(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)

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

for which we define an *inductive* equivalence relation

Definition (Equivalence for the Natural Number)

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

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.

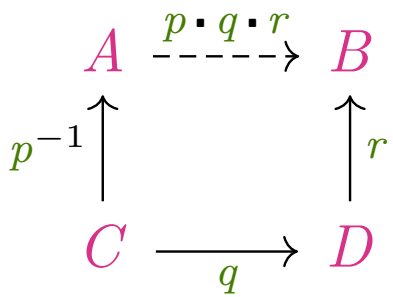


Figure: Composition square

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.

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

Figure: Chain of types / functions

Definition

The limit of a chain is given as

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

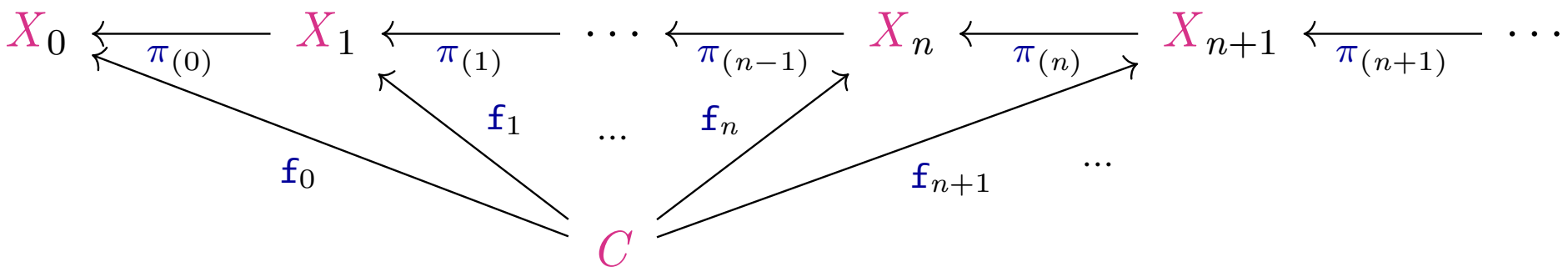
Helper Lemmas

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

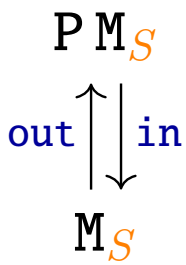


Figure: M-type diagram

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

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

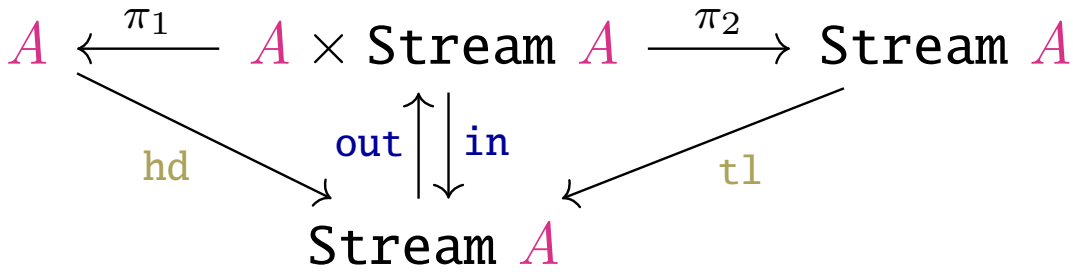


Figure: Streams

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

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

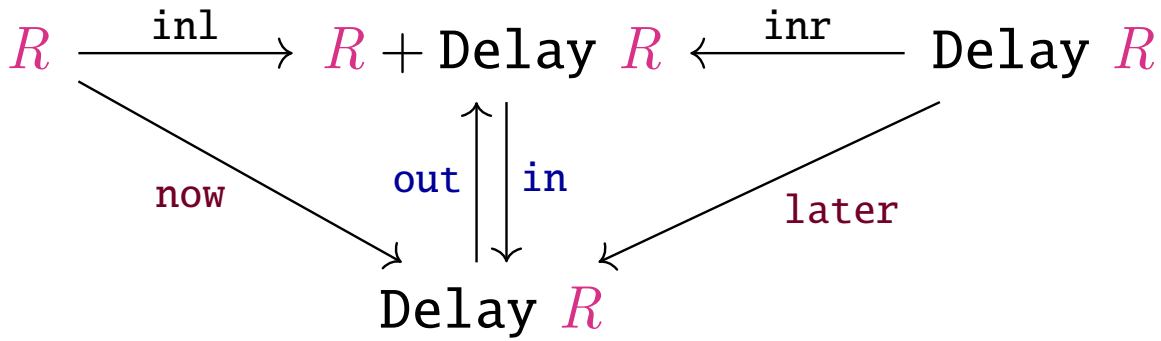
and a polynomial functor

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

which simplifies to

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

such that we get the diagram



Example: Event Trees

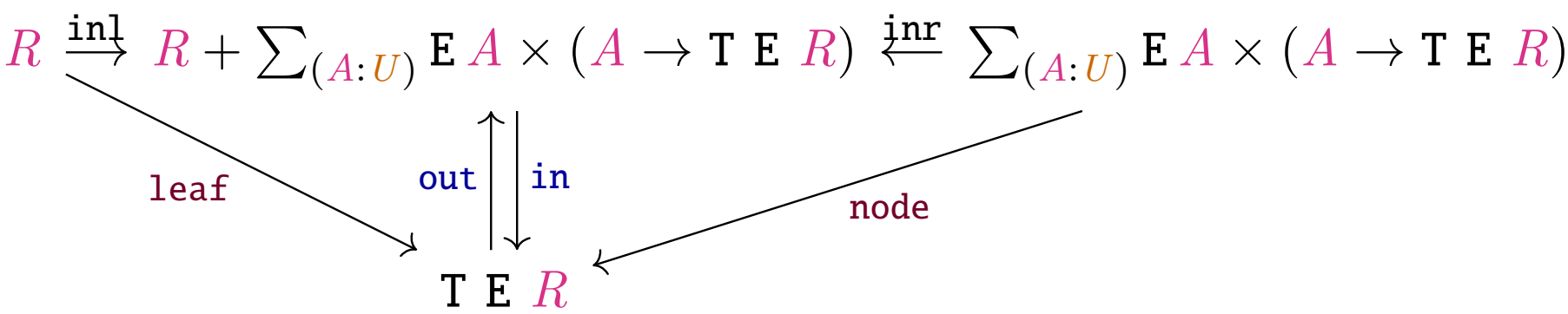
Definition

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

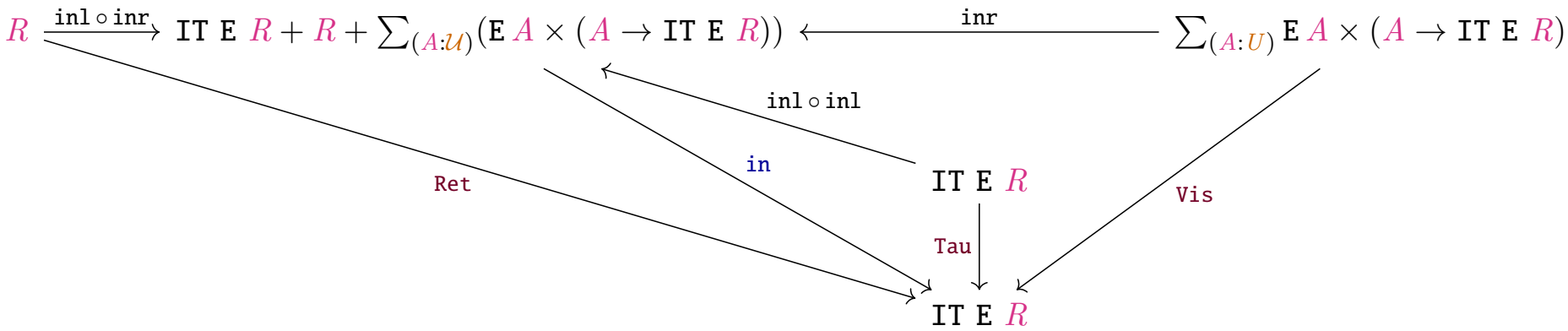
We define a container

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

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

Overview

Conclusion

Future Work