Higher Order Categorical Semantics

Lasse Letager Hansen, 201912345

Master's Thesis, Computer Science
February 17, 2020
Advisor: Bas Spitters

AARHUS UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Abstract

in English...

Resumé

in Danish…

Acknowledgments

...

Lasse Letager Hansen,
Aarhus, February 17, 2020.

Contents

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

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

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

$$\begin{aligned} Pf &: PX \to PY \\ Pf(a, g) &= (a, f \circ g) \end{aligned} \tag{2.2}$$

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

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

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

$$P_S(X) = A \times X \tag{2.4}$$

Now we define the coalgebra for this functor with type

$$\texttt{Coalg}_S = \sum_{C:\mathcal{U}} C \to PC \tag{2.5}$$

and morphisms

$$\begin{aligned} \_ \Rightarrow \_ &: \texttt{Coalg}_S \to \texttt{Coalg}_S \\ (C, \gamma) \Rightarrow (D, \delta) &= \sum_{f:C \to D} \delta \circ f = Pf \circ \gamma \end{aligned} \tag{2.6}$$

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

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

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

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

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

Figure 2.1: M-types of streams

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

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

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

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

2.2 ITrees as M-types

We want the following rules for ITrees

$$\frac{r : \mathtt{Result}e}{\mathtt{Ret}\ r : \mathtt{itree\ Event\ Result}}\ \mathrm{I}_{\mathtt{Ret}} \tag{2.11}$$

$$\frac{A : \mathtt{Type} \quad a : \mathtt{Event}\ A \quad f : \mathtt{A} \to \triangleright\ \mathtt{itree\ Event\ Result}}{\mathtt{Vis}\ A\ a\ f : \mathtt{itree\ Event\ Result}}\ \mathrm{I}_{\mathtt{Vis}}. \tag{2.12}$$

Elimination rules

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

We start by looking at itree without the (Vis :) constructor

Chapter 3

Guarded Cubical Type Theory (GCTT)

3.1 Marin Löf Type Theory / Intuitionistic Type Theory (MLTT)

3.2 Higher Order Category Theory

3.3 Homotopy Theory (HT)

3.4 Homotopy Type Theory (HoTT)

3.5 Cubical Type Theory (CTT)

3.6 Guarded Type Theory (GTT)

3.7 Guarded Cubical Type Theory (GCTT)

Semantics is based on $\widehat{\mathbb{C} \times \omega}$ The category of cubes $\mathbb{C}$ is the opposite of the Kleisli category of the free De Morgan algebra monad on finite set.

$$(\triangleright(X))(I, n) = \begin{cases} \{\star\} & n = 0 \\ X(I, m) & n = m + 1 \end{cases} \tag{3.1}$$

Chapter 4

Interaction Trees (ITrees)

4.1 M-types

Given a container of type

$$\frac{A : \texttt{Type} \quad a : A \vdash B : \texttt{Type}}{(A, B) : \texttt{Container}} \tag{4.1}$$

then M-types are given as

$$\frac{A : Type \quad B : Type}{M(A, B) : \texttt{Container}} \tag{4.2}$$

where nodes a given by type $a : A$ and subtrees are given as $B(a) \to M(A, B)$ ...

Example: We want to show how streams can be defined using containers / signatures and M-types. We know that streams are the final coalgebra for the functor $(Stream, tail)$

4.2 Definitions

In the following definitions, we will let $\texttt{Event} : \texttt{Type} \to \texttt{Type}$ and $\texttt{Result} : \texttt{Type}$.
Introduction rules

$$\frac{r : \texttt{Result}e}{\texttt{Ret}\ r : \texttt{itree Event Result}}\ \mathrm{I}_{\texttt{Ret}} \tag{4.3}$$

$$\frac{A : \texttt{Type} \quad a : \texttt{Event}\ A \quad f : \texttt{A} \to \triangleright\ \texttt{itree Event Result}}{\texttt{Vis}\ A\ a\ f : \texttt{itree Event Result}}\ \mathrm{I}_{\texttt{Vis}}. \tag{4.4}$$

Elimination rules

$$\frac{t : \triangleright\ \texttt{itree Event Result}}{\texttt{Tau}\ t : \texttt{itree Event Result}}\ \mathrm{E}_{\texttt{Tau}}. \tag{4.5}$$

Where we define:

$$\frac{t : \texttt{itree}}{\texttt{now}\ t : \triangleright\ \texttt{itree}} \tag{4.6}$$

$$\frac{t : \triangleright\texttt{itree}}{\texttt{exit}\ t : \texttt{itree}} \tag{4.7}$$

$$\frac{t : \triangleright\ \texttt{itree}}{\texttt{step}\ t : \triangleright\ \texttt{itree}} \tag{4.8}$$

We now want to define strong bisimulation of ITrees as

$$a \equiv b \to \texttt{Ret}\ a \equiv \texttt{Ret}\ b \tag{4.9}$$
$$a \equiv_{\triangleright} b \to \texttt{Tau}\ (\texttt{next}\ a) \equiv \texttt{Tau}\ (\texttt{next}\ b) \tag{4.10}$$
$$a \equiv b \to a \equiv \texttt{Tau}\ (\texttt{next}\ b) \tag{4.11}$$
$$a \equiv b \to \texttt{Tau}\ (\texttt{next}\ a) \equiv b \tag{4.12}$$
$$(p : A \equiv_{Type} B) \to a \equiv_p b \to f \equiv_{\triangleright ?} g \to \texttt{Vis}\ A\ a\ f \equiv \texttt{Vis}\ B\ b\ g \tag{4.13}$$

We then want to weaken the definition, to say that $\texttt{Tau}\ t \approx t$, that is we want to show

$$\texttt{Tau}\ (\texttt{next}\ t) \equiv t, \tag{4.14}$$

but by (4.12) we just have to show

$$\triangleright\ t \equiv t \tag{4.15}$$

we can take the fixed point of apply **Tau**, giving us the element after some (possibly infinite) operations. This can makes two programs one terminating and another not terminating path equal but not judgmentally equal. If we want to preserve termination sensitivity we have to ???.

We define tau as the fixpoint of $Tau = \texttt{fix}\ x. \triangleright [y \leftarrow x].y$?? (Problem).

$$Tau = \triangleright[y \leftarrow dfix^0\ x. \triangleright [y \leftarrow x].y].y$$

We can build a path between t and **Tau**

Negative variance for **Tau**.

4.3 Category Theory Diagrams

Definition of Vis:

$$\begin{tikzcd}
E\ R \times_{itree\ E\ R} R \arrow[dr, "vis"] & R \arrow[d, "k"] \\
E\ R \arrow[r] & \texttt{itree}\ E\ R
\end{tikzcd}$$

Chapter 5

The Great Ideas

Chapter 6

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

. . .