M-types and Coinduction in HoTT and Cubical Type Theory

Lasse Letager Hansen, 201912345

Master's Thesis, Computer Science
March 21, 2020
Advisor: Bas Spitters

AARHUS UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Abstract

in English…

Resumé

in Danish. . .

Acknowledgments

. . .

Lasse Letager Hansen,
Aarhus, March 21, 2020.

Contents

Abstract **iii**

Resumé **v**

Acknowledgments **vii**

1 Introduction **1**

2 M-types **3**
- 2.1 Containers / Signatures . . . 3
- 2.2 ITrees as `M`-types . . . 4
  - 2.2.1 Delay Monad . . . 5
  - 2.2.2 Tree . . . 5
  - 2.2.3 ITrees . . . 6
- 2.3 Co-induction Principle for M-types . . . 6
  - 2.3.1 Bisimulation of Streams . . . 7
  - 2.3.2 Bisimulation of Delay monad . . . 7
  - 2.3.3 Bisimulation of ITrees . . . 7
- 2.4 Stream Formalization using M-types . . . 8
- 2.5 Quotient M-type . . . 8
- 2.6 Closure properties of M-types . . . 8
  - 2.6.1 Zip Function . . . 9
  - 2.6.2 Examples of Fixed Points . . . 11

3 Additions to the Cubical Agda Library **13**
- 3.1 Lemma 10 . . . 14
- 3.2 Lemma 13 . . . 15
- 3.3 Missing postulates . . . 15

4 Conclusion **17**

Bibliography **19**

A The Technical Details **21**

Chapter 1

Introduction

motivate and explain the problem to be addressed

example of a citation: [1]

get your bibtex entries from `https://dblp.org/`

Chapter 2

M-types

2.1 Containers / Signatures

A Container (or Signature) is a pair $S = (A, B)$ of types $\vdash A : \mathcal{U}$ and $a : A \vdash B(a) : \mathcal{U}$. From a container we can define a polynomial functor, defined for objects (types) as

$$\begin{aligned} \mathtt{P}_S &: \mathcal{U} \to \mathcal{U} \\ \mathtt{P}(X) := \mathtt{P}_S(X) &= \sum_{a:A} B(a) \to X \end{aligned} \tag{2.1}$$

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

$$\begin{aligned} \mathtt{P}f &: \mathtt{P}X \to \mathtt{P}Y \\ \mathtt{P}f(a, g) &= (a, f \circ g) \end{aligned} \tag{2.2}$$

As an example lets look at type for streams over the type A, defined by the container $S = (A, \lambda\_, \mathbf{1})$, applying the polynomial functor we get

$$\mathtt{P}_S(X) = \sum_{a:A} \mathbf{1} \to X \tag{2.3}$$

since we are working in a Category with exponentials we get $\mathbf{1} \to X \equiv X^{\mathbf{1}} \equiv X$, furthermore $\mathbf{1}$ and X does not depend on A here, so this will be equivalent to the definition

$$\mathtt{P}_S(X) = A \times X \tag{2.4}$$

Now we define the coalgebra for this functor with type

$$\mathtt{Coalg}_S = \sum_{C:\mathcal{U}} C \to \mathtt{P}C \tag{2.5}$$

we denote a coalgera of C and γ as C-γ. The coalgebra morphisms are defined as

$$\begin{aligned} \_\Rightarrow\_ &: \mathtt{Coalg}_S \to \mathtt{Coalg}_S \\ C\text{-}\gamma \Rightarrow D\text{-}\delta &= \sum_{f:C\to D} \delta \circ f = \mathtt{P}f \circ \gamma \end{aligned} \tag{2.6}$$

$\mathtt{M}$-types can now be defined from a container S as the type $\mathtt{M}_S$ such that the coalgebra for $\mathtt{M}_S$ and $\mathtt{out} : \mathtt{M}_S \to \mathtt{P}_S(\mathtt{M}_S)$ fulfills the property

$$\mathtt{Final}_S := \sum_{(X\text{-}\rho:\mathtt{Coalg}_S)} \prod_{(C\text{-}\gamma:\mathtt{Coalg}_S)} \mathtt{isContr}(C\text{-}\gamma \Rightarrow X\text{-}\rho) \tag{2.7}$$

that is $\prod_{(C\text{-}\gamma:\mathtt{Coalg}_S)} \mathtt{isContr}(C\text{-}\gamma \Rightarrow \mathtt{M}_S\text{-}\mathtt{out})$. We denote this construction of the M-type as $\mathtt{M}_{(A\,,\,B)}$ or $\mathtt{M}_S$ or just $\mathtt{M}$ when the Container is clear from the context.

If we continue our example for streams this will give us the $\mathtt{M}$-type, we can see that $\mathtt{P}_S(\mathtt{M}) = A \times \mathtt{M}$, meaning we have the following diagram, where $\mathtt{out}$ is an isomorphism (because of the finality of

$$\begin{tikzcd}
A & A \times \mathtt{M}_{(A\,,\,\lambda\_,\mathbf{1})} \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & \mathtt{M}_{(A\,,\,\lambda\_,\mathbf{1})} \\
& \mathtt{M}_{(A\,,\,\lambda\_,\mathbf{1})} \arrow[ul, "\mathtt{hd}"] \arrow[u, "\mathtt{out}"'] \arrow[ur, "\mathtt{tl}"'] &
\end{tikzcd}$$

Figure 2.1: M-types of streams

the coalgebra), with inverse $\mathtt{in} : \mathtt{P}_S(\mathtt{M}) \to \mathtt{M}$. We now have a semantic for the rules we would expect for streams, if we let $\mathtt{cons} = \mathtt{in}$ and $\mathtt{stream}\ A = \mathtt{M}_{(A\,,\,\lambda\_,\mathbf{1})}$,

$$\frac{A : \mathcal{U} \quad s : \mathtt{stream}\ A}{\mathtt{hd}\ s : A}\ \mathtt{E}_{\mathtt{hd}} \tag{2.8}$$

$$\frac{A : \mathcal{U} \quad s : \mathtt{stream}\ A}{\mathtt{tl}\ s : \mathtt{stream}\ A}\ \mathtt{E}_{\mathtt{tl}} \tag{2.9}$$

$$\frac{A : \mathcal{U} \quad x : A \quad xs : \mathtt{stream}\ A}{\mathtt{cons}\ x\ xs : \mathtt{stream}\ A}\ \mathtt{I}_{\mathtt{cons}} \tag{2.10}$$

2.2 ITrees as M-types

We want the following rules for ITrees

$$\frac{r : R}{\mathtt{Ret}\ r : \mathtt{itree}\ \mathtt{E}\ R}\ \mathtt{I}_{\mathtt{Ret}} \tag{2.11}$$

$$\frac{A : \mathcal{U} \quad a : \mathtt{E}\ A \quad f : A \to \mathtt{itree}\ \mathtt{E}\ R}{\mathtt{Vis}\ a\ f : \mathtt{itree}\ \mathtt{E}\ R}\ \mathtt{I}_{\mathtt{Vis}}. \tag{2.12}$$

Elimination rules

$$\frac{t : \mathtt{itree}\ \mathtt{E}\ R}{\mathtt{Tau}\ t : \mathtt{itree}\ \mathtt{E}\ R}\ \mathtt{E}_{\mathtt{Tau}}. \tag{2.13}$$

2.2.1 Delay Monad

check this statement

We start by looking at itrees without the $\texttt{Vis}$ constructor, this type is also know as the delay monad. We construct this type by letting $S = (\mathbf{1} + R, \lambda\{\texttt{inl}\ \_ \to \mathbf{1}\ ;\ \texttt{inr}\ \_ \to \mathbf{0}\})$, we then get the polynomial functor

$$\texttt{P}_S(X) = \sum_{x:\mathbf{1}+R} \lambda\{\texttt{inl}\ \_ \to \mathbf{1}; \texttt{inr}\ \_ \to \mathbf{0}\}\ x \to X, \tag{2.14}$$

which is equal to

$$\texttt{P}_S(X) = X + R \times (\mathbf{0} \to X). \tag{2.15}$$

We know that $\mathbf{0} \to X \equiv \mathbf{1}$, so we can reduce further to

$$\texttt{P}_S(X) = X + R \tag{2.16}$$

meaning we get the following diagram.

$$\begin{tikzcd}
R \arrow[r, "\texttt{inl}"] \arrow[dr, "\texttt{now}"'] & R + \texttt{M} & \texttt{M} \arrow[l, "\texttt{inr}"'] \arrow[dl, "\texttt{later}"] \\
& \texttt{M} \arrow[u, "\texttt{out}"'] &
\end{tikzcd}$$

Figure 2.2: Delay monad

Meaning we can define the operations $\texttt{now}$ and $\texttt{later}$ using $\texttt{in} = \texttt{out}^{-1}$ together with the injections $\texttt{inl}$ and $\texttt{inr}$.

(Later = Tau, Ret = Now)

2.2.2 Tree

Now lets look at the example where we remove the $\texttt{Tau}$ constructor. We let

$$S = \left(R + \sum_{A:\mathcal{U}} \texttt{E}\ A\ ,\ \lambda\{\texttt{inl}\ \_ \to \mathbf{0}\ ;\ \texttt{inr}\ (A, e) \to A\}\right). \tag{2.17}$$

This will give us the polynomial functor

$$\texttt{P}_S(X) = \sum_{x:R+\sum_{A:\mathcal{U}} \texttt{E}\ A} \lambda\{\texttt{inl}\ \_ \to \mathbf{0}\ ;\ \texttt{inr}\ (A, e) \to A\}\ x \to X, \tag{2.18}$$

which simplifies to

$$\texttt{P}_S(X) = (R \times (\mathbf{0} \to X)) + (\sum_{A:\mathcal{U}} \texttt{E}\ A \times (A \to X)), \tag{2.19}$$

and further

$$\texttt{P}_S(X) = R + \sum_{A:\mathcal{U}} \texttt{E}\ A \times (A \to X). \tag{2.20}$$

We get the following diagram for the $\texttt{P}$-coalgebra.
Again we can define $\texttt{Ret}$ and $\texttt{Vis}$ using the $\texttt{in}$ function.

$$\begin{tikzcd}
R \arrow[r, "\texttt{inl}"] \arrow[dr, "\texttt{Ret}"'] & R+\sum_{A:U}\texttt{E}\ A\times(A\to\texttt{M}) & \sum_{A:U}\texttt{E}\ A\times(A\to\texttt{M}) \arrow[l, "\texttt{inr}"'] \arrow[dl, "\texttt{Vis}"] \\
& \texttt{M} \arrow[u, "\texttt{out}"] &
\end{tikzcd}$$

Figure 2.3: TODO

2.2.3 ITrees

Now we should have all the knowledge needed to make ITrees using M-types. We define ITrees by the container

$$S=\left(\mathbf{1}+R+\sum_{A:\mathcal{U}}(\texttt{E}\ A)\quad,\quad\lambda\left\{\texttt{inl}\ (\texttt{inl}\ \_)\to\mathbf{1}\ ;\ \texttt{inl}\ (\texttt{inr}\ \_)\to\mathbf{0}\ ;\ \texttt{inr}(A,\_)\to A\right\}\right). \tag{2.21}$$

Such that the (reduced) polynomial functor becomes

$$\texttt{P}_S(X)=X+R+\sum_{A:\mathcal{U}}((\texttt{E}\ A)\times(A\to X)) \tag{2.22}$$

Giving us the diagram

$$\begin{tikzcd}
R \arrow[r, "\texttt{inl}\circ\texttt{inr}"] \arrow[ddrr, "\texttt{Ret}"'] & \texttt{M}+R+\sum_{A:\mathcal{U}}(\texttt{E}\ A\times(A\to\texttt{M})) & & \sum_{A:U}\texttt{E}\ A\times(A\to\texttt{M}) \arrow[lll, "\texttt{inr}"'] \arrow[ddl, "\texttt{Vis}"] \\
& & \texttt{M} \arrow[ul, "\texttt{inl}\circ\texttt{inl}"'] \arrow[d, "\texttt{Tau}"'] & \\
& & \texttt{M} \arrow[uul, "\texttt{out}"] &
\end{tikzcd}$$

Figure 2.4: TODO

2.3 Co-induction Principle for M-types

We can now construct a bisimulation: forall coalgebras C-γ : $\texttt{Coalg}_S$, if we have a relation $\mathcal{R}: C\to C\to\mathcal{U}$, and a type $\overline{\mathcal{R}}=\sum_{a:C}\sum_{b:C}\mathcal{R}\ a\ b$, such that $\overline{\mathcal{R}}$ and $\alpha_{\overline{\mathcal{R}}}:\overline{\mathcal{R}}\to\texttt{P}(\overline{\mathcal{R}})$ forms a P-coalgebra $\overline{\mathcal{R}}$-$\alpha_{\overline{\mathcal{R}}}$: $\texttt{Coalg}_S$, making the diagram in Figure 2.5 commute (where $\Longrightarrow$ are P-coalgebra morphisms). Furthermore for any bisimulation over a final P-coalgebra M-$\texttt{out}$: $\texttt{Coalg}_S$

$$\begin{tikzcd}
C\text{-}\gamma & \overline{\mathcal{R}}\text{-}\alpha_{\overline{\mathcal{R}}} \arrow[l, Rightarrow, "\pi_1{}^{\overline{\mathcal{R}}}"'] \arrow[r, Rightarrow, "\pi_2{}^{\overline{\mathcal{R}}}"] & C\text{-}\gamma
\end{tikzcd}$$

Figure 2.5: Bisimulation for a coalgebra

we get the diagram in Figure 2.6, where $\pi_1^{\overline{\mathcal{R}}}=!=\pi_2^{\overline{\mathcal{R}}}$, which means given $r:\mathcal{R}(m,m')$ we get $m=\pi_1^{\overline{\mathcal{R}}}(m,m',r)=\pi_2^{\overline{\mathcal{R}}}(m,m',r)=m'$.

$$\begin{tikzcd} \texttt{M-out} & \overline{\mathcal{R}}\text{-}\alpha_{\overline{\mathcal{R}}} \arrow[l, Rightarrow, "{\pi_1}^{\overline{\mathcal{R}}}"'] \arrow[r, Rightarrow, "{\pi_2}^{\overline{\mathcal{R}}}"] & \texttt{M-out} \end{tikzcd}$$

Figure 2.6: Bisimulation principle for final coalgebra

Better introduction to this proof!

We want to define a co-induction principle from any bisimulation relation over a final coalgebra, that is if R gives a bisimulation, then it is true that

$$R \equiv \equiv \tag{2.23}$$

meaning we can use the relation R, to show that two things of an M-type are equivalent. So we want to construct an isomorphism between R and the equivalence relation $\equiv$, to do this we must construct functions

$$p : R \to \equiv \tag{2.24}$$
$$q : \equiv \to R \tag{2.25}$$

and relations

$$\alpha : p \circ q \equiv \texttt{id}_{\equiv} \tag{2.26}$$
$$\beta : q \circ p \equiv \texttt{id}_R \tag{2.27}$$

Complete the construction of equality from any bisimulation relation over an M-type

2.3.1 Bisimulation of Streams

TODO

2.3.2 Bisimulation of Delay monad

TODO

2.3.3 Bisimulation of ITrees

We define our bisimulation coalgebra from the strong bisimulation relation $\mathcal{R}$, defined by the following rules.

$$\frac{a, b : R \quad a \equiv_R b}{\texttt{Ret}\ a \cong \texttt{Ret}\ b}\ \texttt{EqRet} \tag{2.28}$$

$$\frac{t, u : \texttt{itree E}\ R \quad t \cong u}{\texttt{Tau}\ t \cong \texttt{Tau}\ u}\ \texttt{EqTau} \tag{2.29}$$

$$\frac{A : \mathcal{U} \quad e : \texttt{E}\ A \quad k_1, k_2 : A \to \texttt{itree E}\ R \quad t \cong u}{\texttt{Vis}\ e\ k_1 \cong \texttt{Tau}\ e\ k_2}\ \texttt{EqVis} \tag{2.30}$$

Now we just need to define $\alpha_{\overline{\mathcal{R}}}$. Now we have a bisimulation relation, which is equivalent to equality, using what we showed in the previous section.

define the $\alpha_{\overline{R}}$ function

2.4 Stream Formalization using M-types

When taking the head of a stream, we get

$$\begin{aligned} \texttt{hd}\ (\texttt{cons}\ x\ xs) &\equiv \pi_1\ \texttt{out}\ (\texttt{cons}\ x\ xs) && (2.31)\\ &\equiv \pi_1\ \texttt{out}\ (\texttt{in}\ (x, \lambda\_, xs)) && (2.32)\\ &\equiv \pi_1\ (x, \lambda\_, xs) && (2.33)\\ &\equiv x && (2.34) \end{aligned}$$

2.5 Quotient M-type

Since we know that M-types preserves the H-level, we can use set-truncated quotients, to define quotient M-types, for examples we can define weak bisimulation of the delay monad as

Quotients of the delay monad

2.6 Closure properties of M-types

We define the product of two containers

$$(A, B) \times (C, D) \equiv (A \times C, \lambda\,(a, c), \texttt{B}\ a \times \texttt{D}\ c), \tag{2.35}$$

we can lift this rule, through the diagram in Figure 2.7, used to define M-types. We now prove that

$$\texttt{P}_{(A,B)}{}^{n}\ \mathbf{1} \times \texttt{P}_{(C,D)}{}^{n}\ \mathbf{1} \equiv \texttt{P}_{(A,B)\times(C,D)}{}^{n}\ \mathbf{1}, \tag{2.36}$$

by induction on n. For $n = 0$, we have $\mathbf{1} \times \mathbf{1} \equiv \mathbf{1}$. For $n = m + 1$, we may assume

$$\texttt{P}_{(A,B)}{}^{m}\ \mathbf{1} \times \texttt{P}_{(C,D)}{}^{m}\ \mathbf{1} \equiv \texttt{P}_{(A,B)\times(C,D)}{}^{m}\ \mathbf{1}, \tag{2.37}$$

and show

$$\begin{aligned} &\texttt{P}_{(A,B)}{}^{m+1}\ \mathbf{1} \times \texttt{P}_{(C,D)}{}^{m+1}\ \mathbf{1} && (2.38)\\ \equiv\ &\texttt{P}_{(A,B)}(\texttt{P}_{(A,B)}{}^{m}\ \mathbf{1}) \times \texttt{P}_{(C,D)}(\texttt{P}_{(C,D)}{}^{m}\ \mathbf{1}) && (2.39)\\ \equiv\ &\sum_{a:A} B\ a \to \texttt{P}_{(A,B)}{}^{m}\ \mathbf{1} \times \sum_{c:C} D\ c \to \texttt{P}_{(C,D)}{}^{m}\ \mathbf{1} && (2.40)\\ \equiv\ &\sum_{a,c:A\times C} (B\ a \to \texttt{P}_{(A,B)}{}^{m}\ \mathbf{1}) \times (D\ c \to \texttt{P}_{(C,D)}{}^{m}\ \mathbf{1}) && (2.41)\\ \equiv\ &\sum_{a,c:A\times C} B\ a \times D\ c \to \texttt{P}_{(A,B)}{}^{m}\ \mathbf{1} \times \texttt{P}_{(C,D)}{}^{m}\ \mathbf{1} && (2.42)\\ \equiv\ &\sum_{a,c:A\times C} B\ a \times D\ c \to \texttt{P}_{(A,B)\times(C,D)}{}^{m}\ \mathbf{1} && (2.43)\\ \equiv\ &\texttt{P}_{(A,B)\times(C,D)}(\texttt{P}_{(A,B)\times(C,D)}{}^{m}\ \mathbf{1}) && (2.44)\\ \equiv\ &\texttt{P}_{(A,B)\times(C,D)}{}^{m+1}\ \mathbf{1} && (2.45) \end{aligned}$$

$$\begin{tikzcd}
A & A \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
\texttt{P}(\texttt{M}_{(A\,,\,B)}) \arrow[u, "\pi_1"'] & \texttt{P}(\texttt{M}_{(A\,,\,B)\times(C\,,\,D)}) \arrow[u, "\pi_1"'] & \texttt{P}(\texttt{M}_{(C\,,\,D)}) \arrow[u, "\pi_1"'] \\
\texttt{M}_{(A\,,\,B)} \arrow[u, "\texttt{out}"'] \arrow[d, "\pi_{(!)}"] & \texttt{M}_{(A\,,\,B)\times(C\,,\,D)} \arrow[u, "\texttt{out}"'] \arrow[d, "\pi_{(!)}"] & \texttt{M}_{(C\,,\,D)} \arrow[u, "\texttt{out}"'] \arrow[d, "\pi_{(!)}"] \\
\texttt{P}_{(A\,,\,B)}{}^{n}\ \mathbf{1} \arrow[d, "\pi_{(n)}"] & \texttt{P}_{(A\,,\,B)\times(C\,,\,D)}{}^{n}\ \mathbf{1} \arrow[l] \arrow[r] \arrow[d, "\pi_{(n)}"] & \texttt{P}_{(C\,,\,D)}{}^{n}\ \mathbf{1} \arrow[d, "\pi_{(n)}"] \\
\vdots \arrow[d, "\pi_{(1)}"] & \vdots \arrow[d, "\pi_{(1)}"] & \vdots \arrow[d, "\pi_{(1)}"] \\
\texttt{P}_{(A\,,\,B)}\ \mathbf{1} \arrow[d, "\pi_{(0)}"] & \texttt{P}_{(A\,,\,B)\times(C\,,\,D)}\ \mathbf{1} \arrow[l] \arrow[r] \arrow[d, "\pi_{(0)}"] & \texttt{P}_{(C\,,\,D)}\ \mathbf{1} \arrow[d, "\pi_{(0)}"] \\
\mathbf{1} & \mathbf{1} \arrow[l, "\texttt{refl}"] \arrow[r, "\texttt{refl}"'] & \mathbf{1}
\end{tikzcd}$$

Figure 2.7: TODO

taking the limit we get

$$\texttt{M}_{(A\,,\,B)} \times \texttt{M}_{(C\,,\,D)} \equiv \texttt{M}_{(A\,,\,B)\times(C\,,\,D)} \tag{2.46}$$

as an example hereof lets look at streams, where we actually get

$$\texttt{stream}\ A \times \texttt{stream}\ B \equiv \texttt{M}_{(A\,,\,\lambda\ \_,\mathbf{1})} \times \texttt{M}_{(B\,,\,\lambda\ \_,\mathbf{1})} \equiv \texttt{M}_{(A\,,\,\lambda\ \_,\mathbf{1})\times(B\,,\,\lambda\ \_,\mathbf{1})} \equiv \texttt{stream}\ (A \times B) \tag{2.47}$$

as expected, transporting along equality (2.47) gives us the definition for `zip`.

2.6.1 Zip Function

We want the diagram in Figure 2.8 to commute, meaning we get the computation rules

$$(\texttt{hd} \times \texttt{hd}) \equiv \texttt{hd} \circ \texttt{zip} \tag{2.48}$$

$$\texttt{zip} \circ (\texttt{tl} \times \texttt{tl}) \equiv \texttt{tl} \circ \texttt{zip} \tag{2.49}$$

$$\begin{tikzcd}
 & A \times B & \\
\texttt{stream}\ A \times \texttt{stream}\ B \arrow[ur, "\texttt{hd}\times\texttt{hd}"] \arrow[rr, "\texttt{zip}"] \arrow[d, "\texttt{tl}\times\texttt{tl}"] & & \texttt{stream}\ (A \times B) \arrow[ul, "\texttt{hd}"'] \arrow[d, "\texttt{tl}"] \\
\texttt{stream}\ A \times \texttt{stream}\ B \arrow[rr, "\texttt{zip}"] & & \texttt{stream}\ (A \times B)
\end{tikzcd}$$

Figure 2.8: TODO

we can define the zip function as we did in the end of the last section. Another way to define the zip function is more directly, using the following lifting property of M-types

$$\texttt{lift}_{\texttt{M}} \left(x : \prod_{n:\mathbb{N}} (A \to \texttt{P}_S{}^n\ \mathbf{1})\right) \left(u : \prod_{n:\mathbb{N}} (A \to \pi_n (x_{n+1} a) \equiv x_n a)\right) (a : A) : \texttt{M}\ S := (\lambda\, n, x\ n\ a), (\lambda\, n\ i, p\ n\ a\ i). \tag{2.50}$$

To use this definition, we first define some helper functions

$$\texttt{zip}_X\ n\ (x,y) = \begin{cases} \mathbf{1} & \texttt{if}\ n = 0 \\ (\texttt{hd}\ x, \texttt{hd}\ y), (\lambda\, \_, \texttt{zip}_X\ m\ (\texttt{tl}\ x, \texttt{tl}\ y)), & \texttt{if}\ n = m+1 \end{cases} \tag{2.51}$$

$$\texttt{zip}_\pi\ n\ (x,y) = \begin{cases} \texttt{refl} & \texttt{if}\ n = 0 \\ \lambda\, i, (\texttt{hd}\ x, \texttt{hd}\ y), (\lambda\, \_, \texttt{zip}_\pi\ m\ (\texttt{tl}\ x, \texttt{tl}\ y)\ i), & \texttt{if}\ n = m+1 \end{cases}, \tag{2.52}$$

we can then define

$$\texttt{zip}_{lift}\ (x,y) := \texttt{lift}_{\texttt{M}}\ \texttt{zip}_X\ \texttt{zip}\ (x,y). \tag{2.53}$$

Equality of Zip Definitions

We would expect that the two definitions for zip are equal

$$\begin{aligned} \texttt{transport}_?\ a &\equiv \texttt{zip}_{lift}\ a && (2.54)\\ &\equiv \texttt{lift}_{\texttt{M}}\ \texttt{zip}_X\ \texttt{zip}_\pi\ (x,y) && (2.55)\\ &\equiv (\lambda\, n, \texttt{zip}_X\ n\ (x,y)), (\lambda\, n\ i, \texttt{zip}_\pi\ n\ (x,y)\ i) && (2.56) \end{aligned}$$

zero case X

$$\texttt{zip}_X\ 0\ (x,y) \equiv \mathbf{1} \tag{2.57}$$

Sucessor case X

$$\begin{aligned} \texttt{zip}_X\ (m+1)\ (x,y) &\equiv (\texttt{hd}\ x, \texttt{hd}\ y), (\lambda\, \_, \texttt{zip}_X\ m\ (\texttt{tl}\ x, \texttt{tl}\ y)) && (2.58)\\ &\equiv (\texttt{hd}\ x, \texttt{hd}\ y), (\lambda\, \_, ?\ (\texttt{tl}\ a)) && (2.59)\\ &\equiv (\texttt{hd}\ (\texttt{transport}_? a)), (\lambda\, \_, \texttt{transport}_? (\texttt{tl}\ a)) && (2.60)\\ &\equiv transport_? a && (2.61)\\ & && (2.62) \end{aligned}$$

Zero case π: $(\lambda\, i, \mathtt{zip}_\pi\ 0\ (x,y)\ i \equiv refl)$.

$$\begin{aligned}
&\equiv (), (\lambda\, i, \mathtt{zip}_\pi\ 0\ (x,y)\ i) && (2.63)\\
&\equiv \mathbf{1}, \mathtt{refl} && (2.64)\\
& && (2.65)
\end{aligned}$$

successor case

$$\begin{aligned}
&\equiv (\mathtt{zip}_X\ (m+1)\ (x,y)), (\lambda\, i, \mathtt{zip}_\pi\ (m+1)\ (x,y)\ i) && (2.66)\\
&\equiv ((\mathtt{hd}\ x, \mathtt{hd}\ y), (\lambda\_, \mathtt{zip}_X\ m\ (\mathtt{tl}\ x, \mathtt{tl}\ y))), (\lambda\, i, (\mathtt{hd}\ x, \mathtt{hd}\ y), (\lambda\_, \mathtt{zip}_\pi\ m\ (\mathtt{tl}\ x, \mathtt{tl}\ y)\ i)) && (2.67)
\end{aligned}$$

Complete this proof

2.6.2 Examples of Fixed Points

Zeros

Let us try to define the zero stream, we do this by lifting the functions

$$\mathtt{const}_\mathtt{X}\ (n : \mathbb{N})\ (c : \mathbb{N}) := \begin{cases} \mathbf{1} & n = 0 \\ (c, \lambda\_, \mathtt{const}_\mathtt{X}\ m\ c) & n = m+1 \end{cases} \tag{2.68}$$

$$\mathtt{const}_\pi\ (n : \mathbb{N})\ (c : \mathbb{N}) := \begin{cases} \mathtt{refl} & n = 0 \\ \lambda\, i, (c, \lambda\_, \mathtt{const}_\pi\ m\ c\ i) & n = m+1 \end{cases} \tag{2.69}$$

to get the definition of zero stream

$$\mathtt{zeros} := \mathtt{lift}_\mathtt{M}\ \mathtt{const}_\mathtt{X}\ \mathtt{const}_\pi\ 0. \tag{2.70}$$

We want to show that we get the expected properties, such as

$$\begin{aligned}
\mathtt{hd}\ \mathtt{zeros} &\equiv 0 && (2.71)\\
\mathtt{tl}\ \mathtt{zeros} &\equiv \mathtt{zeros} && (2.72)
\end{aligned}$$

Spin

We want to define spin, as being the fixed point $\mathtt{spin} = \mathtt{later}\ \mathtt{spin}$, so that is again a final coalgebra, but of a M-type (which is a final coalgebra)
Since it is final, it also must be unique, meaning that there is just one program that spins forever, without returning a value, meaning every other program must return a value. If we just

$$\begin{tikzcd} \texttt{spin} \arrow[d, "\texttt{later}"] \\ \texttt{later spin} \end{tikzcd}$$

Figure 2.9: TODO

Chapter 3

Additions to the Cubical Agda Library

3.1 Lemma 10

`M`-types are part of a final coalgebra, formally $\forall\, S\ C\text{-}\gamma, (C\text{-}\gamma \Rightarrow \texttt{M-out}) \equiv \mathbf{1}$

$$\begin{aligned}
U &\equiv \sum_{f:C\to\mathcal{L}} \texttt{out} \circ f \equiv \texttt{step}\ f && (3.1)\\
&\equiv \sum_{f:C\to\mathcal{L}} \texttt{in} \circ \texttt{out} \circ f \equiv \texttt{in} \circ \texttt{step}\ f && (3.2)\\
&\equiv \sum_{f:C\to\mathcal{L}} f \equiv \texttt{in} \circ \texttt{step}\ f && (3.3)\\
&\equiv \sum_{f:C\to\mathcal{L}} f \equiv \Psi\ f && (3.4)\\
&\equiv \sum_{c:Cone} e\ c \equiv \Psi\ (e\ c) && (3.5)\\
&\equiv \sum_{c:Cone} e\ c \equiv e\ (\phi\ c) && (3.6)\\
&\equiv \sum_{c:Cone} c \equiv \phi\ c && (3.7)\\
&\equiv \sum_{(u,q):Cone} (u, q) \equiv (\phi_0\ u, \phi_1\ u\ q) && (3.8)\\
&\equiv \sum_{(u,q):Cone} \sum_{p:u\equiv\phi_0\ u} q \equiv_{\lambda i, Cone_1 (p\ i)} \phi_1\ u\ q && (3.9)\\
&\equiv \sum_{(u,p):\sum_{u:Cone_0} u\equiv\phi_0\ u} \sum_{q:Cone_1\ u} q \equiv_{\lambda i, Cone_1 (p\ i)} \phi_1\ u\ q && (3.10)\\
&\equiv \sum_{q:Cone_1 u_0} q \equiv_{\lambda i, Cone_1 (funExt\ p_0\ i)} \phi_1\ u_0\ q && (3.11)\\
\vdots & && (3.12)\\
&\equiv \mathbf{1} && (3.13)
\end{aligned}$$

3.2 Lemma 13

$$\sum_{((a,q):\sum_{a:\mathbb{N}\to A}\prod_{n:\mathbb{N}} a_{n+1}\equiv a_n)} \sum_{(u:\prod_{n:\mathbb{N}} Ba_n \to X_n)} \prod_{(n:\mathbb{N})} \pi \circ u_{n+1} \equiv_{\lambda i, B(q_n\ i)\to X_n} u_n \tag{3.14}$$

$$\equiv \sum_{a:A} \sum_{u:\prod_{n:\mathbb{N}} Ba \to X_n} \prod_{n:\mathbb{N}} \pi \circ u_{n+1} \equiv u_n \tag{3.15}$$

$$\equiv \sum_{a:A} \sum_{u:\prod_{n:\mathbb{N}} Ba \to X_n} \prod_{n:\mathbb{N}} \pi \circ u_{n+1} \equiv u_n \tag{3.16}$$

3.3 Missing postulates

Combine

For all $X : \mathbb{N} \to U$ and $p : \prod_{n:\mathbb{N}} X\ (n+1) \to X\ n \to U$

$$\sum_{x_0:X_0} \sum_{y:\prod_{n:\mathbb{N}} X_{n+1}} (p\ y_0\ x_0) \times \left(\prod_{n:\mathbb{N}} p\ y_{n+1}\ y_n \right) \tag{3.17}$$

$$\equiv \sum_{x_0:X_0} \sum_{y:\prod_{n:\mathbb{N}} X_{n+1}} (p\ y_0\ x_0) \times \left(\prod_{n:\mathbb{N}} p\ y_{n+1}\ y_n \right) \tag{3.18}$$

$$\equiv \sum_{x:\prod_{n:\mathbb{N}} \to X_n} (p\ x_1\ x_0) \times \left(\prod_{n:\mathbb{N}} p\ x_{n+2}\ x_{n+1} \right) \tag{3.19}$$

Chapter 4

Conclusion

conclude on the problem statement from the introduction

Bibliography

[1] Amin Timany and Matthieu Sozeau. Cumulative inductive types in coq. *LIPIcs: Leibniz International Proceedings in Informatics*, 2018.

Appendix A

The Technical Details

. . .