Free Commutative Monoids in Homotopy Type Theory

Vikraman Choudhury [1,2] Marcelo Fiore [2]

MFPS XXXVIII, Jul 11, 2022

[1]University of Glasgow

[2]University of Cambridge

Outline

Free commutative monoids

Relational model of Differential Linear Logic

Path space of free commutative monoids

Free commutative monoids

A commutative monoid is a monoid $(M; \cdot, e)$ with a commutation axiom.

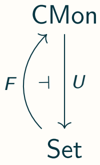
$$\mathrm{comm} : \forall x, y.\ x \cdot y = y \cdot x$$

Free commutative monoids

A commutative monoid is a monoid $(M; \cdot, e)$ with a commutation axiom.

$$\text{comm} : \forall x, y.\ x \cdot y = y \cdot x$$

The forgetful functor from CMon to Set has a left adjoint.

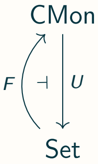


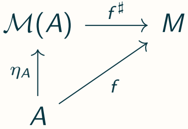
Free commutative monoids

A commutative monoid is a monoid $(M; \cdot, e)$ with a commutation axiom.

$$\text{comm} : \forall x, y.\ x \cdot y = y \cdot x$$

The forgetful functor from CMon to Set has a left adjoint.





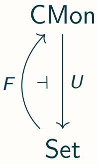
$(\mathcal{M}(A), \eta_A : A \to \mathcal{M}(A))$ is the free commutative monoid on A.

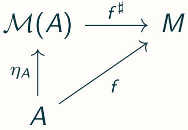
Free commutative monoids

A commutative monoid is a monoid $(M; \cdot, e)$ with a commutation axiom.

$$\mathsf{comm} : \forall x, y.\ x \cdot y = y \cdot x$$

The forgetful functor from CMon to Set has a left adjoint.





$(\mathcal{M}(A), \eta_A : A \to \mathcal{M}(A))$ is the free commutative monoid on A.

It is characterised by the universal property:

$$(-) \circ \eta_A : \mathsf{CMon}(\mathcal{M}(A), M) \xrightarrow{\sim} (A \to M)$$

Free commutative monoid

How do we *constructively* construct $\mathcal{M}(A)$?

Free commutative monoid

How do we *constructively* construct $\mathcal{M}(A)$?

Free monoids are lists.

Free commutative monoid

How do we *constructively* construct $\mathcal{M}(A)$?

Free monoids are lists.

Free commutative monoids are:

- unordered lists, or
- lists upto permutation of elements, or
- finite-multisets, or
- bags

Free commutative monoid

How do we *constructively* construct $\mathcal{M}(A)$?

Free monoids are lists.

Free commutative monoids are:

- unordered lists, or
- lists upto permutation of elements, or
- finite-multisets, or
- bags

We want to define them in univalent type theory:

- without assuming decidable equality,
- and prove the universal property.

Construction of the free commutative monoid

Two easy definitions using HITs:

$$
\begin{aligned}
\underline{\mathsf{ACM}(A)} :\equiv & \\
\eta &: A \to \mathsf{ACM}(A) \\
e &: \mathsf{ACM}(A) \\
- \cdot - &: \mathsf{ACM}(A)^2 \to \mathsf{ACM}(A) \\
\mathsf{assoc} &: x \cdot (y \cdot z) = (x \cdot y) \cdot z \\
\mathsf{unitl} &: e \cdot x = x \\
\mathsf{unitr} &: x \cdot e = x \\
\mathsf{comm} &: x \cdot y = y \cdot x \\
\mathsf{trunc} &: \mathsf{isSet}(\mathsf{ACM}(A))
\end{aligned}
$$

Construction of the free commutative monoid

Two easy definitions using HITs:

$$
\begin{aligned}
\underline{\mathsf{ACM}(A)} &:\equiv \\
\eta &: A \to \mathsf{ACM}(A) \\
e &: \mathsf{ACM}(A) \\
- \cdot - &: \mathsf{ACM}(A)^2 \to \mathsf{ACM}(A) \\
\mathsf{assoc} &: x \cdot (y \cdot z) = (x \cdot y) \cdot z \\
\mathsf{unitl} &: e \cdot x = x \\
\mathsf{unitr} &: x \cdot e = x \\
\mathsf{comm} &: x \cdot y = y \cdot x \\
\mathsf{trunc} &: \mathsf{isSet}(\mathsf{ACM}(A))
\end{aligned}
$$

$$
\begin{aligned}
\underline{\mathsf{sList}(A)} &:\equiv \\
\mathsf{nil} &: \mathsf{sList}(A) \\
- :: - &: A \times \mathsf{sList}(A) \to \mathsf{sList}(A) \\
\mathsf{swap} &: x :: y :: xs = y :: x :: xs \\
\mathsf{trunc} &: \mathsf{isSet}(\mathsf{sList}(A))
\end{aligned}
$$

Construction of the free commutative monoid

Two easy definitions using HITs:

$\underline{\mathsf{ACM}(A)} :\equiv$

$$\begin{aligned} \eta &: A \to \mathsf{ACM}(A) \\ e &: \mathsf{ACM}(A) \\ - \cdot - &: \mathsf{ACM}(A)^2 \to \mathsf{ACM}(A) \\ \mathsf{assoc} &: x \cdot (y \cdot z) = (x \cdot y) \cdot z \\ \mathsf{unitl} &: e \cdot x = x \\ \mathsf{unitr} &: x \cdot e = x \\ \mathsf{comm} &: x \cdot y = y \cdot x \\ \mathsf{trunc} &: \mathsf{isSet}(\mathsf{ACM}(A)) \end{aligned}$$

$\underline{\mathsf{sList}(A)} :\equiv$

$$\begin{aligned} \mathsf{nil} &: \mathsf{sList}(A) \\ - :: - &: A \times \mathsf{sList}(A) \to \mathsf{sList}(A) \\ \mathsf{swap} &: x :: y :: xs = y :: x :: xs \\ \mathsf{trunc} &: \mathsf{isSet}(\mathsf{sList}(A)) \end{aligned}$$

Both satisfy the categorical universal property of free comm. monoids.

$$\mathcal{M}(A) :\equiv \mathsf{ACM}(A) \simeq_{\mathsf{CMon}} \mathsf{sList}(A)$$

Free commutative monoid monad

- Monad structure:

$$\eta_A : A \to \mathcal{M}(A)$$
$$\mu_A :\equiv (\lambda_{(x\,:\,A)}.\, x)^\sharp : \mathcal{M}(\mathcal{M}(A)) \to \mathcal{M}(A)$$

- Functorial action on $f : A \to B$:

$$\mathcal{M}(f) :\equiv (\lambda_{(a\,:\,A)}.\, \eta_B(fa))^\sharp : \mathcal{M}(A) \to \mathcal{M}(B)$$

- Monad strength:

$$\sigma_{A,B} : \mathcal{M}(A) \times B \to \mathcal{M}(A \times B) : (as, b) \mapsto \mathcal{M}(\lambda_{(a\,:\,A)}.\,(a, b))(as)$$
$$\tau_{A,B} : A \times \mathcal{M}(B) \to \mathcal{M}(A \times B) : (a, bs) \mapsto \mathcal{M}(\lambda_{(b\,:\,B)}.\,(a, b))(bs)$$

- Commutative monad structure:

$$\begin{array}{ccc}
\mathcal{M}(A) \times \mathcal{M}(B) & \xrightarrow{\sigma_{A,\mathcal{M}(B)}} & \mathcal{M}(A \times \mathcal{M}(B)) \\
\Big\downarrow{\scriptstyle \tau_{\mathcal{M}(A),B}} & & \Big\downarrow{\scriptstyle {\tau_{A,B}}^\sharp} \\
\mathcal{M}(\mathcal{M}(A) \times B) & \xrightarrow[{\sigma_{A,B}}^\sharp]{} & \mathcal{M}(A \times B)
\end{array}$$

Free commutative monoid monad

- Strong symmetric monoidal functor:

$$\begin{array}{ccc} \mathcal{M}(A) \times \mathcal{M}(B) & \xrightarrow{\quad\simeq\quad} & \mathcal{M}(A+B) \\ {\scriptstyle \mathcal{M}(\imath_1)\times\mathcal{M}(\imath_2)} \searrow & & \nearrow {\scriptstyle \oplus_{(A+B)}} \\ & \mathcal{M}(A+B) \times \mathcal{M}(A+B) & \end{array}$$

$$\mathbf{1} \xrightarrow[\lambda(x\,:\,\mathbf{1}).\,\varnothing_{\mathbf{0}}]{\simeq} \mathcal{M}(\mathbf{0})$$

- Length function:

$$\ell_A :\equiv \mathcal{M}(\lambda_{(a\,:\,A)}.\,\star) : \mathcal{M}(A) \to \mathcal{M}(\mathbf{1})$$

Outline

Free commutative monoids

Relational model of Differential Linear Logic

Path space of free commutative monoids

Category of relations

Power objects: $\mathfrak{P} : \mathsf{hSet}_i \to \mathsf{hSet}_{i+1} : A \longmapsto (A \to \mathsf{hProp}_i)$.

Category of relations

Power objects: $\mathfrak{P} : \mathsf{hSet}_i \to \mathsf{hSet}_{i+1} : A \longmapsto (A \to \mathsf{hProp}_i)$.

Power relative monad:

- unit: $よ_A : A \to \mathfrak{P}(A) : a \longmapsto \lambda_{(x\,:\,A)}.\, a =_A x$
- extension for $f : A \to \mathfrak{P}(B)$:
  $f^* : \mathfrak{P}(A) \to \mathfrak{P}(B) : (\alpha, b) \longmapsto \exists_{(a:A)}.f(a,b) \wedge \alpha(a)$

Category of relations

Power objects: $\mathfrak{P} : \mathrm{hSet}_i \to \mathrm{hSet}_{i+1} : A \longmapsto (A \to \mathrm{hProp}_i)$.

Power relative monad:

- unit: $よ_A : A \to \mathfrak{P}(A) : a \longmapsto \lambda(x : A).\, a =_A x$
- extension for $f : A \to \mathfrak{P}(B)$:
  $f^* : \mathfrak{P}(A) \to \mathfrak{P}(B) : (\alpha, b) \longmapsto \exists(a{:}A).f(a, b) \wedge \alpha(a)$

Rel has objects hSets and homs $A \nrightarrow B :\equiv A \to \mathfrak{P}(B)$.

- Rel is dagger compact.
- $(-)_* : \mathrm{Set} \to \mathrm{Rel}$ maps functions $f : A \to B$ to relations $よ_B \circ f : A \nrightarrow B$.
- $(-)_*$ preserves coproducts, which become biproducts.

Lifting $\mathcal{M}$ to Rel

$\mathcal{M}$ lifts to the cofree commutative comonoid in Rel.

- comonad structure

$$\delta_A :\equiv ((\mu_A)_*)^\dagger : \mathcal{M}(A) \nrightarrow \mathcal{M}(\mathcal{M}(A))$$

$$\epsilon_A :\equiv ((\eta_A)_*)^\dagger : \mathcal{M}(A) \nrightarrow A$$

- commutative comonoid structure

$$w_A :\equiv ((+\!\!+_A)_*)^\dagger : \mathcal{M}(A) \nrightarrow \mathcal{M}(A) \otimes \mathcal{M}(A)$$

$$k_A :\equiv ((\lambda_{(x\,:\,\mathbf{1})}.\,\mathsf{nil})_*)^\dagger : \mathcal{M}(A) \nrightarrow \mathbf{1}$$

The universal property follows from promonoidal convolution (Day 70).

Lifting $\mathcal{M}$ to Rel

$\mathcal{M}$ lifts to the cofree commutative comonoid in Rel.

- comonad structure

$$\delta_A :\equiv ((\mu_A)_*)^\dagger : \mathcal{M}(A) \nrightarrow \mathcal{M}(\mathcal{M}(A))$$
$$\epsilon_A :\equiv ((\eta_A)_*)^\dagger : \mathcal{M}(A) \nrightarrow A$$

- commutative comonoid structure

$$w_A :\equiv ((+\!\!+_A)_*)^\dagger : \mathcal{M}(A) \nrightarrow \mathcal{M}(A) \otimes \mathcal{M}(A)$$
$$k_A :\equiv ((\lambda_{(x\,:\,\mathbf{1})}.\,\mathrm{nil})_*)^\dagger : \mathcal{M}(A) \nrightarrow \mathbf{1}$$

The universal property follows from promonoidal convolution (Day 70).

- monoidal structure (Seely isomorphisms)

$$\varphi_{A,B} :\equiv (\epsilon_A \otimes \epsilon_B)_\sharp : \mathcal{M}(A) \otimes \mathcal{M}(B) \xrightarrow{\sim} \mathcal{M}(A \otimes B)$$
$$\phi :\equiv (\mathrm{id}_{\mathbf{1}})_\sharp : \mathbf{1} \xrightarrow{\sim} \mathcal{M}(\mathbf{1})$$

Differential Structure

Combinatorics of subsingleton multisets:

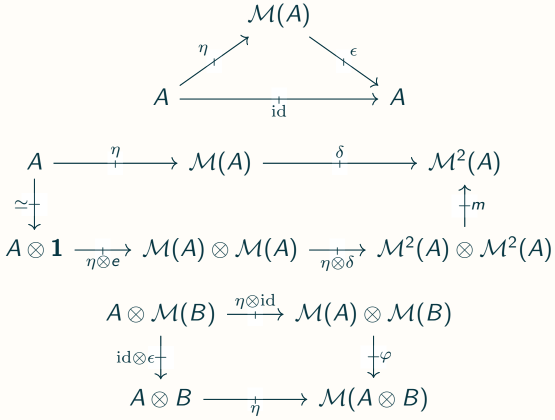
- conical-monoid relation: $as \mathbin{+\!\!+} bs = \mathrm{nil} \iff as = bs = \mathrm{nil}$
- η_A is an embedding: $x =_A y \iff [x] =_{\mathcal{M}(A)} [y]$
- $A \simeq \sum_{as:\mathcal{M}(A)} (\ell(as) = 1) \simeq \sum_{as:\mathcal{M}(A)} \sum_{a:A} (as = [a])$
-
$$[a] = \mu(s) \iff \exists_{(t:\mathcal{M}(\mathcal{M}(A)))}.\ \mu(t) = \mathrm{nil} \wedge [a] :: t = s$$
-
$$[a] = \mathcal{M}(\pi_1)(ps) \wedge bs = \mathcal{M}(\pi_2)(ps) \iff \exists_{(b:B)}.\ bs = [b] \wedge \big[(a, b)\big] = ps$$

Differential Structure

Creation map:

$$\eta_A : A \rightarrow\!\!\!\!\!+ \ \mathcal{M}(A)$$

subject to three laws as follows:



The co-Kleisli category of $\mathcal{M}$:

- has homs $\mathcal{M}(A) \relbar\joinrel\mapsto B$
- is cartesian closed
- is a cartesian differential category

$\mathcal{M}$Rel

The co-Kleisli category of $\mathcal{M}$:

- has homs $\mathcal{M}(A) \twoheadrightarrow B$
- is cartesian closed
- is a cartesian differential category

These are the set-truncated version of generalised species of structures (Fiore, Gambino, Hyland, Winskel 2008).

Outline

Free commutative monoids

Relational model of Differential Linear Logic

Path space of free commutative monoids

Bialgebra law

Every set has a biproduct commutative bialgebra structure.

$$
\begin{array}{ccccc}
A + A & \xrightarrow{\nabla} & A & \xrightarrow{\Delta} & A + A \\
{\scriptstyle \Delta+\Delta}\downarrow & & & & \uparrow{\scriptstyle \nabla+\nabla} \\
A + A + A + A & & \xrightarrow[\mathrm{id}_A + c + \mathrm{id}_A]{} & & A + A + A + A
\end{array}
$$

By the Seely isomorphism, this transfers to the bialgebra law.

$$
\begin{array}{ccccc}
\mathcal{M}(A) \otimes \mathcal{M}(A) & \xrightarrow{m} & \mathcal{M}(A) & \xrightarrow{w} & \mathcal{M}(A) \otimes \mathcal{M}(A) \\
{\scriptstyle w\otimes w}\downarrow & & & & \uparrow{\scriptstyle m\otimes m} \\
\mathcal{M}(A) \otimes \mathcal{M}(A) \otimes \mathcal{M}(A) \otimes \mathcal{M}(A) & & \xrightarrow[\mathrm{id}_{\mathcal{M}(A)} \otimes c \otimes \mathrm{id}_{\mathcal{M}(A)}]{} & & \mathcal{M}(A) \otimes \mathcal{M}(A) \otimes \mathcal{M}(A) \otimes \mathcal{M}(A)
\end{array}
$$

where $c :\equiv (\langle \pi_2, \pi_1 \rangle)_*$ is the symmetry isomorphism.

Commutation relation

Riesz refinement-monoid relation:

$$as \mathbin{+\!\!+} bs = cs \mathbin{+\!\!+} ds$$

$$\iff$$

$$\exists_{(xs_1, xs_2, ys_1, ys_2 : \mathcal{M}(A))}.\ (as = xs_1 \mathbin{+\!\!+} xs_2)\ \wedge\ (bs = ys_1 \mathbin{+\!\!+} ys_2)$$
$$\wedge\ (xs_1 \mathbin{+\!\!+} ys_1 = cs) \wedge (xs_2 \mathbin{+\!\!+} ys_2 = ds)$$

Commutation relation

Riesz refinement-monoid relation:

$$as ++ bs = cs ++ ds$$
$$\iff$$
$$\exists_{(xs_1, xs_2, ys_1, ys_2 : \mathcal{M}(A))}.\ (as = xs_1 ++ xs_2) \ \wedge\ (bs = ys_1 ++ ys_2)$$
$$\wedge\ (xs_1 ++ ys_1 = cs) \wedge (xs_2 ++ ys_2 = ds)$$

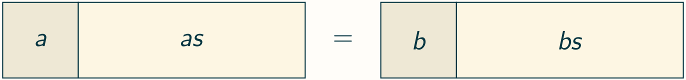
Commutation relation:

$$a :: as = b :: bs$$
$$\Leftrightarrow$$
$$(a = b \wedge as = bs) \vee (\exists_{(cs : \mathcal{M}(A))}.\ as = b :: cs \wedge a :: cs = bs)$$

This commutation relation comes from the creation/annihilation operators associated with the free commutative monoid construction seen as a combinatorial Fock space (Fiore 2015).

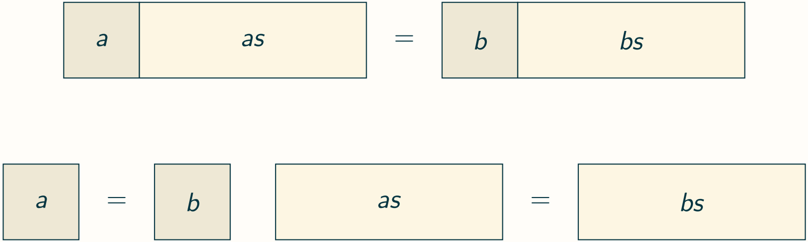
Commutation relation

Pointwise equality:



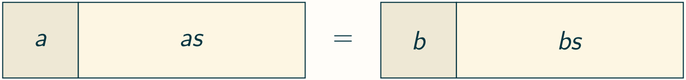
Commutation relation

Pointwise equality:



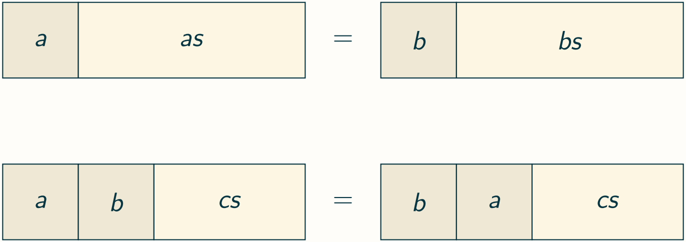
Commutation relation

Generalised swapping operation:



Commutation relation

Generalised swapping operation:



Deduction system

Deduction system for multiset equality:

$$\frac{}{\text{nil} \sim \text{nil}}\ \text{nil-cong} \qquad\qquad \frac{a = b \qquad as \sim bs}{a :: as \sim b :: bs}\ \text{cons-cong}$$

$$\frac{as \sim b :: cs \qquad a :: cs \sim bs}{a :: as \sim b :: bs}\ \text{comm}$$

Deduction system

Deduction system for multiset equality:

$$\frac{}{\text{nil} \sim \text{nil}}\ \text{nil-cong} \qquad \frac{a = b \quad as \sim bs}{a :: as \sim b :: bs}\ \text{cons-cong}$$

$$\frac{as \sim b :: cs \quad a :: cs \sim bs}{a :: as \sim b :: bs}\ \text{comm}$$

The relation $\sim$ generates the path space of $\mathcal{M}(A)$:

$$(as = bs) \Leftrightarrow \|as \sim bs\|\,.$$

Deduction system

The $\sim$ relation is transitive (admits cut):

$$\frac{as \sim bs \qquad bs \sim cs}{as \sim cs}$$

Deduction system

The $\sim$ relation is transitive (admits cut):

$$\frac{as \sim bs \qquad bs \sim cs}{as \sim cs}$$

Given two deduction trees, we compute the underlying permutations, compose them, and reify it back to a tree (NbE).

Deduction system

The $\sim$ relation is transitive (admits cut):

$$\frac{as \sim bs \qquad bs \sim cs}{as \sim cs}$$

Given two deduction trees, we compute the underlying permutations, compose them, and reify it back to a tree (NbE).

$$\mathsf{vec} \;:\; \mathcal{L}(A) \;\simeq\; \left(\textstyle\sum_{\ell:\mathbb{N}} \mathsf{Fin}_\ell \to A\right) \;:\; \mathsf{list}$$

$$(m, f) \approx_A (n, g) :\equiv (\phi : \mathsf{Fin}_m \xrightarrow{\sim} \mathsf{Fin}_n) \times (f = g \circ \phi) \;.$$

For $as, bs : \mathcal{L}(A)$, we have

$$\mathsf{eval} : \; as \sim_A bs \to \mathsf{vec}(as) \approx_A \mathsf{vec}(bs)$$

and, for $(m, f), (n, g) : \left(\sum_{\ell:\mathbb{N}} \mathsf{Fin}_\ell \to A\right)$, we have

$$\mathsf{quote} : \; (m, f) \approx_A (n, g) \to \mathsf{list}(m, f) \sim_A \mathsf{list}(n, g)$$

Commuted-list construction

The composite $A \to \mathcal{L}(A) \to \mathcal{L}(A)_{/\bar{\sim}_A}$ is the free comm. monoid on A.

Commuted-list construction

The composite $A \to \mathcal{L}(A) \to \mathcal{L}(A)_{/\tilde{\sim}_A}$ is the free comm. monoid on A.

Alternatively, we can define another HIT with a conditional path constructor comm.

$$
\begin{aligned}
\underline{\mathsf{cList}(A)} &:\equiv \\
\mathsf{nil} &: \mathsf{cList}(A) \\
\_ :: \_ &: A \times \mathsf{cList}(A) \to \mathsf{cList}(A) \\
\mathsf{comm} &: \{a\ b : A\}\{as\ bs\ cs : \mathsf{cList}(A)\} \\
&\to (as = b :: cs) \to (a :: cs = bs) \\
&\to a :: as = b :: bs \\
\mathsf{trunc} &: \mathsf{isSet}(\mathsf{cList}(A))
\end{aligned}
$$

Epilogue

Summary:

- Different constructions of free commutative monoids:

$$\mathsf{ACM}(A) \simeq_{\mathsf{CMon}} \mathsf{sList}(A) \simeq_{\mathsf{CMon}} \mathcal{L}(A)_{/\tilde{\approx}_A} \simeq_{\mathsf{CMon}} \mathsf{cList}(A)$$

- Formal construction of the relational model of differential linear logic
- Constructive combinatorics of free commutative monoids:
  - Subsingleton multisets
  - Conical and Refinement-monoid relations
  - Commutation relation
  - Characterisation of the path space
- More details in the paper and formalisation!

Epilogue

Summary:

- Different constructions of free commutative monoids:

$$\mathsf{ACM}(A) \simeq_{\mathsf{CMon}} \mathsf{sList}(A) \simeq_{\mathsf{CMon}} \mathcal{L}(A)_{/\tilde{\approx}_A} \simeq_{\mathsf{CMon}} \mathsf{cList}(A)$$

- Formal construction of the relational model of differential linear logic
- Constructive combinatorics of free commutative monoids:
  - Subsingleton multisets
  - Conical and Refinement-monoid relations
  - Commutation relation
  - Characterisation of the path space
- More details in the paper and formalisation!

Future work:

- Generalise to free symmetric monoidal groupoids
- Construction of the bicategory of generalised species of structures over groupoids and its differential structure