(Q)M-types and Coinduction in HoTT / CTT

Master's Thesis, Computer Science

Lasse Letager Hansen, 201508114

Supervisor: Bas Spitters

Aarhus University

June 20, 2020

AARHUS
UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Overview

We can define the natural numbers *inductively* from the two constructors

Definition (The Natural Numbers)

$$\frac{}{0 : \mathbb{N}} \quad (1) \qquad \frac{n : \mathbb{N}}{\mathtt{succ}\ n : \mathbb{N}} \quad (2)$$

for which we define an *inductive* equivalence relation

Definition (Equivalence for the Natural Number)

$$\frac{}{0 = 0}\ \sim_{\mathtt{0}} \quad (3) \qquad \frac{n \sim_{\mathbb{N}} m}{\mathtt{succ}\ n \sim_{\mathbb{N}} \mathtt{succ}\ m}\ \sim_{\mathtt{succ}} \quad (4)$$

which is also an equality relation.

We can define streams coinductively from the two constructors

Definition (Stream)

$$\frac{s : \mathtt{stream}\ A}{\mathtt{hd}\ s : A} \quad (5) \qquad \frac{s : \mathtt{stream}\ A}{\mathtt{tl}\ s : \mathtt{stream}\ A} \quad (6)$$

for which we can coinductively define an equivalence relation

Definition (Equivalence for Streams)

$$\frac{\mathtt{hd}\ s = \mathtt{hd}\ t \quad \mathtt{tl}\ s \sim_{\mathtt{stream}} \mathtt{tl}\ t}{s \sim_{\mathtt{stream}} t} \quad (7)$$

however, this bisimulation does not give us equality.

The univalence axiom says that equivalence is equivalent to equality

Definition (Univalence)

$$(A = B) \simeq (A \simeq B) \tag{8}$$

so if we work in a type theory where this holds, then $\sim_{\texttt{stream}}$ becomes an equality relation for streams.

Homotopy Type Theory (HoTT)

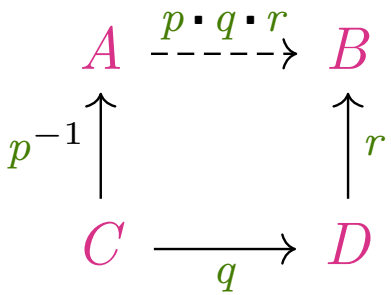
Homotopy Type Theory (HoTT)

- is an intensional dependent type theory (builts on MLTT)
- assumes the univalence axiom
- has higher inductive types (HITs)

Types can be seen as spaces and elements as points of a space, meaning we get non-trivial equalities, that is equalities that are not reflexivity. As such we can construct higher inductive types, where we have both point and equality constructors.

Cubical Type Theory (CTT)

HoTT is constructive, however the univalence axiom is not, since it is an axiom. We therefore use a Cubical Type Theory, where we can prove the univalence axiom and get constructivity. Cubical Type Theory is named after the composition principle, where given all but one side of a square, we get the entire square.



This work has been formalized using the Cubical Agda proof assistant.

H-Levels

HoTT is proof relevant, so we can have two proofs of a statement that might not be equal. Types in HoTT is classified in H-levels starting at -2.

(-2) Contractible types, with an element equal to all other elements

(-1) Mere propositions / hProp, all elements in a type are equal, but the type might not be inhabited.

(0) hSets, given two elements, then all equalities between the two elements are equal.

$\vdots$

(n) all equalities at H-level $(n+1)$ are equal.

Propositional Truncation

We can truncate types to H-level (-1) with propositional truncation, defined as the following HIT

Definition (Propositional Truncation)

$$\frac{x : A}{|\, x \,| : \|A\|} \quad (9) \qquad \frac{x, y : \|A\|}{\mathtt{squash}\ x\ y : x \equiv y} \quad (10)$$

which removes the proof relevance, since when truncated all proofs of a statement only states that the type is inhabited.

Set Truncated Quotients

We can define quotients as by the following HIT.

Definition (Set truncated quotient)

$$\frac{x : A}{[\,x\,] : A/\mathcal{R}} \quad (11) \qquad \frac{x, y : A/\mathcal{R} \quad r : x\ \mathcal{R}\ y}{\mathtt{eq}/\ x\ y\ r : x \equiv y} \quad (12)$$

$$\frac{}{\mathtt{squash}/ : \mathtt{isSet}\ (A/\mathrm{R})} \quad (13)$$

Axiom of Choice

The axiom of choice in HoTT

Definition (Axiom of Choice)

$$\prod_{(x:X)} \|\mathtt{Y}\ x\| \to \left\| \prod_{(x:X)} \mathtt{Y}\ x \right\| \tag{14}$$

and the axiom of countable choice

Definition (Axiom of countable choice)

$$\prod_{(n:\mathbb{N})} \|\mathtt{Y}\ n\| \to \left\| \prod_{(n:\mathbb{N})} \mathtt{Y}\ n \right\| \tag{15}$$

Overview

Containers and Polynomial functors

Definition

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

Definition

A polynomial functor $\mathtt{P}_S$ (or extension) for a container $S = (A\,, \mathtt{B})$ is defined, for types as

$$\begin{aligned} \mathtt{P}_S &: \mathcal{U} \to \mathcal{U} \\ \mathtt{P}_S\, X &= \sum_{(a:A)} \mathtt{B}a \to X \end{aligned} \tag{16}$$

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

$$\begin{aligned} \mathtt{P}_S\, \mathtt{f} &: \mathtt{P}_S X \to \mathtt{P}_S\, Y \\ \mathtt{P}_S\, \mathtt{f}\ (a, \mathtt{g}) &= (a, \mathtt{f} \circ \mathtt{g}). \end{aligned} \tag{17}$$

Coalgebra

Definition

A $\mathtt{P}_S$-coalgebra is defined as

$$\mathtt{Coalg}_S = \sum_{(C:\mathcal{U})} C \to \mathtt{P}_S\, C. \tag{18}$$

where we denote a $\mathtt{P}_S$-coalgebra given by a type C and function γ as C-γ. Coalgebras morphisms are defined as

$$C\text{-}\gamma \Rightarrow D\text{-}\delta = \sum_{(\mathtt{f}:C\to D)} \delta \circ \mathtt{f} = \mathtt{P}\mathtt{f} \circ \gamma \tag{19}$$

Final Coalgebra

Definition (Final Coalgebra / M-type)

A final coalgebra $X\text{-}\rho$, is a coalgebra that fulfills

$$\texttt{Final}_S := \sum_{(X\text{-}\rho:\texttt{Coalg}_S)} \prod_{(C\text{-}\gamma:\texttt{Coalg}_S)} \texttt{isContr}\,(C\text{-}\gamma \Rightarrow X\text{-}\rho). \tag{20}$$

We define M-types as the coinductive type that fulfill the property

$$\prod_{(C\text{-}\gamma:\texttt{Coalg}_S)} \texttt{isContr}\,(C\text{-}\gamma \Rightarrow \texttt{M}_S\text{-}\texttt{out}) \tag{21}$$

We denote M-types as $\texttt{M}_{(A\,,\,\texttt{B})}$, $\texttt{M}_S$ or just M when the container is clear from the context.

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.

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

Definition

The limit of a chain is given as

$$\mathcal{L} = \sum_{(x:\prod_{(n:\mathbb{N})} X_n)} \prod_{(n:\mathbb{N})} (\pi_{(n)}\ x_{n+1} \equiv x_n) \tag{22}$$

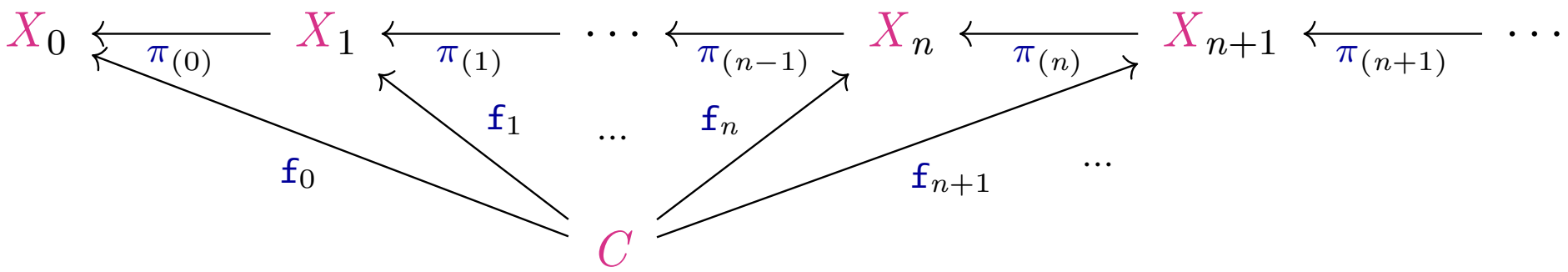
Helper Lemmas

Lemma (Limit Collapse)

Given $\ell : \prod_{(n:\mathbb{N})}(X_n \to X_{n+1})$ *and* $y : \sum_{(x:\prod_{(n:\mathbb{N})} X_n)} x_{n+1} \equiv \ell_n\ x_n$ *we get the equality* $\mathcal{L} \equiv X_0$.

Lemma (Cone and function to M-type equality)

For all coalgebras C*-*γ *defined over the container* S*, we get* $C \to \mathtt{M}_S \equiv \mathsf{Cone}_{C,\gamma}$*, where*
$\mathsf{Cone} = \sum_{(\mathtt{f}:\prod_{(n:\mathbb{N})} C \to X_n)} \prod_{(n:\mathbb{N})} \pi_{(n)} \circ \mathtt{f}_{n+1} \equiv \mathtt{f}_n$. *(See figure)*



Equality between $\mathcal{L}$ and $\mathrm{P}\,\mathcal{L}$

Theorem

Given a container $(A\,, \mathrm{B})$*, we define a chain as the repeated application of* P *to the unit element* $X_n = \mathrm{P}^n\,\mathbf{1}$*, and* $\pi_{(n)} = \mathrm{P}^n\,!$ *where* $! : A \to \mathbf{1}$ *is the unique function into the unit type. Then there is an equality*

$$shift : \mathcal{L} \equiv \mathrm{P}\mathcal{L} \tag{30}$$

where $\mathcal{L}$ *is the limit of this chain.*

Proof structure

The proof is done using the two helper lemmas

$$\alpha : \mathcal{L}^{\mathrm{P}} \equiv \mathrm{P}\,\mathcal{L} \tag{31}$$

$$\mathcal{L}unique : \mathcal{L} \equiv \mathcal{L}^{\mathrm{P}} \tag{32}$$

where $\mathcal{L}^{\mathrm{P}}$ 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 $shift = \alpha \cdot \mathcal{L}unique$.

Proof of uniqueness

$\mathcal{L}unique$ says the limit of a chain is unique.

Proof.

The proof of the equality $\mathcal{L}unique$ is given by the isomorphism

$$\mathtt{fun}_{\mathcal{L}unique}\ (\mathtt{a},\mathtt{b}) = \rangle\!\rangle\ \star\ ,\ \mathtt{a}\ \langle\!|, \rangle\!\rangle\ \mathtt{refl}_{\star}\ ,\ \mathtt{b}\ \langle\!| \tag{33}$$

$$\mathtt{inv}_{\mathcal{L}unique}\ (\mathtt{a},\mathtt{b}) = \mathtt{a} \circ \mathtt{succ}\ ,\ \mathtt{b} \circ \mathtt{succ} \tag{34}$$

$$\mathtt{rinv}_{\mathcal{L}unique}\ (\mathtt{a},\mathtt{b}) = \mathtt{refl}_{(\mathtt{a},\mathtt{b})} \tag{35}$$

$$\mathtt{linv}_{\mathcal{L}unique}\ (\mathtt{a},\mathtt{b}) = \mathtt{refl}_{(\mathtt{a},\mathtt{b})} \tag{36}$$

□

Proof of equality to shifted chain

Proof.

The proof of α is given by the equalities

$$\mathcal{L}^{\mathsf{P}} \equiv \sum_{(x:\prod_{(n:\mathbb{N})} X_{n+1})} \prod_{(n:\mathbb{N})} \pi_{(n+1)}\ x_{n+1} \equiv x_n \tag{37}$$

$$\equiv \sum_{(x:\prod_{(n:\mathbb{N})} \sum_{(a:A)} \mathsf{B}\, a \to X_n)} \prod_{(n:\mathbb{N})} \pi_{(n+1)}\ x_{n+1} \equiv x_n \tag{38}$$

$$\equiv \sum_{(\sum_{(a:\prod_{(n:\mathbb{N})} A)} \prod_{(n:\mathbb{N})} a_{n+1} \equiv a_n)} \sum_{(\mathbf{u}:\prod_{(n:\mathbb{N})} \mathsf{B}\, a_n \to X_n)} \prod_{(n:\mathbb{N})} \pi_{(n)} \circ \mathbf{u}_{n+1} \equiv_* \mathbf{u}_n \tag{39}$$

$$\equiv \sum_{(a:A)} \sum_{(\mathbf{u}:\prod_{(n:\mathbb{N})} \mathsf{B}\, a \to X_n)} \prod_{(n:\mathbb{N})} \pi_{(n)} \circ \mathbf{u}_{n+1} \equiv \mathbf{u}_n \tag{40}$$

$$\equiv \sum_{(a:A)} \mathsf{B}\, a \to \mathcal{L} \qquad \equiv \mathsf{P}\mathcal{L} \tag{41}$$

□

In and Out of M-types

We define functions to construct and destruct M-types

Definition

For the equality $shift$ we denote the functions back and forth as

$$\mathtt{out} = \mathtt{transport}\ shift \tag{42}$$

$$\mathtt{in} = \mathtt{transport}\ shift^{-1}. \tag{43}$$

This gives us the following diagram

$$\begin{array}{c} \mathtt{P}\,\mathtt{M}_S \\ \mathtt{out} \uparrow \downarrow \mathtt{in} \\ \mathtt{M}_S \end{array}$$

Example: Streams

We can now define streams for a given type A

Definition

We start with a container

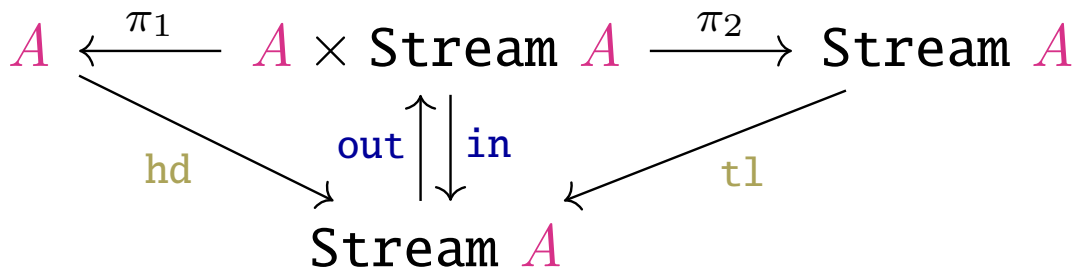
$$(A \,, \mathbf{1}) \tag{44}$$

For which we get the polynomial functor

$$\mathtt{P}\, X = A \times X \tag{45}$$

For which the we get the M-type for stream

$$\mathtt{Stream}\ A = A \times \mathtt{Stream}\ A \tag{46}$$



M-types are final

Theorem

The M-type $\mathtt{M}_S$ *is defined as the limit for a polynomial functor* $\mathtt{P}_S$*. This definition fulfills the requirement that* $\mathtt{Final}_S\ \mathcal{L}$*.*

Proof.

By unfolding the definition, we need to show

$$\prod_{(C\text{-}\gamma:\mathtt{Coalg}_S)} \mathtt{isContr}\ (C\text{-}\gamma \Rightarrow \mathcal{L}\text{-}\mathtt{out}) \tag{47}$$

We do this by showing $(C\text{-}\gamma \Rightarrow \mathcal{L}\text{-}\mathtt{out}) \equiv \mathbf{1}$. We get

$$\equiv \sum_{(\mathtt{f}:C\to\mathcal{L})} (\mathtt{out} \circ \mathtt{f} \equiv \mathtt{P}\mathtt{f} \circ \gamma) \tag{48}$$

$$\equiv \sum_{(\mathtt{f}:C\to\mathcal{L})} (\mathtt{in} \circ \mathtt{out} \circ \mathtt{f} \equiv \mathtt{in} \circ \mathtt{P}\mathtt{f} \circ \gamma) \tag{49}$$

$$\equiv \sum_{(\mathtt{f}:C\to\mathcal{L})} (\mathtt{f} \equiv \mathtt{in} \circ \mathtt{P}\mathtt{f} \circ \gamma) \tag{50}$$

Proof. (cont.)

We now let $\psi = \mathtt{in} \circ \mathtt{Pf} \circ \gamma$, which simplifies the expression to

$$\sum_{(\mathtt{f}:C \to \mathcal{L})} (\mathtt{f} \equiv \psi\ \mathtt{f}) \tag{51}$$

We then define $\mathtt{e}$ to be the function from right to left for the equality $C \to \mathcal{L} \equiv \mathsf{Cone}_{C,\gamma}$, this gives us the equality

$$\sum_{(\mathtt{f}:C \to \mathcal{L})} (\mathtt{f} \equiv \psi\ \mathtt{f}) \quad \equiv \quad \sum_{(c:\mathsf{Cone}_{C,\gamma})} (\mathtt{e}\ c \equiv \psi\ (\mathtt{e}\ c)) \tag{52}$$

By defining a function $\phi\ (\mathtt{u}, \mathtt{g}) = (\phi_0\ \mathtt{u}, \phi_1\ \mathtt{u}\ \mathtt{g})$ where

$$\phi_0\ \mathtt{u} = \langle\!|\ (\lambda \_, \star)\ ,\ \mathtt{Pf} \circ \gamma \circ \mathtt{u}\ |\!\rangle \tag{53}$$

$$\phi_1\ \mathtt{u}\ \mathtt{g} = \langle\!|\ \mathtt{funExt}\ (\lambda \_, \mathtt{refl}_\star)\ ,\ \mathtt{ap}\ (\mathtt{Pf} \circ \gamma) \circ \mathtt{g}\ |\!\rangle \tag{54}$$

we get the commuting square

$$\begin{array}{ccc} \mathsf{Cone}_{C,\gamma} & \xrightarrow{\mathtt{e}} & (C \to \mathcal{L}) \\ \downarrow{\scriptstyle \phi} & & \downarrow{\scriptstyle \psi} \\ \mathsf{Cone}_{C,\gamma} & \xrightarrow{\mathtt{e}} & (C \to \mathcal{L}) \end{array}$$

Proof. (cont.)

We can then simplify to

$$\sum_{(c:\mathsf{Cone}_{C\text{-}\gamma})} (\mathsf{e}\ c \equiv \mathsf{e}\ (\phi\ c)) \tag{55}$$

We unfolding the definition of ϕ, to get the equalities

$$\sum_{(c:\mathsf{Cone}_{C\text{-}\gamma})} (c \equiv \phi\ c) \tag{56}$$

$$\equiv \sum_{((\mathsf{u},\mathsf{g}):\mathsf{Cone}_{C\text{-}\gamma})} ((\mathsf{u},\mathsf{g}) \equiv (\phi_0\ \mathsf{u}, \phi_1\ \mathsf{u}\ \mathsf{g})) \tag{57}$$

$$\equiv \sum_{((\mathsf{u},\mathsf{g}):\mathsf{Cone}_{C\text{-}\gamma})} \sum_{(p:\mathsf{u}\equiv\phi_0\,\mathsf{u})} \mathsf{g} \equiv_* \phi_1\ \mathsf{u}\ \mathsf{g} \tag{58}$$

We rearrange the terms and unfold the definition of Cone to get

$$\sum_{\left((\mathsf{u},p):\sum_{(\mathsf{u}:\prod_{(n:\mathbb{N})} C\to X_n)} \mathsf{u}\equiv\phi_0\,\mathsf{u}\right)} \sum_{(\mathsf{g}:\prod_{(n:\mathbb{N})} \pi_{(n)}\circ\mathsf{u}_{n+1}\equiv\mathsf{u}_n)} \mathsf{g} \equiv_* \phi_1\ \mathsf{u}\ \mathsf{g} \tag{59}$$

Proof. (cont.)

We define the following two equalities as instances of the limit collapse equality $\mathcal{L} \equiv X_0$.

$$\mathbf{1} \equiv \left(\sum_{(\mathbf{u}:\prod_{(n:\mathbb{N})} C \to X_n)} \mathbf{u} \equiv \phi_0\, \mathbf{u} \right) \tag{60}$$

$$\mathbf{1} \equiv_* \left(\sum_{(\mathbf{g}:\prod_{(n:\mathbb{N})} \pi_{(n)} \circ \mathbf{u}_{n+1} \equiv \mathbf{u}_n)} \mathbf{g} \equiv_* \phi_1\ \mathbf{u}\ \mathbf{g} \right) \tag{61}$$

Using these two equalities, the proof simplifies to

$$\sum_{(\star:\mathbf{1})} \mathbf{1} \equiv \mathbf{1} \tag{62}$$

which is trivial. □

Bisimulation

Definition

A relation $\mathcal{R} : C \to C \to \mathcal{U}$ for a coalgebra $C\text{-}\gamma : \mathtt{Coalg}_S$, is a (strong) bisimulation relation if the type $\overline{\mathcal{R}} = \sum_{(a:C)} \sum_{(b:C)} a\ \mathcal{R}\ b$ and the function $\alpha_{\mathcal{R}} : \overline{\mathcal{R}} \to \mathtt{P}_S\, \overline{\mathcal{R}}$ forms a $\mathtt{P}_S$-coalgebra $\overline{\mathcal{R}}\text{-}\alpha_{\mathcal{R}} : \mathtt{Coalg}_S$, making the diagram bellow commute ($\Longrightarrow$ represents $\mathtt{P}_S$-coalgebra morphisms). That is

$$\gamma \circ \pi_1^{\overline{\mathcal{R}}} \equiv \mathtt{P}_S\, \pi_1^{\overline{\mathcal{R}}} \circ \alpha_{\mathrm{R}} \tag{63}$$

and similarly for $\pi_2^{\overline{\mathcal{R}}}$.

$$C\text{-}\gamma \xleftarrow{\pi_1{}^{\overline{\mathcal{R}}}} \overline{\mathcal{R}}\text{-}\alpha_{\mathcal{R}} \xrightarrow{\pi_2{}^{\overline{\mathcal{R}}}} C\text{-}\gamma$$

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.

Given a relation $\mathcal{R}$ that is bisimulation relation over a final P-coalgebra $\mathtt{M\text{-}out} : \mathtt{Coalg}_S$ we get the diagram

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

By the finality of M-out, we get a function $!$ from M to $\overline{R}$, which is unique, meaning $\pi_1^{\overline{\mathcal{R}}} \equiv\ ! \equiv \pi_2^{\overline{\mathcal{R}}}$. Now given $r : x\ \mathcal{R}\ y$, we can construct the equality

$$x \equiv \pi_1^{\overline{\mathcal{R}}}(x, y, r) \equiv \pi_2^{\overline{\mathcal{R}}}(x, y, r) \equiv y, \tag{64}$$

giving us the coinduction principle for M-types. □

Example: Delay Monad

We define a container

$$(R + \mathbf{1}\,, [\mathbf{0}, \mathbf{1}]) \tag{65}$$

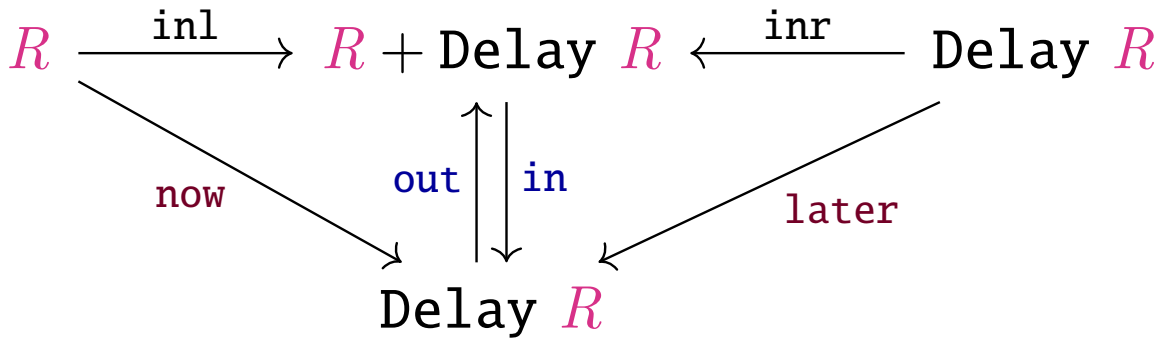
and a polynomial functor

$$\mathtt{P}\, X = \sum_{(x : R + \mathbf{1})} \begin{cases} \mathbf{0} & x = \mathtt{inl}\ r \\ \mathbf{1} & x = \mathtt{inr}\ \star \end{cases} \to X, \tag{66}$$

which simplifies to

$$\mathtt{P}_S\, X = R + X \tag{67}$$

such that we get the diagram



Example: Event Trees

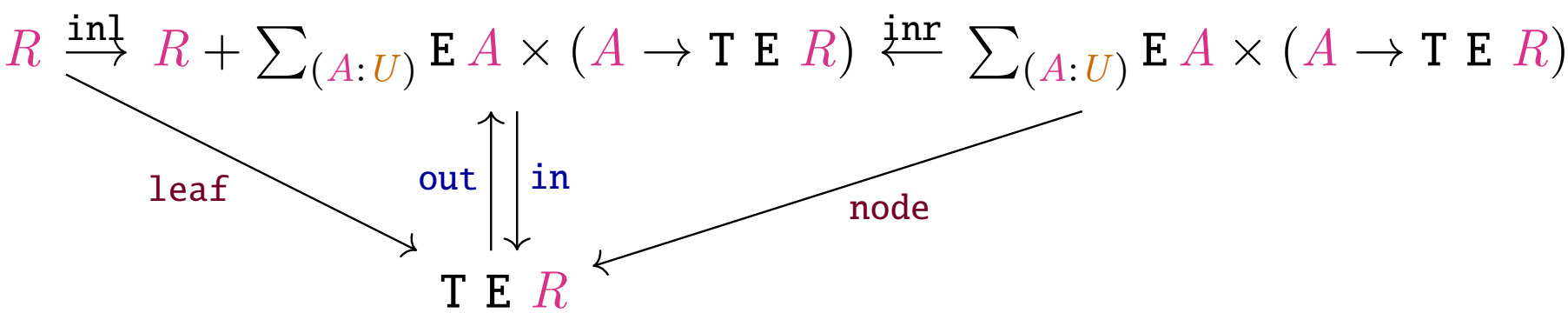
Definition

We define a container

$$\left(R + \sum_{(A:\mathcal{U})} \mathtt{E}\, A \,,\, [\mathbf{0}, A] \right) . \tag{68}$$

and a polynomial functor which simplifies to

$$\mathtt{P}\, X = R + \sum_{(A:\mathcal{U})} \mathtt{E}\, A \times (A \to X). \tag{69}$$



Example: Interaction Trees (ITrees)

$$\frac{r : R}{\texttt{Ret}\ r : \texttt{itree}\ \texttt{E}\ R}\ \texttt{I}_{\texttt{Ret}} \tag{70}$$

$$\frac{A : \mathcal{U} \quad a : \texttt{E}\ A \quad f : A \to \texttt{itree}\ \texttt{E}\ R}{\texttt{Vis}\ a\ f : \texttt{itree}\ \texttt{E}\ R}\ \texttt{I}_{\texttt{Vis}}. \tag{71}$$

$$\frac{t : \texttt{itree}\ \texttt{E}\ R}{\texttt{Tau}\ t : \texttt{itree}\ \texttt{E}\ R}\ \texttt{E}_{\texttt{Tau}}. \tag{72}$$

Definition

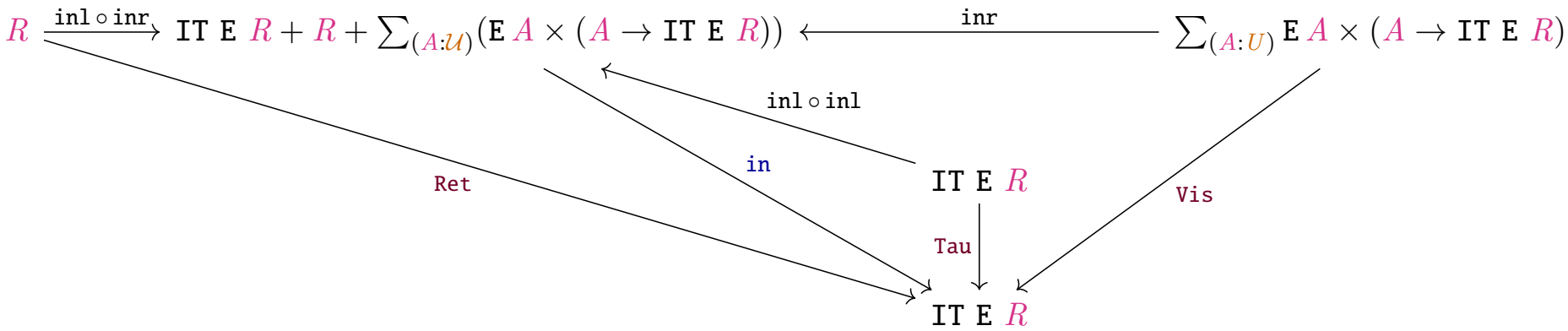
We define a container

$$\left(R + \mathbf{1} + \sum_{(A:\mathcal{U})} (\texttt{E}\ A)\ , [\mathbf{0}, \mathbf{1}, A] \right). \tag{73}$$

and a polynomial functor which simplifies to

$$\texttt{P}\ X = R + X + \sum_{(A:\mathcal{U})} (\texttt{E}\ A \times (A \to X)) \tag{74}$$

Example: Interaction Trees (ITrees)



Rules for Constructing M-types

Adding containers $(A \,, \mathtt{B})$ and $(C \,, \mathtt{D})$ for two constructors together is done by

$$\left(A + C \,, \begin{cases} \mathtt{B}\, a & \mathtt{inl}\ a \\ \mathtt{D}\, c & \mathtt{inr}\ c \end{cases} \right) \tag{75}$$

whereas adding containers for two destructors is done by

$$(A \times C \,, \lambda \_, \, \lambda\, (a, c), \ \mathtt{B}\, a + \mathtt{D}\, c) \tag{76}$$

However combining both destructors and constructors is not as simple, which is also the case if we use records or other constructions to define such types.

Rules for Constructing M-types

Given a record, which is a list of fields $f_1 : F_1, f_2 : F_2, \ldots, f_n : F_n$, we can construct the M-types by the container

$$(F_1 \times F_2 \times \cdots \times F_n \,, \lambda \_, \mathbf{0}) \tag{77}$$

where each destructor $\mathrm{d_n} : T \to F_n$ for the field f_n will be defined as $\mathrm{d_n}\ t = \pi_n\ (\mathtt{out}\ t)$. However fields in a coinductive record may depend on previous defined fields.

$$\left(\sum_{(f_1 : F_1)} \sum_{(f_2 : \mathbf{F}_2)} \cdots \sum_{(f_{n-1} : \mathbf{F}_{n-1})} \mathbf{F}_n \,, \lambda \_, \mathbf{0} \right) \tag{78}$$

even in a dependent way

$$\left(\sum_{(f_1 : \mathcal{U})} \left(\mathbf{1} \times \sum_{(f_3 : \mathbf{F}_3)} \cdots \sum_{(f_{n-1} : \mathbf{F}_{n-1})} \mathbf{F}_n \right) \,, \lambda\, (f_1, \star, f_3, \ldots), \mathbf{F}_2 \right) \tag{79}$$

Rules for Constructing $\mathtt{M}$-types

Another case is that the type of a field is dependent, but by currying we get

$$\left(\sum_{(f_1 : \mathcal{U})} \sum_{(f_3 : \mathtt{F}_3)} \cdots \sum_{(f_{n-1} : \mathtt{F}_{n-1})} \mathtt{F}_n \, , \lambda \, (f_1, f_3, \dots), \sum_{(x : f_1)} (\mathtt{B} \, x) \right) \tag{80}$$

so we expect that a type defined as a (coinductive) record is equal to the version defined as a $\mathtt{M}$-type. However, this has not been shown formally.

Overview

QM-types

We will start by looking at two ways of quotienting M-types.
Set truncated quotients and quotient inductive-inductive type (QIIT). A QIIT is a type that is defined at the same time as a relation over that type and then set truncated.

Multiset

We construct the type of multisets as a QM-type and as a QIIT and show these are equal. The base type is given by

Definition

We define A-valued $\mathbb{N}$-branching trees $\mathtt{T}\,A$ as the M-type defined by the container

$$(A + \mathbf{1}\,, [\mathbf{0}, \mathbb{N}]) \tag{81}$$

For which we get the constructors

$$\frac{a : A}{\mathtt{leaf}\ a : \mathtt{T}\,A} \tag{82}$$

$$\frac{\mathtt{f} : \mathbb{N} \to \mathtt{T}\,A}{\mathtt{node}\ \mathtt{f} : \mathtt{T}\,A} \tag{83}$$

The QIIT is defined by the point constructors `leaf` and `node`, and equality constructors

$$\frac{\mathtt{f} : \mathbb{N} \to \mathtt{T}\,A \quad \mathtt{g} : \mathbb{N} \to \mathbb{N} \quad \mathtt{isIso}\ \mathtt{g}}{\mathtt{node}\ \mathtt{f} \equiv \mathtt{node}\ (\mathtt{f} \circ \mathtt{g})}\ \mathtt{perm} \tag{84}$$

and `MS-isSet` which set truncates the type.

Multisets

We get the QM-type for multisets by quotienting the tree type by the relation defined by the constructors

$$\frac{x \equiv y}{\mathtt{leaf}\ x \sim_{\mathrm{T}} \mathtt{leaf}\ y}\ \sim_{\mathtt{leaf}} \tag{85}$$

$$\frac{\prod_{(n:\mathbb{N})} \mathtt{f}\ n \sim_{\mathrm{T}} \mathtt{g}\ n}{\mathtt{node}\ \mathtt{f} \sim_{\mathrm{T}} \mathtt{node}\ \mathtt{g}}\ \sim_{\mathtt{node}} \tag{86}$$

$$\frac{\mathtt{f} : \mathbb{N} \to \mathrm{T}\, A \quad \mathtt{g} : \mathbb{N} \to \mathbb{N} \quad \mathtt{isIso}\ \mathtt{g}}{\mathtt{node}\ \mathtt{f} \sim_{\mathrm{T}} \mathtt{node}\ (\mathtt{f} \circ \mathtt{g})}\ \sim_{\mathtt{perm}} \tag{87}$$

Multisets

We define a function between the two definitions

$$\begin{aligned} \mathtt{T{\to}MS}\ (\mathtt{leaf_T}\ x) &= \mathtt{leaf_{MS}}\ x \\ \mathtt{T{\to}MS}\ (\mathtt{node_T}\ \mathtt{f}) &= \mathtt{node_{MS}}\ (\mathtt{T{\to}MS} \circ \mathtt{f}) \end{aligned} \tag{88}$$

Lemma

If $x, y : \mathtt{T}\ A$ *are weakly bisimilar* $p : x \sim_{\mathtt{T}} y$ *then* $\mathtt{T{\to}MS}\ x \equiv \mathtt{T{\to}MS}\ y$.

Proof.

We do the proof by casing on the constructor for the weak bisimilarity.

$$\begin{aligned} &\mathtt{T{\to}MS\text{-}{\sim}{\to}{\equiv}}\ (\sim_{\mathtt{leaf}}\ p) = \mathtt{ap}\ \mathtt{leaf_{MS}}\ p \\ &\mathtt{T{\to}MS\text{-}{\sim}{\to}{\equiv}}\ (\sim_{\mathtt{node}}\ \mathtt{k}) = \mathtt{ap}\ \mathtt{node_{MS}}\ (\mathtt{funExt}\ (\mathtt{T{\to}MS\text{-}{\sim}{\to}{\equiv}} \circ \mathtt{k})) \\ &\mathtt{T{\to}MS\text{-}{\sim}{\to}{\equiv}}\ (\sim_{\mathtt{perm}}\ \mathtt{f}\ \mathtt{g}\ e) = \mathtt{perm}\ (\mathtt{T{\to}MS} \circ \mathtt{f})\ \mathtt{g}\ e \end{aligned} \tag{89}$$

□

Multisets

With this lemma, we can lift the function $\mathtt{T}{\to}\mathtt{MS}$ to the quotient, using the recursor for quotients. We now show that this function is injective and surjective. We start with a helper lemma.

Lemma

Given an equality $\mathtt{T}{\to}\mathtt{MS}\ x \equiv \mathtt{T}{\to}\mathtt{MS}\ y$ *then* $x \sim_{\mathtt{T}} y$.

Proof.

If x and y are leaves with values a and b then $a \equiv b$ by the injectivity of the constructor $\mathtt{leaf}_{\mathtt{MS}}$, giving us the wanted bisimilarity by $\sim_{\mathtt{leaf}}$. If x and y are nodes defined by functions $\mathtt{f}$ and $\mathtt{g}$, then by the injectivity of the constructor $\mathtt{node}_{\mathtt{MS}}$, we get $\prod_{(n:\mathbb{N})} \mathtt{f}\ n \equiv \mathtt{g}\ n$ and by induction we get $\prod_{(n:\mathbb{N})} \mathtt{f}\ n \sim_{\mathtt{T}} \mathtt{g}\ n$, making us able to use $\sim_{\mathtt{node}}$ to construct the bisimilarity. We do not get any other equalities, since $\mathtt{leaf}_{\mathtt{MS}}$ and $\mathtt{node}_{\mathtt{MS}}$ are disjoint. □

Multiset

Lemma

The function $\mathtt{T/{\sim}{\to}MS}$ *is injective, meaning* $\mathtt{T/{\sim}{\to}MS}\ x \equiv \mathtt{T/{\sim}{\to}MS}\ y$ *implies* $x \equiv y$.

Proof.

We show injectivity by doing propositional elimination, with

$$\mathrm{P} = (\lambda\, x,\ \mathtt{T/{\sim}{\to}MS}\ x \equiv \mathtt{T/{\sim}{\to}MS}\ y \to x \equiv y) \tag{90}$$

then

$$\mathtt{elimProp}\ (\lambda\, a,\ \mathtt{elimProp}(\lambda\, b,\ \mathtt{eq/}\ a\ b \circ \mathtt{T{\to}MS\text{-}{\equiv}{\to}{\sim}}\ a\ b)\ y)\ x \tag{91}$$

□

Multiset

Definition

We define the propositional eliminator for MS.

$$\begin{aligned}
&\mathtt{elimProp_{MS}}\ (\mathtt{leaf}\ x) = \mathtt{P_{leaf}}\ x \\
&\mathtt{elimProp_{MS}}\ (\mathtt{node}\ f) = \mathtt{P_{node}}\ (\mathtt{elimProp_{MS}} \circ f) \\
&\mathtt{elimProp_{MS}}\ (\mathtt{perm\ f\ g}\ e\ i) = \\
&\quad \mathtt{isProp{\to}isDepProp}\ \mathtt{P_{prop}} \\
&\qquad (\mathtt{elimProp_{MS}}\ (\mathtt{node\ f}))\ (\mathtt{elimProp_{MS}}\ (\mathtt{node}\ (\mathtt{f} \circ \mathtt{g}))) \\
&\qquad (\mathtt{perm\ f\ g}\ e)\ i \qquad (92) \\
&\mathtt{elimProp_{MS}}\ (\mathtt{MS\text{-}isSet}\ a\ b\ p\ q\ i\ j) = \\
&\quad \mathtt{isSet{\to}isDepSet}\ (\mathtt{isProp{\to}isSet} \circ \mathtt{P_{prop}}) \\
&\qquad (\mathtt{elimProp_{MS}}\ a)\ (\mathtt{elimProp_{MS}}\ b) \\
&\qquad (\mathtt{ap\ elimProp_{MS}}\ p)\ (\mathtt{ap\ elimProp_{MS}}\ q) \\
&\qquad (\mathtt{MS\text{-}isSet}\ a\ b\ p\ q)\ i\ j
\end{aligned}$$

Multiset

Lemma

The function $\mathtt{T}/{\sim}{\to}\mathtt{MS}$ *is surjective, assuming the axiom of countable choice. Being surjective means for all* $b : \mathtt{MS}\ X$ *we have* $\|\Sigma_{(x:\mathtt{T}\ X/{\sim_\mathtt{T}})}\ \mathtt{T}/{\sim}{\to}\mathtt{MS}\ x \equiv b\|$.

Proof.

We do propositional elimination of $\mathtt{MS}\ X$, with

$$\mathtt{P}\ y = \|\Sigma_{(x:\mathtt{T}\ X/{\sim_\mathtt{T}})}\ \mathtt{T}/{\sim}{\to}\mathtt{MS}\ x \equiv y\| \tag{93}$$

For the leaf case, we have the simple equality

$$\mathtt{T}/{\sim}{\to}\mathtt{MS}\ [\mathtt{leaf}_\mathtt{T}\ x/{\sim_\mathtt{T}}] \equiv \mathtt{leaf}_\mathtt{MS}\ x \tag{94}$$

For the node case with $\mathtt{node}\ \mathtt{f}$, we get the following function by induction

$$\mathtt{f}' : \prod_{(n:\mathbb{N})} \left\| \sum_{(y:\mathtt{T}\ X/{\sim_\mathtt{T}})} \mathtt{T}/{\sim}{\to}\mathtt{MS}\ y \equiv \mathtt{f}\ n \right\| \tag{95}$$

Proof. (cont.)

Using the axiom of countable choice, we get

$$\left\| \prod_{(n:\mathbb{N})} \sum_{(y:\mathtt{T}\,X/\sim_{\mathtt{T}})} \mathtt{T}/\sim\!\!\to\mathtt{MS}\ y \equiv \mathtt{f}\,n \right\| \tag{96}$$

By the surjectivity of $[\cdot]$ we get

$$\left\| \prod_{(n:\mathbb{N})} \sum_{(y:\mathtt{T}\,X)} \mathtt{T}\to\mathtt{MS}\ y \equiv \mathtt{f}\,n \right\| \tag{97}$$

By swapping Π and Σ and functional extensionality we get

$$\left\| \sum_{(g:\mathbb{N}\to\mathtt{T}\,X)} \mathtt{T}\to\mathtt{MS} \circ \mathtt{g} \equiv \mathtt{f} \right\| \tag{98}$$

which gives us the equalities

$$\mathtt{T}/\sim\!\!\to\mathtt{MS}\ [\mathtt{node}\ \mathtt{g}] \equiv \mathtt{node}\ \mathtt{f} \tag{99}$$

completing the case for the node constructor and thereby the proof. □

Partiality Monad

TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO
TODO TODO TODO TODO TODO TODO TODO TODO TODO

Quotient Polynomial Functor (QPF)

We can define quotiented $\mathtt{M}$-types from a quotient polynomial functor.

Definition (Quotient Polynomial Functor)

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

$$\mathtt{F}\, X = \sum_{(a:A)} ((\mathtt{B}\, a \to X)/\sim_a) \tag{100}$$

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

$$\mathtt{P} = \lambda \_, (\mathtt{B}\, a \to Y)/\sim_a \tag{101}$$

The base case is $\mathtt{f} = \lambda \_, [\, \mathtt{f} \circ \mathtt{g}\,]$ and equality case is $\mathtt{f}_{\mathtt{eq}} = \lambda\, \mathtt{x}\, \mathtt{y}\, r,\ \mathtt{eq}/\ (\mathtt{f} \circ \mathtt{x})\ (\mathtt{f} \circ \mathtt{y})\ (\sim_{\mathtt{ap}}\ \mathtt{f}\ r)$, where $\sim_{\mathtt{ap}}$ says that given a function $\mathtt{f}$ and $\mathtt{x} \sim_a \mathtt{y}$ then $\mathtt{f} \circ \mathtt{x} \sim_a \mathtt{f} \circ \mathtt{y}$. With this the definition for the quotient polynomial functor for functions is

$$\mathtt{F}\, \mathtt{f}\ (a, \mathtt{g}) = (a, \mathtt{elim}\ \mathtt{g}) \tag{102}$$

completing the definition of a quotiented polynomial functor.

Alternative Construction of QM-types

If there is a function $\mathtt{abs}_X : \mathtt{P}\,X \to \mathtt{F}\,X$ that makes the diagram bellow commute, then we can construct the final F-coalgebra, to get another notion of quotiented M-type.

$$\begin{array}{ccc} \mathtt{P}\,X & \xrightarrow{\mathtt{P}\,\mathtt{f}} & \mathtt{P}\,Y \\ {\scriptstyle \mathtt{abs}_X}\downarrow & & \downarrow{\scriptstyle \mathtt{abs}_Y} \\ \mathtt{F}\,X & \xrightarrow{\mathtt{F}\,\mathtt{f}} & \mathtt{F}\,Y \end{array}$$

We get a surjective function $\mathtt{QM\text{-}intro} : \mathtt{M}_S \to \mathtt{M}_S/\sim$ from this construction. The function is given as $\mathtt{abs}_{\mathtt{M}}$, which is the instantiation of $\mathtt{abs}$ between the limits of the two sequences.

Construction QIITs

We can construct a QIIT from a M-type and a relation, by defining them simultanouosly, and we can then show, it equal to the QM-type by giving

- A function QM→QIIT from QM to QIIT, taking the QM constructors to their corresponding QIIT constructor.
- A proof QM→QIIT-∼→≡ that $x \sim_{\mathtt{QM}} y$ implies QM→QIIT $x \equiv$ QM→QIIT y, which should follow by the constructors we added to the QIIT for the relation.
- Lifting QM→QIIT to QM/∼→QIIT using the quotient recursor with QM→QIIT-∼→≡ and QIIT-isSet
- Showing injectivity by using propositional elimination of the quotient together with the inverse of QM→QIIT-∼→≡, namely QM→QIIT-injective saying that QM→QIIT $x \equiv$ QM→QIIT y implies $x \sim_{\mathtt{QM}} y$, which again should follow from how the relations constructors were defined.
- Lastly we show surjectivity by induction using the eliminator of QIIT and the axiom of choice and the surjectivity of $[\cdot]$.

However this is not enough...

Conclusion

Future Work