $L(A)$:



Symmetric Lists

Theorem 4

For any symmetric list $L(A)$:

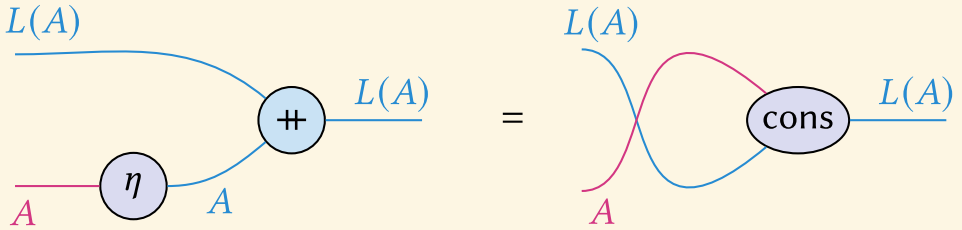


Proof. Let the two maps be f and g respectively.

Symmetric Lists

Theorem 4

For any symmetric list $L(A)$:



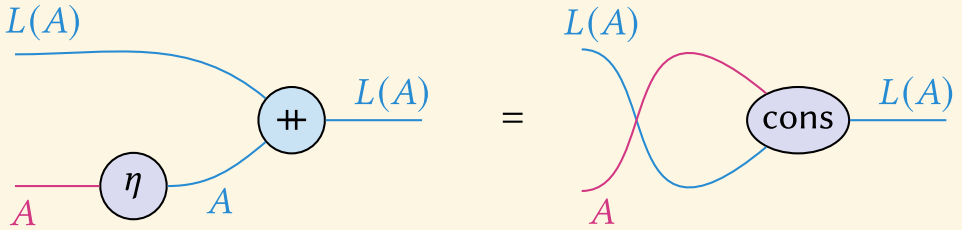
Proof. Let the two maps be f and g respectively.

We claim that $f = \mathrm{it}(\eta, \mathrm{cons}) = g$.

Symmetric Lists

Theorem 4

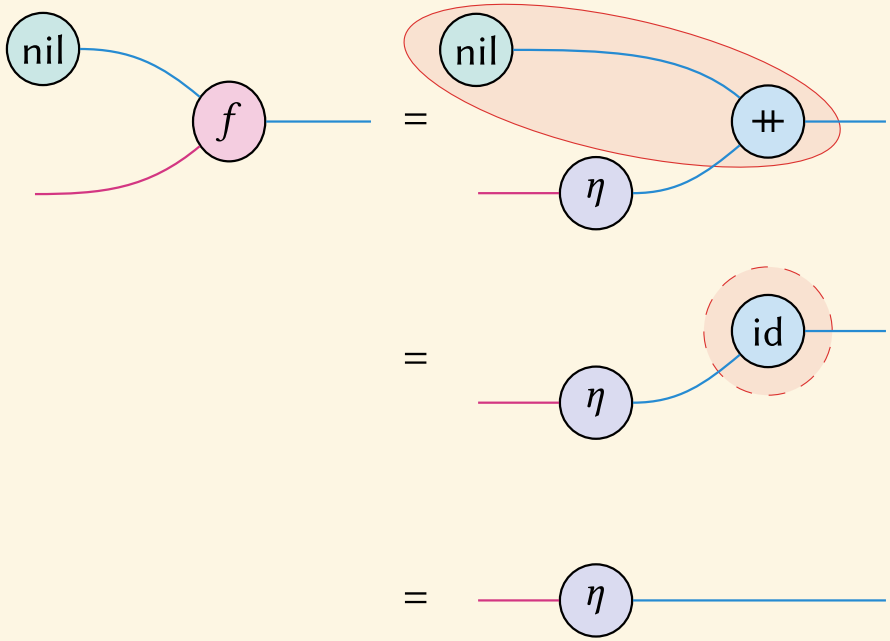
For any symmetric list $L(A)$:

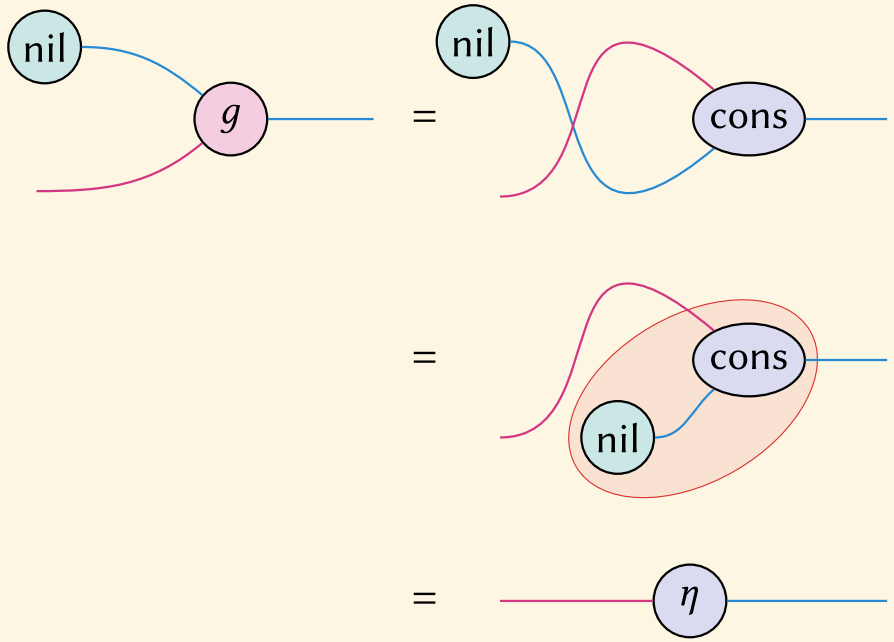


Proof. Let the two maps be f and g respectively.

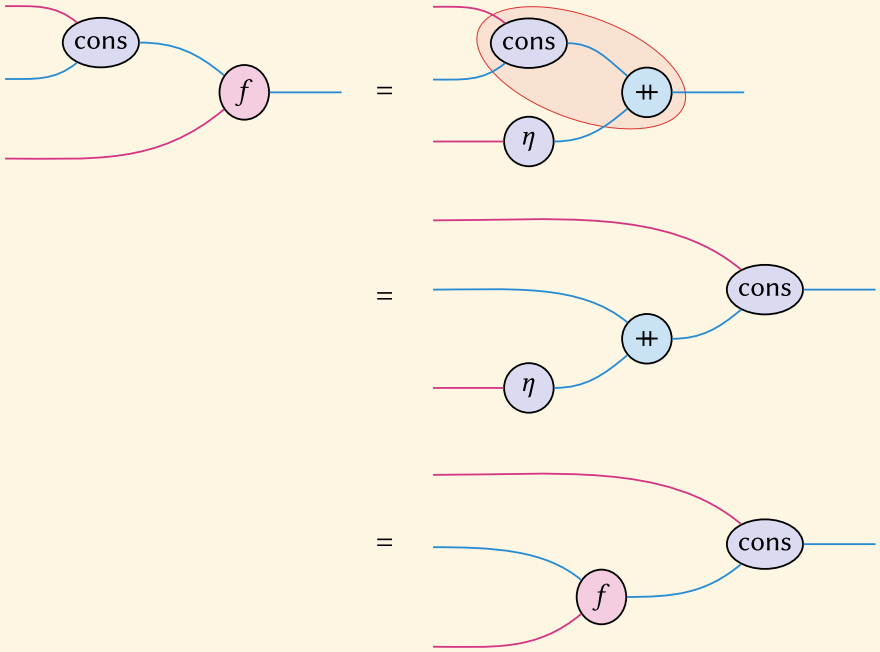
We claim that $f = \mathrm{it}(\eta, \mathrm{cons}) = g$.

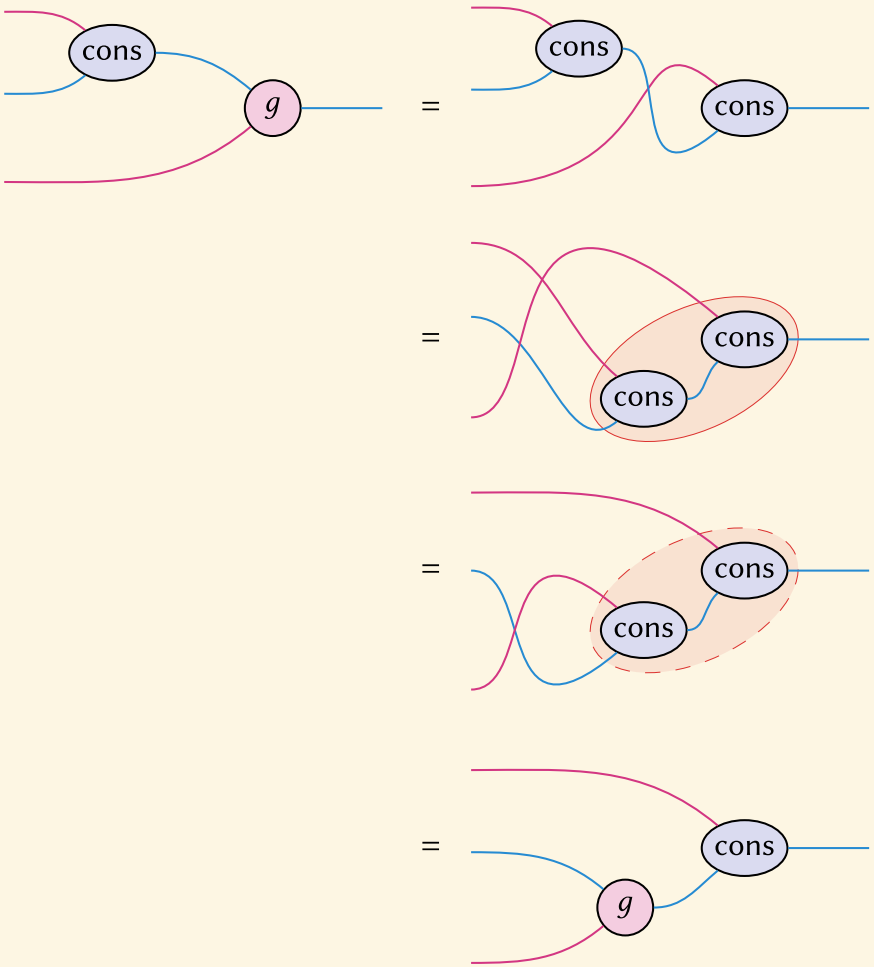
Since the iterator is unique, it suffices to show that they both satisfy the two computation rules $\mathrm{nil}\text{-}\beta$ and $\mathrm{cons}\text{-}\beta$.


nil
f
=
nil
⧺
η
=
id
η
=
η


nil
g
=
nil
cons
=
cons
nil
=
η

□


cons
f
=
cons
η
=
cons
η
=
cons
f


cons
g
=
cons
cons
=
cons
cons
=
cons
cons
=
cons
g

□

Symmetric Lists

Theorem 5

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

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

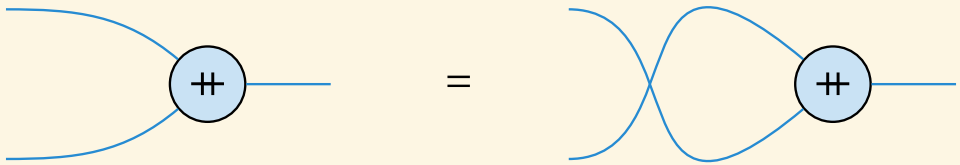
Symmetric Lists

Theorem 5

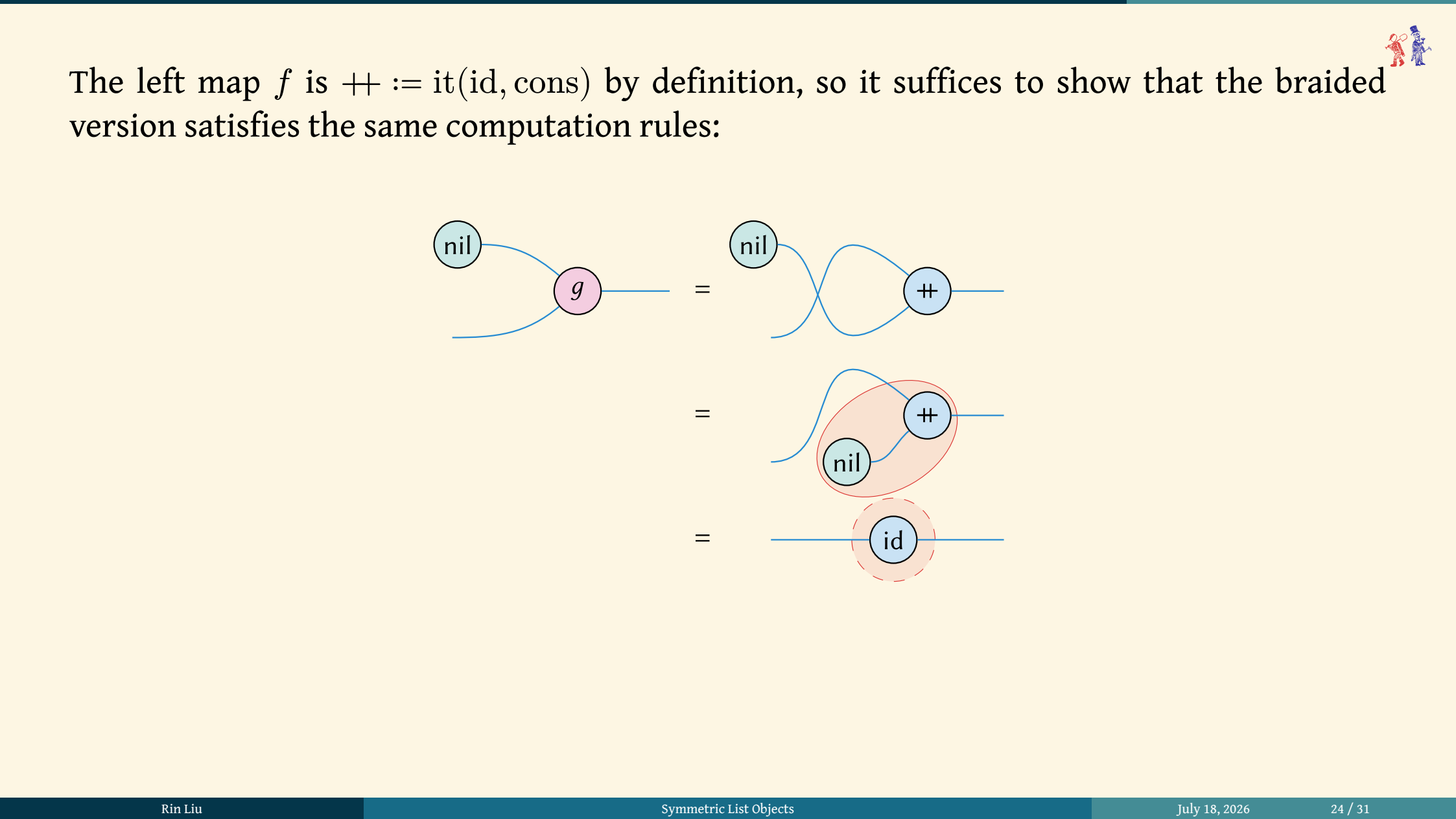
Let $(L(A), \mathrm{nil}, \mathrm{cons})$ be a list object on A. Then, the following are equivalent:

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

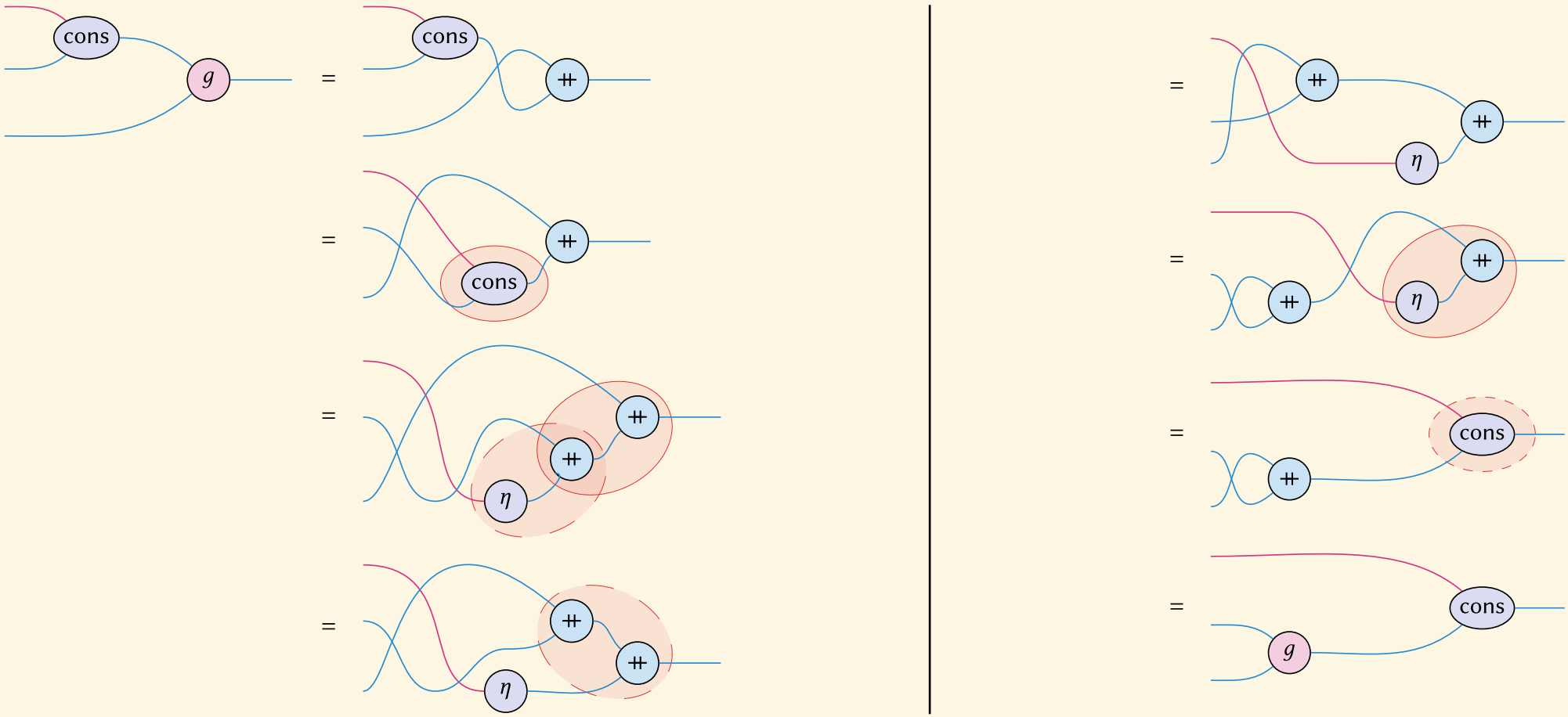
Proof. $(1 \Rightarrow 2)$ We already have that list objects are monoids, so we need only show that $+\!\!+$ is commutative:



Call the two maps f and g, respectively.



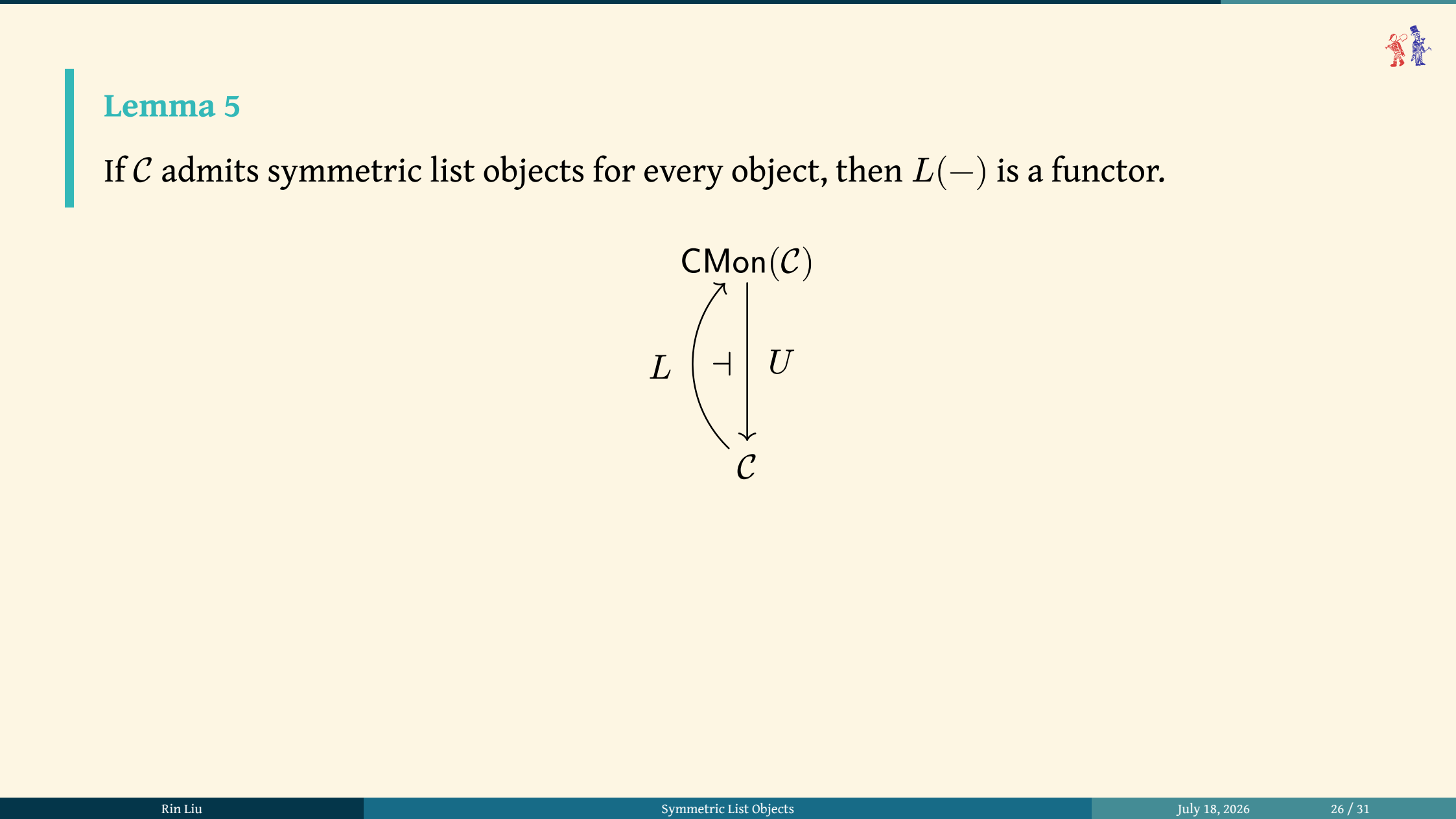
The left map f is $+\!\!+ := \mathrm{it}(\mathrm{id}, \mathrm{cons})$ by definition, so it suffices to show that the braided version satisfies the same computation rules:


cons
g
=
cons
=
cons
=
η
=
η
=
η
=
cons
=
g
cons

□

Lemma 5

If $\mathcal{C}$ admits symmetric list objects for every object, then $L(-)$ is a functor.



Lemma 5

If $\mathcal{C}$ admits symmetric list objects for every object, then $L(-)$ is a functor.

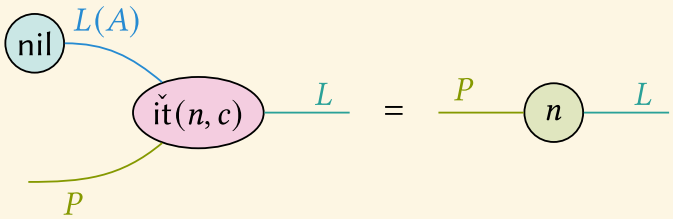
Theorem 6

If $(\mathcal{C}, \times, 1)$ is cartesian monoidal, and $L(A)$ is a list object, then $L(A)$ satisfies the following stronger universal property:

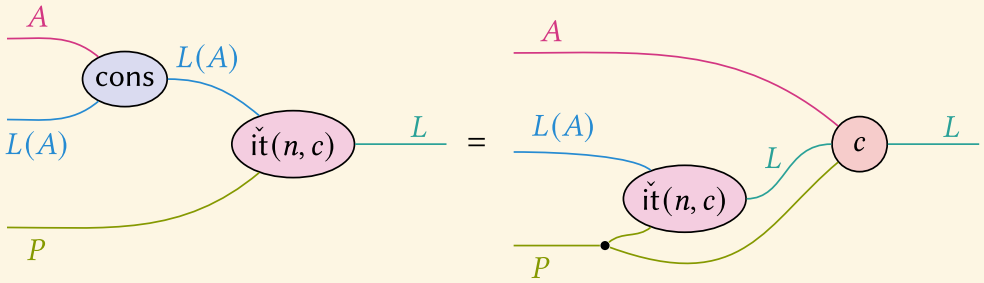
For every L with maps $P \xrightarrow{n} L \xleftarrow{c} A \times L \times P$, there exists a *doubly parametrised iterator*:

$$\check{\mathrm{it}}(n, c) : L(A) \times P \to L$$

such that the following computation rules hold:



(a) nil-β



(a) cons-β

Theorem 7

If $(\mathcal{C}, \times, 1)$ is cartesian monoidal, then the induced monad is strong, and further, commutative.

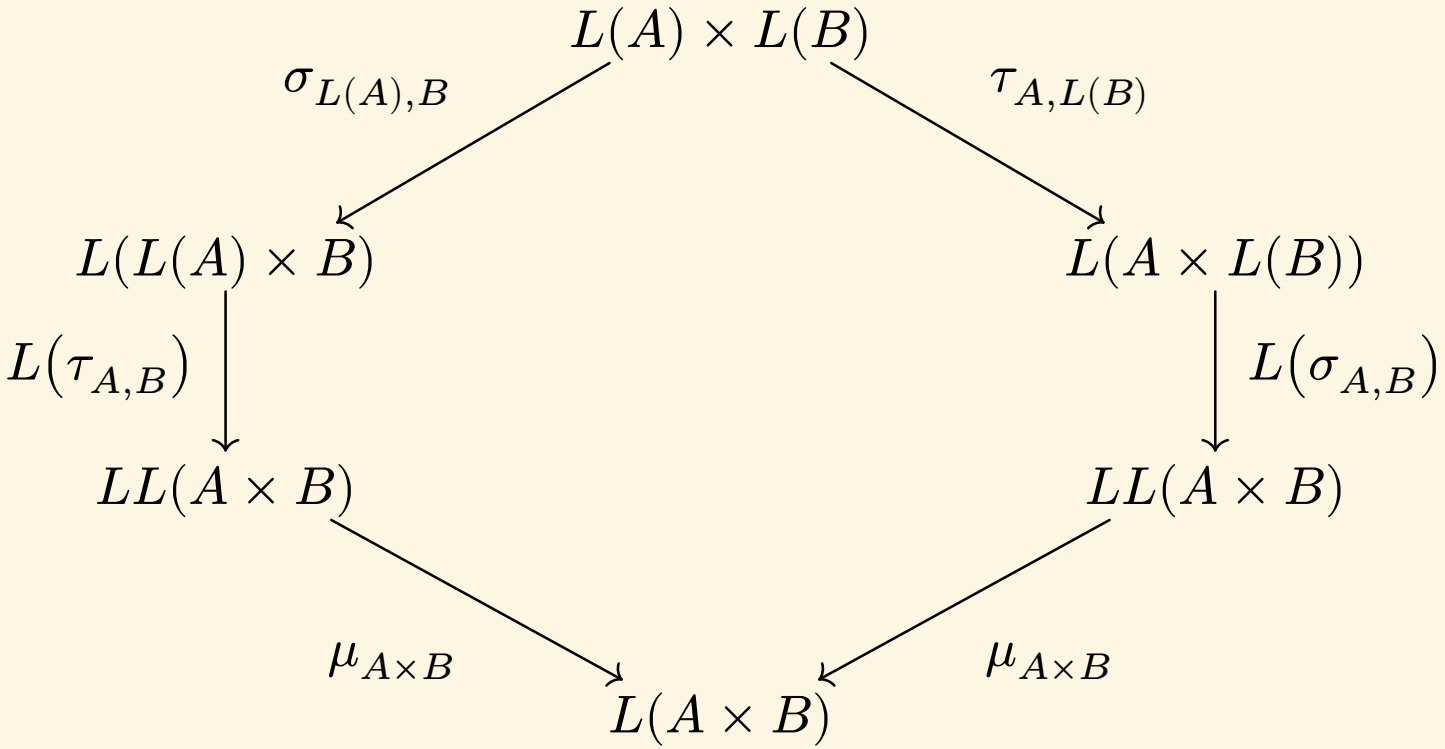
A construction of the right strength $\tau_{A,B} : L(A) \times B \to L(A \times B)$ is as follows.

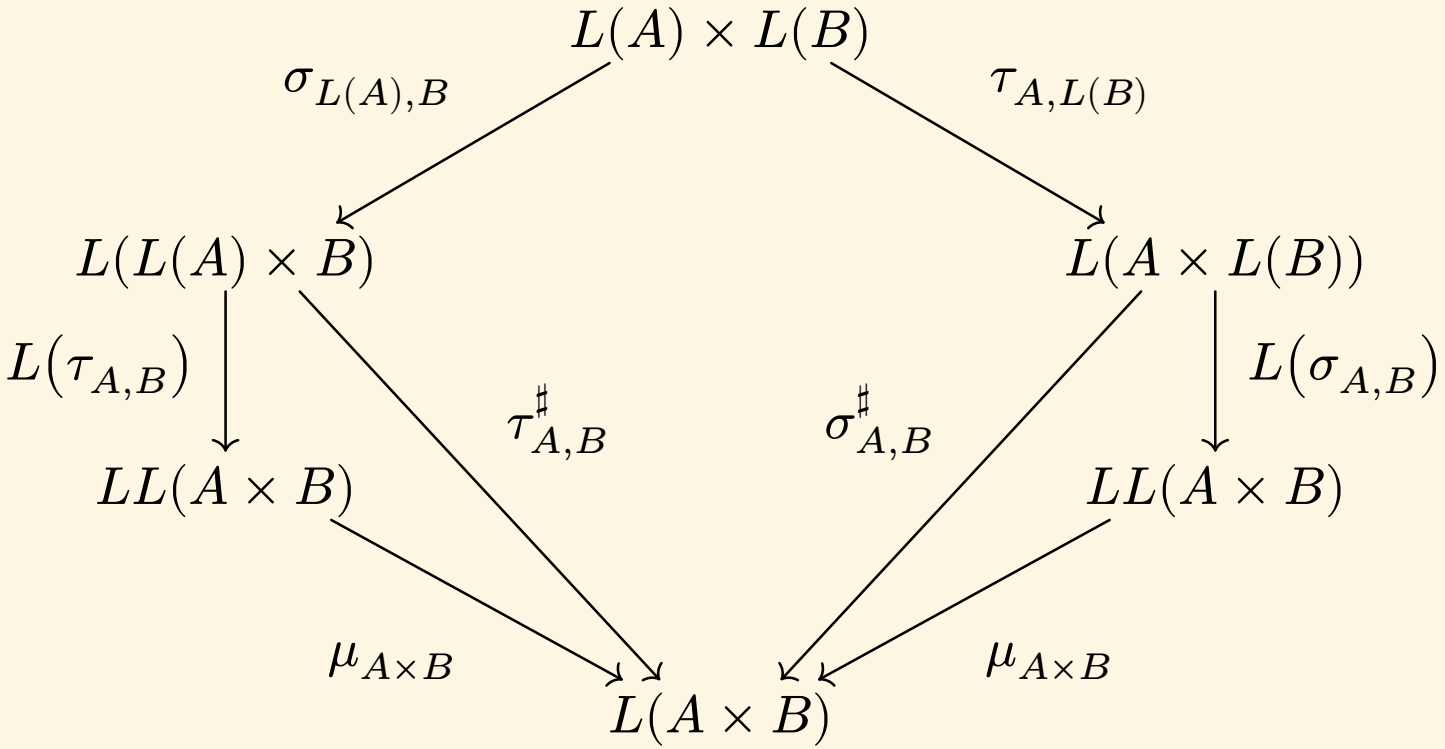
We can equip $L(A \times B)$ with a doubly-parametrised list algebra structure using:

- $n := B \xrightarrow{!} 1 \xrightarrow{\text{nil}} L(A \times B)$
- $c := A \times L(A \times B) \times B \xrightarrow{\cong} A \times B \times L(A \times B) \xrightarrow{\text{cons}} L(A \times B)$.

From the strong universal property, we obtain a map

$$\tau_{A,B} := \check{\text{it}}(n, c) : L(A) \times B \to L(A \times B)$$


$L(A) \times L(B)$
$\sigma_{L(A),B}$
$\tau_{A,L(B)}$
$L(L(A) \times B)$
$L(A \times L(B))$
$L(\tau_{A,B})$
$L(\sigma_{A,B})$
$LL(A \times B)$
$LL(A \times B)$
$\mu_{A\times B}$
$\mu_{A\times B}$
$L(A \times B)$


$L(A) \times L(B)$
$\sigma_{L(A),B}$
$\tau_{A,L(B)}$
$L(L(A) \times B)$
$L(A \times L(B))$
$L(\tau_{A,B})$
$L(\sigma_{A,B})$
$\tau^{\sharp}_{A,B}$
$\sigma^{\sharp}_{A,B}$
$LL(A \times B)$
$LL(A \times B)$
$\mu_{A\times B}$
$\mu_{A\times B}$
$L(A \times B)$

Conclusion

Summary

- Parametrised list objects give free monoids
- Symmetric list objects give free commutative monoids
- Iterator as recursion
- Doubly parametrised if Cartesian
- Strong monad if Cartesian
- Commutative monad if Symmetric

Future Work

- General theory of parametrised inital algebras
- Constructions of exponentials in LL
- Higher symmetric monoidal structures

Extended abstract with proofs