M-types and Coinduction in HoTT and Cubical Type Theory

Lasse Letager Hansen, 201912345

Master's Thesis, Computer Science
March 15, 2020
Advisor: Bas Spitters

AARHUS UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Abstract

in English...

Resumé

in Danish...

Acknowledgments

. . .

Lasse Letager Hansen,
Aarhus, March 15, 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 Quotient M-type . . . 8
- 2.5 Closure properties of M-types . . . 8
  - 2.5.1 Closure under products . . . 10
  - 2.5.2 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

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

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

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

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

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

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

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

Now we define the coalgebra for this functor with type

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

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

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

$\mathtt{M}$-types can now be defined from a container S as the type $\mathtt{M}_S$ such that the coalgebra for $\mathtt{M}_S$ and $\mathtt{out} : \mathtt{M}_S \to \mathbf{P}_S(\mathtt{M}_S)$ 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 $\mathbf{P}_S(\mathtt{M}) = A \times \mathtt{M}$, meaning we have the following diagram, where out is an isomorphism (because of the finality of the

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

Figure 2.1: M-types of streams

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

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

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

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

2.2 ITrees as M-types

We want the following rules for ITrees

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

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

Elimination rules

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

2.2.1 Delay Monad

We start by looking at itrees without the 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

check this statement

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

which is equal to

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

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

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

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

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

which simplifies to

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

and further

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

We get the following diagram for the **P**-coalgebra.
Again we can define Ret and Vis using the in function.

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

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

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

Such that the (reduced) polynomial functor becomes

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

Giving us the diagram

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

Figure 2.4: TODO

2.3 Co-induction Principle for M-types

We can now construct a bisimulation: forall coalgebras C-γ : $\mathtt{Coalg}_S$, 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 $\alpha_{\overline{\mathcal{R}}} : \overline{\mathcal{R}} \to \mathbf{P}(\overline{\mathcal{R}})$ forms a $\mathbf{P}$-coalgebra $\overline{\mathcal{R}}$-$\alpha_{\overline{\mathcal{R}}}$: $\mathtt{Coalg}_S$, making the following diagram commute (where $\Longrightarrow$ are $\mathbf{P}$-coalgebra morphisms).

$$
C\text{-}\gamma \xLeftarrow{\pi_1^{\overline{\mathcal{R}}}} \overline{\mathcal{R}}\text{-}\alpha_{\overline{\mathcal{R}}} \xRightarrow{\pi_2^{\overline{\mathcal{R}}}} C\text{-}\gamma
$$

Figure 2.5: TODO

Furthermore for any bisimulation over a final $\mathbf{P}$-coalgebra $\mathtt{M}$-$\mathtt{out}$: $\mathtt{Coalg}_S$ we have the following diagram,

$$\texttt{M-out} \xLeftarrow{\pi_1{}^{\overline{\mathcal{R}}}} \overline{\mathcal{R}}\text{-}\alpha_{\overline{\mathcal{R}}} \xRightarrow{\pi_2{}^{\overline{\mathcal{R}}}} \texttt{M-out}$$

Figure 2.6: TODO

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

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.

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

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

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

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 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

2.5 Closure properties of M-types

We define the product of two containers

$$(A, B) \times (C, D) \equiv (A \times C, \lambda(a, c), Ba \times Dc), \tag{2.31}$$

we can lift this rule, through the following diagram, used to define M-types
We now prove that

$$\mathbf{P}_{(A,B)}{}^n\ \mathbf{1} \times \mathbf{P}_{(C,D)}{}^n\ \mathbf{1} \equiv \mathbf{P}_{(A,B)\times(C,D)}{}^n\ \mathbf{1}, \tag{2.32}$$

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

$$\mathbf{P}_{(A,B)}{}^m\ \mathbf{1} \times \mathbf{P}_{(C,D)}{}^m\ \mathbf{1} \equiv \mathbf{P}_{(A,B)\times(C,D)}{}^m\ \mathbf{1}, \tag{2.33}$$

and show

$$\mathbf{P}_{(A,B)}{}^{m+1}\ \mathbf{1} \times \mathbf{P}_{(C,D)}{}^{m+1}\ \mathbf{1} \tag{2.34}$$
$$\equiv \mathbf{P}_{(A,B)}(\mathbf{P}_{(A,B)}{}^m\ \mathbf{1}) \times \mathbf{P}_{(C,D)}(\mathbf{P}_{(C,D)}{}^m\ \mathbf{1}) \tag{2.35}$$
$$\equiv \sum_{a:A} B\ a \to \mathbf{P}_{(A,B)}{}^m\ \mathbf{1} \times \sum_{c:C} D\ c \to \mathbf{P}_{(C,D)}{}^m\ \mathbf{1} \tag{2.36}$$
$$\equiv \sum_{a,c:A\times C} (B\ a \to \mathbf{P}_{(A,B)}{}^m\ \mathbf{1}) \times (D\ c \to \mathbf{P}_{(C,D)}{}^m\ \mathbf{1}) \tag{2.37}$$
$$\equiv \sum_{a,c:A\times C} B\ a \times D\ c \to \mathbf{P}_{(A,B)}{}^m\ \mathbf{1} \times \mathbf{P}_{(C,D)}{}^m\ \mathbf{1} \tag{2.38}$$
$$\equiv \sum_{a,c:A\times C} B\ a \times D\ c \to \mathbf{P}_{(A,B)\times(C,D)}{}^m\ \mathbf{1} \tag{2.39}$$
$$\equiv \mathbf{P}_{(A,B)\times(C,D)}(\mathbf{P}_{(A,B)\times(C,D)}{}^m\ \mathbf{1}) \tag{2.40}$$
$$\equiv \mathbf{P}_{(A,B)\times(C,D)}{}^{m+1}\ \mathbf{1} \tag{2.41}$$

taking the limit we get

$$M_{(A,B)} \times M_{(C,D)} \equiv M_{(A,B)\times(C,D)} \tag{2.42}$$

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

$$\mathtt{stream}\ A \times \mathtt{stream}\ B \equiv M_{(A,\lambda\ \_,1)\times(B,\lambda\ \_,1)} \equiv \mathtt{stream}\ (A \times B) \tag{2.43}$$

$$
\begin{tikzcd}
A & A \times C \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & C \\
\mathbf{P}(M_{(A,B)}) \arrow[u, "fst"] & \mathbf{P}(M_{(A\times C, B\times D)}) \arrow[u, "fst"] & \mathbf{P}(M_{(C,D)}) \arrow[u, "fst"'] \\
M_{(A,B)} \arrow[u, "out"] \arrow[d, "\pi_!"'] & M_{(A\times C, B\times D)} \arrow[u, "out"] \arrow[d, "\pi_!"'] & M_{(C,D)} \arrow[u, "out"] \arrow[d, "\pi_!"'] \\
\mathbf{P}_{(A,B)}{}^n\ \mathbf{1} \arrow[d, "\pi_n"'] & \mathbf{P}_{(A,B)\times(C,D)}{}^n\ \mathbf{1} \arrow[l] \arrow[r] \arrow[d, "\pi_n"'] & \mathbf{P}_{(C,D)}{}^n\ \mathbf{1} \arrow[d, "\pi_n"'] \\
\vdots \arrow[d, "\pi_1"'] & \vdots \arrow[d, "\pi_1"'] & \vdots \arrow[d, "\pi_1"'] \\
\mathbf{P}_{(A,B)}\ \mathbf{1} \arrow[d, "\pi_0"'] & \mathbf{P}_{(A,B)\times(C,D)}\ \mathbf{1} \arrow[l] \arrow[r] \arrow[d, "\pi_0"'] & \mathbf{P}_{(C,D)}\ \mathbf{1} \arrow[d, "\pi_0"'] \\
\mathbf{1} & \mathbf{1} \arrow[l] \arrow[r] & \mathbf{1}
\end{tikzcd}
$$

Figure 2.7: TODO

as expected, transporting along this gives us the definition for zip. More precisely we define zip as:

We start by defining the equality,

$$
\begin{aligned}
& W\ (A,B)\ (n+1) \times W\ (C,D)\ (n+1) \\
\equiv\ & \left(\sum_{(a:A)} B\ a \to W\ (A,B)\ n \right) \times \left(\sum_{(c:C)} D\ c \to W\ (C,D)\ n \right) \\
\equiv\ & \sum_{((a,c):A\times C)} (B\ a \to W\ (A,B)\ n) \times (D\ c \to W\ (C,D)\ n)
\end{aligned} \tag{2.44}
$$

by the two functions

$$f : ((a,c),(b,d)) \to ((a,b),(c,d)) \tag{2.45}$$
$$g : ((a,b),(c,d)) \to ((a,c),(b,d)) \tag{2.46}$$

such that $f \circ g \equiv id$ and $g \circ f \equiv id$ by $\texttt{refl}$.

$$zip = \texttt{transport}_{(2.44)} \square \; (\textstyle\sum_{(a,c):\texttt{refl}} ? \; \square \texttt{cong} \; (\lambda x, Ba \times Dc \to x)((\textbf{??}))) \tag{2.47}$$

transporting over an equality $A \equiv B$, given by $f : A \to B$ and $g : B \to A$, is the same as applying the function f. The definition of zip therefore reduce do

$$zip = f \circ (\texttt{transport}_{\sum_{(a,c):\texttt{refl}} ? \; \square \texttt{cong} \; (\lambda x, Ba \times Dc \to x)((\textbf{??}))}) \tag{2.48}$$

2.5.1 Closure under products

The product of two M-types is again an M-type

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

Figure 2.8: TODO

From this we get the computation rules

$$hd \times hd \equiv hd \circ zip \tag{2.49}$$

$$zip \circ tl \times tl \equiv tl \circ zip \tag{2.50}$$

another way to define the zip function is more directly, using the following lifting property of M-types

$$\texttt{lift}_{\texttt{M}} \left(x : \prod_{n:\mathbb{N}} (A \to \mathbf{P}_S{}^n \; \mathbf{1})\right) \left(u : \prod_{n:\mathbb{N}} (A \to \pi_n(x_{n+1} a) \equiv x_n a)\right) (a : A) : \texttt{M} \; S := (\lambda \, n, x \; n \; a), (\lambda \, n \; i, p \; n \; a \; i). \tag{2.51}$$

We can then define $\texttt{zip}_{lift}$ as

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

$$\texttt{zip}_\pi \; n \; (x, y) = \begin{cases} \texttt{refl} & \texttt{if } n = 0 \\ \lambda \, i, (\texttt{hd} \; x, \texttt{hd} \; y), (\lambda \_, \texttt{zip}_\pi \; m \; (\texttt{tl} \; x, \texttt{tl} \; y) \; i), & \texttt{if } n = m + 1 \end{cases} \tag{2.53}$$

$$\texttt{zip}_{lift} \; (x, y) := \texttt{lift}_{\texttt{M}} \; \texttt{zip}_X \; \texttt{zip} \; (x, y) \tag{2.54}$$

We would expect that the two definitions for zip are equal

$$\begin{aligned}
\texttt{transport}_{?}\ a &\equiv \texttt{zip}_{lift}\ a && (2.55)\\
&\equiv \texttt{lift}_{\texttt{M}}\ \texttt{zip}_{X}\ \texttt{zip}_{\pi}\ (x, y) && (2.56)\\
&\equiv (\lambda\, n, \texttt{zip}_{X}\ n\ (x, y)), (\lambda\, n\ i, \texttt{zip}_{\pi}\ n\ (x, y)\ i) && (2.57)
\end{aligned}$$

zero case X

$$\texttt{zip}_{X}\ 0\ (x, y) \equiv \mathbf{1} \qquad (2.58)$$

Sucessor case X

$$\begin{aligned}
\texttt{zip}_{X}\ (m + 1)\ (x, y) &\equiv (\texttt{hd}\ x, \texttt{hd}\ y), (\lambda\, _, \texttt{zip}_{X}\ m\ (\texttt{tl}\ x, \texttt{tl}\ y)) && (2.59)\\
&\equiv (\texttt{hd}\ x, \texttt{hd}\ y), (\lambda\, _, ?\ (\texttt{tl}\ a)) && (2.60)\\
&\equiv (\texttt{hd}\ (\texttt{transport}_{?} a)), (\lambda\, _, \texttt{transport}_{?}(\texttt{tl}\ a)) && (2.61)\\
&\equiv transport_{?} a && (2.62)\\
& && (2.63)
\end{aligned}$$

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

$$\begin{aligned}
&\equiv (), (\lambda\, i, \texttt{zip}_{\pi}\ 0\ (x, y)\ i) && (2.64)\\
&\equiv \mathbf{1}, \texttt{refl} && (2.65)\\
& && (2.66)
\end{aligned}$$

successor case

$$\begin{aligned}
&\equiv (\texttt{zip}_{X}\ (m + 1)\ (x, y)), (\lambda\, i, \texttt{zip}_{\pi}\ (m + 1)\ (x, y)\ i) && (2.67)\\
&\equiv ((\texttt{hd}\ x, \texttt{hd}\ y), (\lambda\, _, \texttt{zip}_{X}\ m\ (\texttt{tl}\ x, \texttt{tl}\ y))), (\lambda\, i, (\texttt{hd}\ x, \texttt{hd}\ y), (\lambda\, _, \texttt{zip}_{\pi}\ m\ (\texttt{tl}\ x, \texttt{tl}\ y)\ i)) && (2.68)\\
& && (2.69)
\end{aligned}$$

2.5.2 Examples of fixed points

We want to define spin, as being the fixed point $\texttt{spin} = \texttt{later spin}$, so that is again a final coalgebra, but of a M-type (which is a final coalgebra)

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

Figure 2.9: TODO

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

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

...