

---

# Higher Order Categorical Semantics

Lasse Letager Hansen, 201912345

---

Master's Thesis, Computer Science

February 17, 2020

Advisor: Bas Spitters



# Abstract

in English...



# Resumé

in Danish...



# Acknowledgments

...

*Lasse Letager Hansen,  
Aarhus, February 17, 2020.*





# Contents

<b>Abstract</b>	<b>iii</b>
<b>Resumé</b>	<b>v</b>
<b>Acknowledgments</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 M-types</b>	<b>3</b>
2.1 Containers / Signatures . . . . .	3
2.2 ITrees as <b>M</b> -types . . . . .	4
2.2.1 Delay Monad . . . . .	4
2.2.2 Tree . . . . .	5
2.3 ITrees . . . . .	6
<b>3 Conclusion</b>	<b>7</b>
<b>Bibliography</b>	<b>9</b>
<b>A The Technical Details</b>	<b>9</b>



# Chapter 1

## Introduction

motivate and explain the problem to be addressed

example of a citation: [?]

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

$$P_S : \mathcal{U} \rightarrow \mathcal{U}$$

$$P(X) := P_S(X) = \sum_{a:A} B(a) \rightarrow X \quad (2.1)$$

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

$$Pf : PX \rightarrow PY$$

$$Pf(a, g) = (a, f \circ g) \quad (2.2)$$

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

$$P_S(X) = \sum_{a:A} \mathbf{1} \rightarrow X \quad (2.3)$$

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

$$P_S(X) = A \times X \quad (2.4)$$

Now we define the coalgebra for this functor with type

$$\text{Coalg}_S = \sum_{C:\mathcal{U}} C \rightarrow PC \quad (2.5)$$

and morphisms

$$\_ \Rightarrow \_ : \text{Coalg}_S \rightarrow \text{Coalg}_S$$

$$(C, \gamma) \Rightarrow (D, \delta) = \sum_{f:C \rightarrow D} \delta \circ f = Pf \circ \gamma \quad (2.6)$$

M-types can now be defined from a container  $S$  as the type  $\mathbf{M}$  such that  $(\mathbf{M}, \text{out} : \mathbf{M} \rightarrow P_S \mathbf{M})$  fulfills the property

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

that is  $\prod_{(C, \gamma) : \text{Coalg}_S} \text{isContr}((C, \gamma) \Rightarrow (\mathbf{M}, \text{out}))$ . We denote this construction of the type  $\mathbf{M}$ , as  $\mathbf{M}(A, B)$  or  $\mathbf{M}S$ .

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

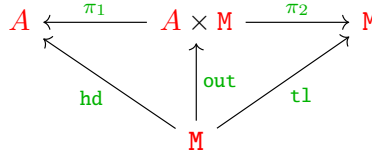


Figure 2.1: M-types of streams

the coalgebra), with inverse  $\text{in} : P_S \mathbf{M} \rightarrow \mathbf{M}$ . We now have a semantic for the rules we would expect for streams, if we let  $\text{cons} = \text{in}$  and  $\text{Stream } A = \mathbf{M}(A, \mathbf{1})$ ,

$$\frac{A : \mathcal{U} \quad s : \text{Stream } A}{\text{hd } s : A} E_{\text{hd}} \quad (2.8)$$

$$\frac{A : \mathcal{U} \quad s : \text{Stream } A}{\text{tl } s : \text{Stream } A} E_{\text{tl}} \quad (2.9)$$

$$\frac{A : \mathcal{U} \quad x : A \quad xs : \text{Stream } A}{\text{cons } x \, xs : \text{Stream } A} I_{\text{cons}} \quad (2.10)$$

## 2.2 ITrees as M-types

### 2.2.1 Delay Monad

We want the following rules for ITrees

$$\frac{r : R}{\text{Ret } r : \text{itree } E \, R} I_{\text{Ret}} \quad (2.11)$$

$$\frac{A : \mathcal{U} \quad a : E \, A \quad f : A \rightarrow \text{itree } E \, R}{\text{Vis } a \, f : \text{itree } E \, R} I_{\text{Vis}}. \quad (2.12)$$

Elimination rules

$$\frac{t : \text{itree } E \, R}{\text{Tau } t : \text{itree } E \, R} E_{\text{Tau}}. \quad (2.13)$$

We start by looking at **itrees** without the **Vis** constructor, this type is also known as the delay monad. We say this type is given by  $S = (\mathbf{1} + R, \lambda\{\text{inl } \_ \rightarrow \mathbf{1} ; \text{inr } \_ \rightarrow R\})$  equal to  $\mathbf{M}S$ , we then get the polynomial functor

$$P_S(X) = \sum_{x : \mathbf{1} + R} \lambda\{\text{inl } \_ \rightarrow \mathbf{1} ; \text{inr } \_ \rightarrow R\} x \rightarrow X \quad (2.14)$$

check  
this  
state-  
ment

This type is equal to the type:

$$P_S(X) = X + R \times (\mathbf{0} \rightarrow X) \quad (2.15)$$

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

$$P_S(X) = X + R \quad (2.16)$$

meaning we get the following diagram. What this diagram says is that we can define the operations

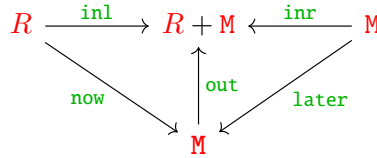


Figure 2.2: Delay monad

**now** and **later** using  $\mathbf{in} = \mathbf{out}^{-1}$  together with the injections **inl** and **inr**.

(Later  
= Tau,  
Ret =  
Now)

### 2.2.2 Tree

Now lets look at the example where we remove the **Tau** constructor. We let

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

This will give us the polynomial functor:

$$P_S(X) = \sum_{x:R + \sum_{A:\mathcal{U}} E A} \lambda\{\mathbf{inl} \_ \rightarrow \mathbf{0} ; \mathbf{inr} (A, e) \rightarrow A\} x \rightarrow X \quad (2.18)$$

which simplifies to

$$P_S(X) = (R \times (\mathbf{0} \rightarrow X)) + (\sum_{A:\mathcal{U}} E A \times (A \rightarrow X)) \quad (2.19)$$

and further

$$P_S(X) = R + \sum_{A:\mathcal{U}} E A \times (A \rightarrow X) \quad (2.20)$$

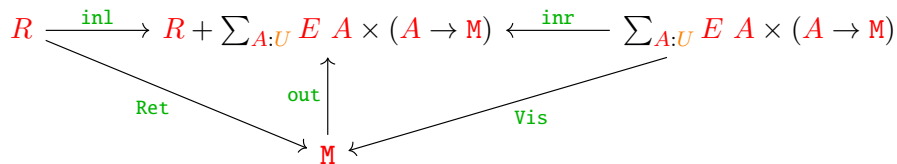


Figure 2.3: TODO: ???

Again we can define **Ret** and **Vis** using the **in** functor.

## 2.3 ITrees

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

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

Then the (reduced) polynomial functor becomes

$$P_S(X) = X + R + \sum_{A:\mathcal{U}} ((E\ A) \times (A \rightarrow X)) \quad (2.22)$$

Giving us the diagram

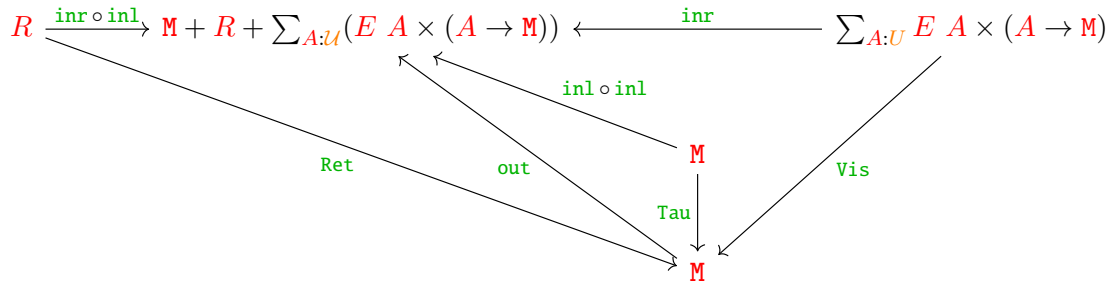


Figure 2.4: TODO: ???



## Chapter 3

# Conclusion

conclude on the problem statement from the introduction



## Appendix A

# The Technical Details

