# (Q)M-types and Coinduction in HoTT / CTT Master's Thesis, Computer Science

Lasse Letager Hansen, 201508114

Supervisor: Bas Spitters

Aarhus University

June 28, 2020



## Overview

- Introduction
  - Goals
- M-types
  - Definition of M-types
  - Construction of M-types and Examples
  - Equality for Coinductive types
- Quotient M-types
  - Propositional Truncation and Set Truncated Quotient
- 4 Conclusion

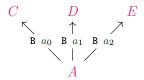
## Goals

- Formalize coinductive types as M-types
- Define equality for coinductive types
- Explore ways to define quotiented M-types

# Containers and Polynomial functors

### Definition

A Container (or signature) is a dependent pair S = (A, B) for the types  $A : \mathcal{U}$  and  $B : A \to \mathcal{U}$ .



# Containers and Polynomial functors

#### Definition

A polynomial functor  $P_S$  (or extension) for a container  $S=(A\,,B)$  is defined, for types as

$$P_S X = \sum_{(a:A)} Ba \to X \tag{1}$$

and for a function  $\mathbf{f}: X \to Y$  as

$$P_{S} f (a, g) = (a, f \circ g).$$
 (2)

## Chain

## Definition (Chain)

We define a chain as a family of morphisms  $\pi_{(n)}: X_{n+1} \to X_n$ , over a family of types  $X_n$ . See figure.

$$X_0 \leftarrow \pi_{(0)}$$
  $X_1 \leftarrow \pi_{(1)}$   $\cdots \leftarrow \pi_{(n-1)}$   $X_n \leftarrow \pi_{(n)}$   $X_{n+1} \leftarrow \pi_{(n+1)}$   $\cdots$ 

#### Definition

The limit of a chain is given as

$$\mathcal{L} = \sum_{(x:\prod_{(n:\mathbb{N})}X_n)} \prod_{(n:\mathbb{N})} (\pi_{(n)} \ x_{n+1} \equiv x_n)$$
(3)

We let  $M_S$  be the limit, for a chain defined by  $X_n = P^n \mathbf{1}$ , and  $\pi_{(n)} = P^n \mathbf{!}$ 

# Equality between $\mathcal L$ and P $\mathcal L$

### Theorem

There is an equality

$$shift: M \equiv PM \tag{4}$$

from which we can define helper functions

$$in: PM \to M$$
 out:  $M \to PM$  (5)

# Equality between ${\cal L}$ and P ${\cal L}$

#### Theorem

There is an equality

$$shift: \mathbf{M} \equiv \mathbf{P}\,\mathbf{M} \tag{4}$$

from which we can define helper functions

$$in: PM \to M$$
 out:  $M \to PM$  (5)

### Proof structure

The proof is done using the two helper lemmas

$$\alpha: \mathcal{L}^{\mathsf{P}} \equiv \mathsf{P} \, \mathcal{L} \tag{6}$$

$$\mathcal{L}unique: \mathcal{L} \equiv \mathcal{L}^{\mathbf{P}} \tag{7}$$

where  $\mathcal{L}^{P}$  is the limit of the shifted chain defined as  $X'_{n} = X_{n+1}$  and  $\pi'_{(n)} = \pi_{(n+1)}$ . With these two lemmas we get  $shift = \alpha \cdot \mathcal{L}unique$ .

#### M-types are final coalgebras

We want to show that M-types are final coalgebras.

#### Definition

A P-coalgebra is defined as

$$\sum_{(C:\mathcal{U})} C \to P C \tag{8}$$

which we denote C- $\gamma$ . We define P-coalgebra morphisms as

$$C - \gamma \Rightarrow D - \delta = \sum_{(\mathbf{f}: C \to D)} \delta \circ \mathbf{f} \equiv \mathsf{Pf} \circ \gamma \tag{9}$$

A coalgebra is final, if the following is true

$$\sum_{(D-\rho)} \prod_{(C-\gamma)} \text{isContr} (C-\gamma \Rightarrow D-\rho)$$
 (10)

# M-types are final coalgebras

### Theorem

M-types are final coalgebras. That is Final<sub>S</sub> M.

9/32

# M-types are final coalgebras

#### **Theorem**

M-types are final coalgebras. That is  $Final_S$  M.

### Proof structure

The definition of finality is

$$\prod_{\substack{(C-\gamma:\mathsf{Coalg}_S)}} \mathsf{isContr} \; (C-\gamma \Rightarrow \mathsf{M-out}) \tag{11}$$

which we show by  $(C-\gamma \Rightarrow M-out) \equiv 1$ .

## Overview

- Introduction
  - Goals
- 2 M-types
  - Definition of M-types
  - Construction of M-types and Examples
  - Equality for Coinductive types
- Quotient M-types
  - Propositional Truncation and Set Truncated Quotient
- 4 Conclusion

## Example: Delay Monad

A delay monad is defined by the two constructors

$$\frac{r:R}{\text{now }r:\text{Delay }R} \quad \text{(12)} \qquad \frac{t:\text{Delay }R}{\text{later }t:\text{Delay }R} \quad \text{(13)}$$

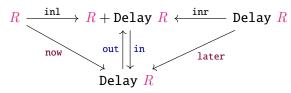
We define a container

$$(R+\mathbf{1},[\mathbf{0},\mathbf{1}]) \tag{14}$$

and a polynomial functor

$$PX = \sum_{(x:R+1)} \begin{cases} \mathbf{0} & x = \text{inl } r \\ \mathbf{1} & x = \text{inr } \star \end{cases} \to X = R + X, \tag{15}$$

such that we get the diagram



# Rules for Constructing M-types

Adding containers (A, B) and (C, D) for two constructors together is done by

$$\begin{pmatrix}
A + C, & B a & \text{inl } a \\
D c & \text{inr } c
\end{pmatrix}$$
(16)

whereas adding containers for two destructors is done by

$$(A \times C, \lambda(a, c), Ba + Dc)$$
 (17)

However combining both destructors and constructors is not as simple. Which is similar to rules for coinductive records.

We can take a coinductive record and transform it to an M-type. The types of fields in a coinductive records are

- non-dependent fields
- dependent fields
- recursive fields
- dependent and recursive fields

We will give some examples of these.

Example: Record to M-type

Lets try and convert the record

record 
$$tree: \mathcal{U}$$
 where

$$value: \mathbb{N}$$
  $left\text{-}child: tree$ 

right-child: tree

To an M-type defined by the container

$$(\mathbb{N}, \mathbf{1} + \mathbf{1}) \tag{19}$$

(18)

Example: Record to M-type

Lets try and convert the record

record 
$$bet : \mathcal{U}$$
 where  $value_a : \mathbb{N}$   $value_b : \mathbb{N}$   $winner : value_a \le value_b \to bool$  (20)

To an M-type defined by the container

$$\left(\sum_{(value_a:\mathbb{N})} \sum_{(value_b:\mathbb{N})} value_a \le value_b \to bool, \mathbf{0}\right)$$
(21)

Example: Record to M-type

Lets try and convert the record

record example 
$$A: \mathcal{U}$$
 where

$$value: A$$
 $index-type: \mathcal{U}$ 
(22)

 $continue: index-type \rightarrow \texttt{example}\ A$ 

To an M-type defined by the container

$$(A \times \mathcal{U}, \lambda (\underline{\ }, index-type), index-type)$$
 (23)

## Overview

- Introduction
  - Goals
- 2 M-types
  - Definition of M-types
  - Construction of M-types and Examples
  - Equality for Coinductive types
- Quotient M-types
  - Propositional Truncation and Set Truncated Quotient
- 4 Conclusion

## Example: Streams

We can now define streams for a given type A, as a records

record Stream 
$$A: \mathcal{U}$$
 where

$$hd: \mathbf{A} \tag{24}$$

 $tl: \mathtt{Stream}\ A$ 

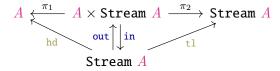
corresponding to the container

$$(A, \mathbf{1}) \tag{25}$$

for which we get the polynomial functor

$$PX = A \times X \tag{26}$$

For which the we get the M-type for stream



## Bisimulation for Streams

We can define an equivalence relation

$$\frac{\text{hd } s \equiv \text{hd } t \quad \text{tl } s \sim_{\text{stream}} \text{tl } t}{s \sim_{\text{stream}} t}$$
 (27)

We can formalize this as a bisimulation. A (strong) bisimulation for a P-coalgebra C- $\gamma$  is given by

- a relation  $\mathcal{R}: C \to C \to \mathcal{U}$
- a type  $\overline{\mathcal{R}} = \sum_{(a:C)} \sum_{(b:C)} a \mathcal{R} b$
- and a function  $\alpha_{\mathcal{R}}:\overline{\mathcal{R}}\to P_S\,\overline{\mathcal{R}}$

Such that  $\overline{\mathcal{R}}$ - $\alpha_{\mathcal{R}}$  is a P-coalg, making the following diagram commute.

$$C-\gamma \xleftarrow{\pi_1^{\overline{R}}} \overline{R} - \alpha_R \xrightarrow{\pi_2^{\overline{R}}} C-\gamma$$

In MLTT this is not enough to define an equality, however in HoTT and CTT it is.

# Coinduction Principle

## Theorem (Coinduction principle)

Given a relation  $\mathcal{R}$ , that is a bisimulation for a M-type, then (strongly) bisimilar elements  $x \mathcal{R} y$  are equal  $x \equiv y$ .

# Coinduction Principle

## Theorem (Coinduction principle)

Given a relation  $\mathcal{R}$ , that is a bisimulation for a M-type, then (strongly) bisimilar elements  $x \mathcal{R} y$  are equal  $x \equiv y$ .

#### Proof.

We get the diagram

$$\operatorname{M-out} \xleftarrow{\pi_1^{\overline{\mathcal{R}}}} \overline{\mathcal{R}} - \alpha_{\mathcal{R}} \xrightarrow{\pi_2^{\overline{\mathcal{R}}}} \operatorname{M-out}$$

Since M-out is a final coalgebra, functions into it are unique, meaning

$$\pi_1^{\overline{R}} \equiv \pi_2^{\overline{R}} \tag{28}$$

therefore given  $r: x \mathcal{R} y$ , we can construct the equality

$$x \equiv \pi_1^{\overline{R}}(x, y, r) \equiv \pi_2^{\overline{R}}(x, y, r) \equiv y.$$
 (29)



## Overview

- - Goals
- - Definition of M-types
  - Construction of M-types and Examples
  - Equality for Coinductive types
- Quotient M-types
  - Propositional Truncation and Set Truncated Quotient

## Propositional Truncation and Set Truncated Quotient

A Higher Inductive Type (HIT) is a type defined by point constructors as well as equality constructors.

## Propositional Truncation and Set Truncated Quotient

A Higher Inductive Type (HIT) is a type defined by point constructors as well as equality constructors.

# Propositional Truncation and Set Truncated Quotient

A Higher Inductive Type (HIT) is a type defined by point constructors as well as equality constructors. We can define set truncated quotients as the following HIT.

## Definition (Set Truncated Quotient)

$$\frac{x:A}{[x]:A/\mathcal{R}} \quad (30) \qquad \frac{x,y:A \quad r:x \mathcal{R} \ y}{\operatorname{eq}/x \ y \ r:[x] \equiv [y]} \quad (31)$$

$$\overline{\text{squash}/:\text{isSet}(A/R)}$$

(32)

# Partiality monad

QM-type

The partiality monad describes equality of partial computations.

23 / 32

The partiality monad describes equality of partial computations.

We define the partiality monad by quotienting the delay monad by a relation defined by the constructors

$$\frac{x \sim y}{ {\tt later} \; x \sim y} \sim_{ {\tt later}_1 }$$
 (33)  $\frac{x \sim y}{x \sim {\tt later} \; y} \sim_{ {\tt later}_r }$  (34)

$$\frac{a \equiv b}{\text{now } a \sim \text{now } b} \sim_{\text{now}} \text{ (35)} \qquad \frac{x \sim y}{\text{later } x \sim \text{later } y} \sim_{\text{later}} \text{ (36)}$$

the partiality monad is then given by the set truncated quotient

Delay 
$$R/\sim$$
 (37)

however we need the axiom of (countable) choice

# Partiality Monad

A QIIT is a type that is defined at the same time as a relation and then set truncated.

## Partiality Monad QIIT

A QIIT is a type that is defined at the same time as a relation and then set truncated.

We can define the partiality monad as a QIIT with type constructors

$$\frac{a:R}{R_{\perp}:\mathcal{U}} \qquad (38) \qquad \frac{a:R}{\eta \ a:R_{\perp}} \qquad (40)$$

and an ordering relation  $(\cdot \sqsubseteq_{\perp} \cdot)$  indexed twice over  $R_{\perp}$ 

$$\frac{x : \mathbb{R}_{\perp}}{x \sqsubseteq_{\perp} x} \sqsubseteq_{\mathsf{refl}} (41) \qquad \frac{x \sqsubseteq_{\perp} y \quad y \sqsubseteq_{\perp} z}{x \sqsubseteq_{\perp} z} \sqsubseteq_{\mathsf{trans}} (42)$$

$$\frac{x, y : R_{\perp} \quad p : x \sqsubseteq_{\perp} y \quad q : y \sqsubseteq_{\perp} x}{\alpha_{\perp} \quad p \quad q : x \equiv y}$$
(43)

where  $\perp$  is the smallest element in the order

$$\frac{x:R_{\perp}}{\perp \Box \perp x} \sqsubseteq_{\text{never}} \tag{44}$$

# Partiality Monad

with an upper bound

$$\frac{\mathsf{s}: \mathbb{N} \to R_{\perp} \quad \mathsf{b}: \prod_{(n:\mathbb{N})} \mathsf{s}_n \sqsubseteq_{\perp} \mathsf{s}_{n+1}}{\bigsqcup(\mathsf{s},\mathsf{b}): R_{\perp}} \tag{45}$$

which gives a bound for a sequence

$$\frac{\mathbf{s}: \mathbb{N} \to \mathbf{R}_{\perp} \quad \mathbf{b}: \prod_{(n:\mathbb{N})} \mathbf{s}_n \sqsubseteq_{\perp} \mathbf{s}_{n+1}}{\prod_{(n:\mathbb{N})} \mathbf{s}_n \sqsubseteq_{\perp} \bigsqcup(\mathbf{s}, \mathbf{b})}$$
(46)

and which is a least upper bound

$$\frac{\prod_{(n:\mathbb{N})} \mathsf{s}_n \sqsubseteq_{\perp} x}{\bigsqcup(\mathsf{s},\mathsf{b}) \sqsubseteq_{\perp} x} \tag{47}$$

and finally set truncated by the constructor  $(-)_{\perp}$ -isSet

To show these two definitions are equal we

• Define an intermediate type of sequences

$$\operatorname{Seq} A = \sum_{(\mathbf{s}: \mathbb{N} \to A + \mathbf{1})} \prod_{(n:\mathbb{N})} \mathbf{s}_n \sqsubseteq_{\mathbf{R} + \mathbf{1}} \mathbf{s}_{n+1}$$
 (48)

- Show that the Delay monad is equal to this intermediate type
- Define weak bisimilarty for sequences, and show it respects the equality to the Delay monad
- ullet Show that  $\mathrm{Seq}_R/\sim$  is equal to the partiality monad, by
  - ullet Defining a function from  ${\sf Seq}_R/\!\!\sim$  to  $R_\perp$
  - Show this function is injective and surjective

However the proof for surjectivity requires the axiom of countable choice!

# Simple Partialty Monad QIIT

This construction is quite involved, however doing a trivial construction

$$\frac{a:R}{\text{now }a:R_{\perp}} \quad \text{(49)} \qquad \frac{t:R_{\perp}}{\text{later }t:R_{\perp}} \qquad \text{(50)}$$

$$\frac{x\equiv y}{\text{later }x\equiv y} \text{ later}_{\equiv} \qquad \text{(51)}$$

would not give us the intended functionality, so we need to think more about how we define the QIIT.

We have to think about how we define a QM-type.

# What do we want from a quotient M-type?

- We would like to be able to construct a quotient from an M-type and a relation.
- We should be able to lift constructors to the quotient without the axiom of choice.
- The type should be equal to the type defined by the set truncated quotient if we assume the axiom of choice.

# Alternative: Quotient Polynomial Functor (QPF)

We can define quotiented M-types from a quotient polynomial functor.

## Definition (Quotient Polynomial Functor)

We define a quotient polynomial functor (QPF), for types as

$$FX = \sum_{(a:A)} ((B \ a \to X)/\sim_a)$$
 (52)

and for a function  $f: X \to Y$ , we use the quotient eliminator with

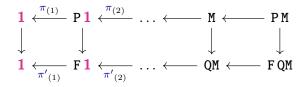
$$P = \lambda_{\underline{\phantom{A}}}, (B a \to \underline{\phantom{A}})/\sim_a$$
 (53)

for which we need  $\sim_{\sf ap}$ , which says that given a function  ${\bf f}$  and  ${\bf x}\sim_a {\bf y}$  then  ${\bf f}\circ {\bf x}\sim_a {\bf f}\circ {\bf y}$ .

$$Ff(a,g) = (a, elim g)$$
 (54)

# Alternative: Quotient Polynomial Functor (QPF)

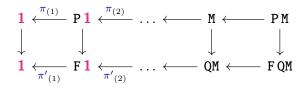
We get the diagram



For which  $\mathtt{M} \equiv \mathtt{P}\,\mathtt{M}$  and we would hope  $\mathtt{QM} \equiv \mathtt{F}\,\mathtt{QM}$ , however this requires the axiom of choice.

# Alternative: Quotient Polynomial Functor (QPF)

We get the diagram



For which  ${\tt M} \equiv {\tt P} \, {\tt M}$  and we would hope  ${\tt QM} \equiv {\tt F} \, {\tt QM}$ , however this requires the axiom of choice.

However looking further into this is future research.

## Conclusion

#### We have

- given a formalization/semantic of M-types
- shown examples of and rules for how to construct M-types
- given a coinduction principle for M-types
- described the construction of the partiality monad as a QIIT
- discussed ways of constructing quotient M-types

#### Contribution

- Formalization in Cubical Agda
- Introducing QM-types

## Future Work

- Indexed M-types
- Showing finality of QM-types and fully formalizing the constructions
- Equality between Coinductive records and M-types
- Explore Guarded Cubical Type Theory