Algebraic Representation of Higher Order Functions

Dirk Pattinson, Imperial College London

(joint work with Neil Ghani and Peter Hancock)

Uppsala, June 2011

Background

Goal. Representation of *continuous higher order* functions as *data types*

Path.

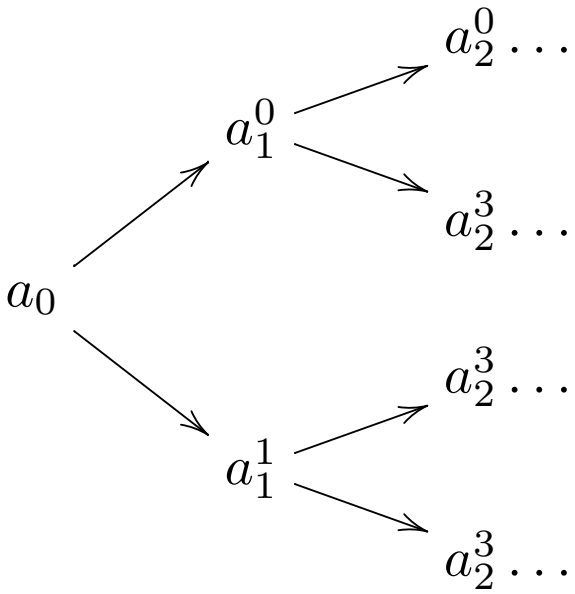
- representation of functions on *data types*
- data types code higher order functions

Infinite Objects: Examples

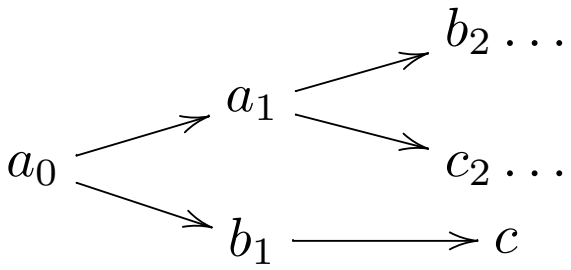
Infinite Streams over A

$$a_0 \to a_1 \to a_2 \to \dots$$

Infinite Binary Trees over A



Signatures (variable branching)



Coalgebraic Representation

Infinite Streams. Coalgebras of type $S \to A \times S$

- every $s \in s$ generates a label $\sigma_0(s) \in A$ and one successor $\sigma_1(s) \in S$

Infinite Binary Trees. Coalgebras of type $S \xrightarrow{\sigma} S \times A \times S$

- every $s \in S$ generates a label $\sigma_0(s)$ and two successors $\sigma_1(s)$ and $\sigma_2(s)$

Signatures. Coalgebras of type $S \xrightarrow{\sigma} A \times S^2 + B \times S + C$

- every $s_0 \in S$ generatates
  - either a label $\sigma_0(s) \in A$ and two successors $\sigma_1(s), \sigma_2(s) \in S$, or
  - a label $\sigma_0(s) \in B$ and one successors $\sigma_1(s) \in S$, or
  - a label $\sigma_0(s) \in C$ and no successors.

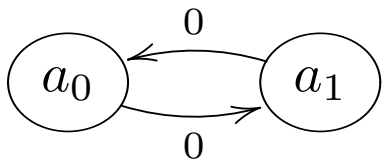
Coalgebras in General. $S \xrightarrow{\sigma} T(S)$ for a functor $T : \mathsf{Set} \to \mathsf{Set}$

Representation of Infinite Objects

Idea. In a coalgebra $S \xrightarrow{\sigma} TS$

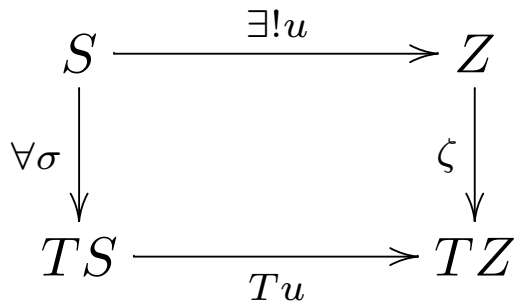
- every $s \in S$ represents "infinite behaviour" – but not uniquely
- not every "type of infinite behaviour" corresponds to some $s \in S$

Example. Take $S = \{a_0, a_1\}$ and $S \to \{0,1\} \times S$



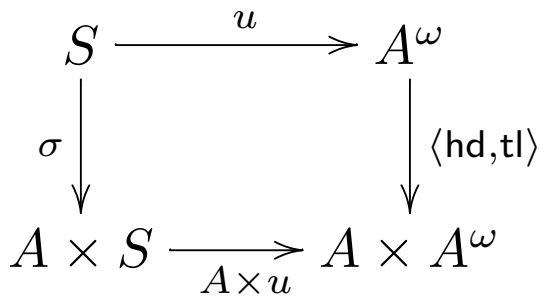
- both a_0 and a_1 represent $(0, 0, \dots)$, but none represents e.g. $(0, 1, \dots)$

Final Coalgebras. (Z, ζ) final if every "behaviour" has a unique representative



Examples

Streams, i.e. T-coalgebras for $TS = A \times S$



where $u(s)_n = \sigma_0 \circ \underbrace{\sigma_1 \circ \cdots \circ \sigma_1}_{n \text{ times}}(s)$ is the unique morphism

- *final coalgebra*: infinite streams, i.e. $(A^\omega, \langle \mathsf{hd}, \mathsf{tl} \rangle)$

Infinite Binary Trees, i.e. coalgebras for $TS = A \times S^2$

- *final coalgebra*: $(\mathsf{Tree}(A), \langle \mathsf{lab}, \mathsf{lft}, \mathsf{rt} \rangle)$ where $\mathsf{lab}/\mathsf{lft}/\mathsf{rt}$ are label and subtrees

Enter Topology . . .

Intuition. Continuous Functions $f : \nu X.TX \to B$

- output $b \in B$ after reading *finite amount* of information in $\nu X.TX$

Example. Infinite Streams, or coalgebraically $\nu X.A \times X \to B$

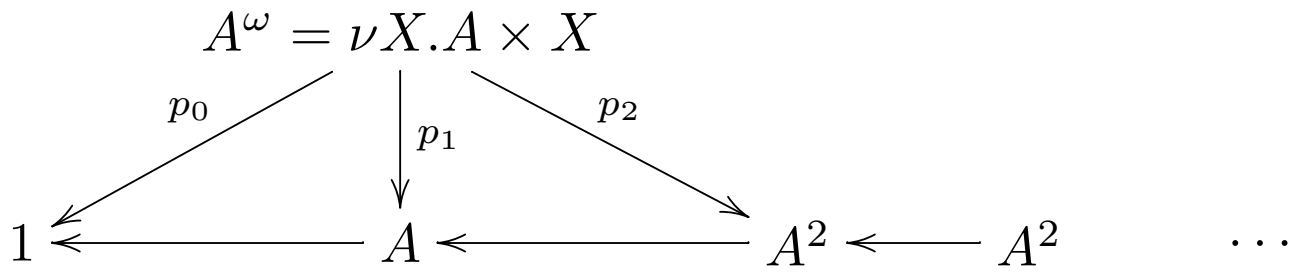
- $f(\alpha)$ depends on finite initial prefix of α

Conceptually. This is the Cantor topology on A^ω (with A discrete)

- generated by $\overline{\alpha} \cdot A^\omega$ where $\overline{\alpha} \in A^*$

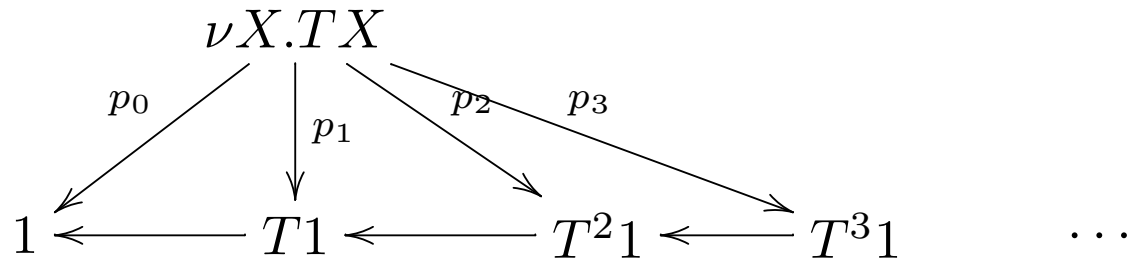
Coalgebraic View

Final Coalgebras arise as *infinite limits*: e.g. streams



Topology generated by $p_i^{-1}(o)$, $o \subseteq A^i$ open

Coalgebraic Generalisation. Suppose $\nu X.TX \xrightarrow{\sigma} T(\nu X.TX)$



where $p_{i+1} = Tp_i \circ \sigma$. Topology generated by $p_i^{-1}(o)$, "$o \subseteq T^i 1$ open"

But what does 'open' mean?

Observation. The forgetful functor

$$U : \mathsf{Top} \to \mathsf{Set}$$

has both a left (discrete topology) and right (indiscrete topology) adjoint.

Corollary. The forgetful functor preserves all limits and colimits:

$$U(\nu X.FX) = \nu X.FX$$

in case F is a polynomial / container.

Topology on (Co)inductive Types. By constructing objects in Top.

Continuous Functions: The Case of Streams

Goal. Characterise continuous functions of type $A^\omega \to B$ with B discrete.

Continuity. (A and B discrete) $f : A^\omega \to B$ is continuous . . .

- iff f locally constant.

$$(\forall (a_0, a_1, \dots) \in A^\omega)(\exists n \in \omega)\ f \text{ constant on } (a_0, a_1, \dots, a_n) \cdot A^\omega$$

- iff f is in the least class C closed under

$$\frac{f \text{ constant}}{C(f)} \qquad \frac{(\forall a \in A) C(f(a : \_))}{C(f)}$$

Proof ($\Leftarrow$) locally constant functions are so closed.

Proof ($\Rightarrow$) classical logic and dependent choice.

Representation of Continuous Stream Functions

Idea. Proofs of Continuity define the *least class* of functions

$$\frac{f \text{ constant}}{C(f)} \qquad \frac{(\forall a \in A) C(f(a:\_))}{C(f)}$$

and can be represented as an *inductive* data type:

$$R = \mu X.B + (A \to X) \cong B + (A \to R)$$

with two constructors: $\texttt{Ret} : B \to R$ and and $\texttt{Rd} : (A \to R) \to R$

from which a continuous function can be extracted:

$$\begin{array}{llll} \texttt{eat}: & \mu X.B + (A \to X) & \to A^\omega & \to B \\ \texttt{eat} & (\texttt{Ret}\, b) & (a:\alpha) & = b \\ \texttt{eat} & (\texttt{Rd}\, f) & (a:\alpha) & = \texttt{eat}(f\ a)\alpha \end{array}$$

Theorem. If $\to_c$ is continuous functions, then $\texttt{eat} : R \to (A^\omega \to_c B)$ is onto.

From Streams to Trees

Goal. Classify functions $\mathsf{Tree}(A) \to_c B$ where $\mathsf{Tree}(A) = \nu X.A \times X^2$

Idea. Let R denote the type of representatives with constructors $\mathtt{Rd}$ and $\mathtt{Ret}$.

$$\begin{array}{llll} \mathsf{eat}: & R & \to \mathsf{Tree}(A) & \to B \\ \mathsf{eat} & (\mathtt{Ret}\ b) & (a,l,r) & = b \\ \mathsf{eat} & (\mathtt{Rd}\ f) & (a,l,r) & = \mathsf{eat}(f\ a)(l,r) \end{array}$$

Observation. $\mathsf{eat}(f\ a) : \mathsf{Tree}(A)^2 \to B$, so $f(a)$ represents $\mathsf{Tree}(A)^2 \to B$

Mathematical Obfuscation. R_n represents $\mathsf{Tree}(A)^n \to B$

$$\begin{array}{llll} \mathsf{eat}_n: & R_n & \to \mathsf{Tree}(A)^n & \to B \\ \mathsf{eat}_n & (\mathtt{Ret}\ b) & (t_1,\ldots,t_n) & = b \\ \mathsf{eat}_n & (\mathtt{Rd}_i\ f) & (t_1,\ldots,t_n) & = \mathsf{eat}_{n+1}(f\ a_i)(t_1,\ldots,t_{i-1},l,r,t_{i+1},\ldots,t_n) \end{array}$$

where l, r are the left/right subtree of t_i. *Constructors.* $\mathtt{Ret}, \mathtt{Rd}_1, \ldots, \mathtt{Rd}_n$

Escaping the Underworld of Indices

Desired (Inductive) Type with constructors $\mathtt{Ret}, \mathtt{Rd}_1, \ldots, \mathtt{Rd}_n$ as above.

$$R_n \cong B + \sum_{i \in n} (A \to R_{n+1})$$

Realisation Mapping.

$$\begin{array}{llll} \mathsf{eat}_n : & R_n & \to \mathsf{Tree}(A)^n & \to B \\ \mathsf{eat}_n & (\mathtt{Ret}\ b) & (t_1, \ldots, t_n) & = b \\ \mathsf{eat}_n & (\mathtt{Rd}_i\ f) & (t_1, \ldots, t_n) & = \mathsf{eat}_{n+1}(f\ a_i)(t_1, \ldots, t_{i-1}, l, r, t_{i+1}, \ldots, t_n) \end{array}$$

Taking Indices Seriously.

$$R(n) \cong B + \sum_{i \in n} (A \to R(n+1))$$

Observation. Now R has type $\mathsf{Set} \to \mathsf{Set}$ – and we want the *least* such

$$R = \mu F : \mathsf{Set} \to \mathsf{Set}. \Lambda I : \mathsf{Set}.\ B + I \times (A \to R(I+1))$$

Conceptual Digression

Streams. Represent $\mathsf{Stream}(A)^S \to B$ by $R(S)$ where

$$R(S) = \mu X.B + S \times (A \to X)$$

- each $R(S)$ is an initial algebra for a functor of type $\mathsf{Set} \to \mathsf{Set}$
- $\mathsf{eat}(S)$ defined by initiality of $R(S)$ – *separately* for all arities

Linearity: Family of Inductive Types

Trees. Represent $\mathsf{Tree}(A)^S \to_c B$ by $R(S)$ where

$$R = \mu F : \mathsf{Set} \to \mathsf{Set}.\Lambda S : \mathsf{Set}.B + S \times (A \to R(X + 1))$$

- R is an initial algebra for a functor $(\mathsf{Set} \to \mathsf{Set}) \to (\mathsf{Set} \to \mathsf{Set})$
- eat is natural and defined by initiality of R – *simultaneously* for all arities

Nonlinearity: Inductive Family of Types

Infinite Objects of Container Type

Container Functors. (Abbot, Altenkirch, Ghani)

$$(S \lhd P)(X) = \sum_{s \in S} X^{P(s)}$$

- S : Set is a set of *shapes*, each of which stores data
- $P : S \to$ Set associates a set of *positions* to every shape

Continuous Functions of type $(\nu X.(S \lhd P)X)^I \to B$

$$R = \mu F : \text{Set} \to \text{Set}.\Lambda I : \text{Set}.B + \sum_{i \in I} \prod_{s \in S} F(I + P(s))$$

Unfolding Isomorphisms.

$$R(I) \cong B + \sum_{i \in I} \prod_{s \in S} F(I + P(s))$$

Intuition.

- if not constant, select tree $(i \in I)$, extract root ($s \in S$), behead and continue

Discrete Codomains are Boring

Next Goal. Represent $A^\omega \to_c B^\omega$

Idea. $f : A^\omega \to B^\omega$ is continuous iff we have an *infinite* proof

$$(R)\frac{\forall a(C(f(a:\_)))}{C(f)} \qquad (W)\frac{C(f)}{C(\lambda\alpha.b : f(\alpha))}$$

where, on any branch in a proof, the right hand rule occurs infinitely often.

Induced Data Type. Wrap up finite occurrences of (R) using a μ

$$R \cong \nu X.\mu Y.B \times X + (A \to Y) \cong B \times R + (A \to R)$$

with constructors $\mathtt{Ret} : B \times R \to R$ and $\mathtt{Rd} : (A \to R) \to R$

Extracted Continuous Function.

$$\begin{array}{llll} \mathsf{eat}: & \nu X.\mu Y.B \times X + (A \to Y) & \to A^\omega & \to B^\omega \\ \mathsf{eat} & (\mathtt{Ret}\,(b, r)) & (a:\alpha) & = b : \mathsf{eat}\ r\ (a:\alpha) \\ \mathsf{eat} & (\mathtt{Rd}\,f) & (a:\alpha) & = \mathsf{eat}\ (f\ a)\ \alpha \end{array}$$

Alternative Computational Representation

We know. Continuous functions of type $A^w \to B$ are represented by

$$A^\omega \to_C B \rightsquigarrow R = \mu X.B + (A \to X)$$

Idea. Re-start the computation as soon as a digit has been produced

$$A^\omega \to_C B^\omega \rightsquigarrow \nu X.\mu Y.B \times X + (A \to Y)$$

with the same computational interpretation

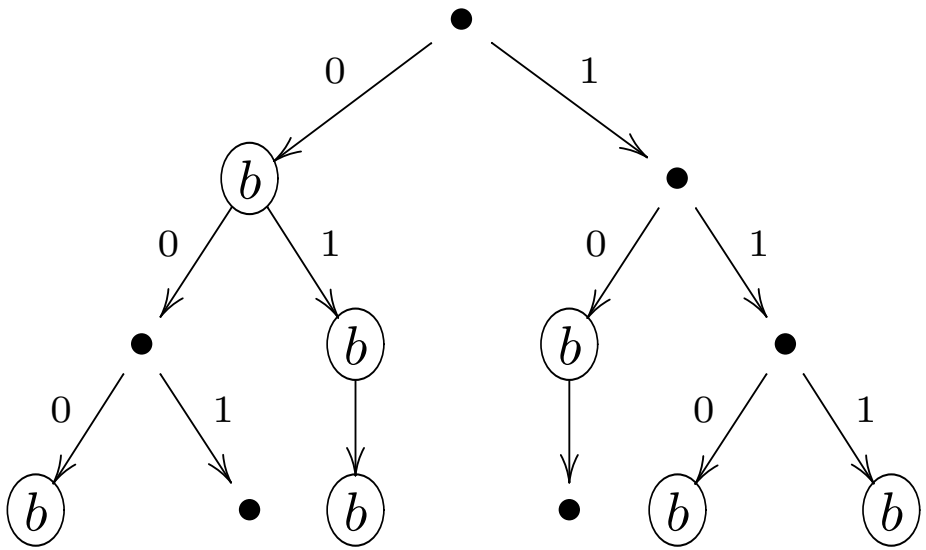
$$\begin{array}{llll} \mathsf{eat}: & \nu X.\mu Y.B \times X + (A \to Y) & \to A^\omega & \to B^\omega \\ \mathsf{eat} & (\mathtt{Ret}\,(b, r)) & (a : \alpha) & = b : \mathsf{eat}\ r\ (a : \alpha) \\ \mathsf{eat} & (\mathtt{Rd}\,f) & (a : \alpha) & = \mathsf{eat}\ (f\ a)\ \alpha \end{array}$$

Note. Occurrence of $B \times X$ suggests that "codomain slots in"

Stream Functions are Trees

Observation. First-Order Functions $A^\omega \to B^\omega$ are *trees*

$$R = \nu X.\mu Y.B \times X + (A \to Y)$$



Initiality guarantees infinitely many labels on every path

General Codomain

More Ambitious Goal. Represent $A^\omega \to \nu X.(S \lhd P)X = \nu X.\sum_{s \in S} X^{P(s)}$

By Analogy.

$$R = \nu X.\mu Y.\sum_{s \in S} X^{P(s)} + (A \to Y) \cong \sum_{s \in S} R^{P(s)} + (A \to R)$$

with constructors $\mathtt{Ret}_s : (P(s) \to R) \to R$ and $\mathtt{Rd} : (A \to R) \to R$

Associated Functional.

$$\begin{array}{llll} \mathsf{eat}: & \nu X.\mu Y.\sum_{s \in S} X^{P(s)} + (A \to Y) & \to A^\omega & \to \nu Z.(P \lhd S)Z \\ \mathsf{eat} & (\mathtt{Ret}_s\,(r_i)) & (a:\alpha) & = (s, (\mathsf{eat}\ r_i\ (a:\alpha))_{i \in P(s)}) \\ \mathsf{eat} & (\mathtt{Rd}\,f) & (a:\alpha) & = \mathsf{eat}\ (f\ a)\ \alpha \end{array}$$

Observation.

- *codomain* just “slots in”, more general *domains* by same recipie

Induction Meets Coinduction

Example. Continuous Stream Functions

$$f : A^\omega \to_c B^\omega$$

are represented by

$$\underbrace{\nu X.\, \overbrace{\mu Y\ B \times X + Y^A}^{T_A(B \times X)}}_{P_A(B)}$$

Lambek's Lemma.

$$P_A(B) = (\nu X)(\mu Y) B \times X + Y^A \cong (\mu Y) B \times P_A(B) + Y^A$$

Pleasant Mathematical Theory.

- supports both *inductive* and *coinductive* definitions and proofs.
- similar for other (co)domains

Inductive Maps Between Coinductive Types

Example. Composition: $P_B(C) \times P_A(B) \to P_A(C)$ where

$$T_A(B) = \mu X.B + (A \to X) \quad \text{and} \quad P_A(B) = \nu X.T_A(B \times X)$$

Operation on *representatives*

$$\begin{array}{ccc} P_B(C) \times P_A(B) & \longrightarrow & P_A(C) \\ \downarrow & & \downarrow \\ (B^\omega \to_c C^\omega) \times (A^\omega \to_c B^\omega) & \longrightarrow & (A^\omega \to_c C^\omega) \end{array}$$

As $P_A(B) \cong T_A(B \times P_A(B))$ is bi-inductive: composition

$$\gamma : S = T_B(C \times P_B C) \times T_A(B \times P_A B) \to T_A(C \times S)$$

is an *inductively defined* map between *coinductive* types

More on Composition

Inductive definition of composition

$$\begin{aligned}
\gamma : S = & T_B(C \times P_B C) \times T_A(B \times P_A B) \to T_A(C \times S) \\
& \langle \, \mathtt{Ret}\, \langle c, p_{bc} \rangle \quad , t_{ab} \quad \rangle \mapsto \mathtt{Ret}\, \langle c, \mathsf{out}\, p_{bc}, t_{ab} \rangle \\
& \langle \, \mathtt{Rd}\, \phi \quad , \mathtt{Ret}\, \langle b, p_{ab} \rangle \quad \rangle \mapsto \gamma \langle \phi\, b, \mathsf{out}\, p_{ab} \rangle \\
& \langle \, t_{bc} \quad , \mathtt{Rd}\, \psi \quad \rangle \mapsto \mathtt{Rd}\, \lambda a. \gamma \langle t_{bc}, \psi\, a \rangle
\end{aligned}$$

whose *coinductive* cousin ($\mathsf{out} : \nu F \to F(\nu F)$)

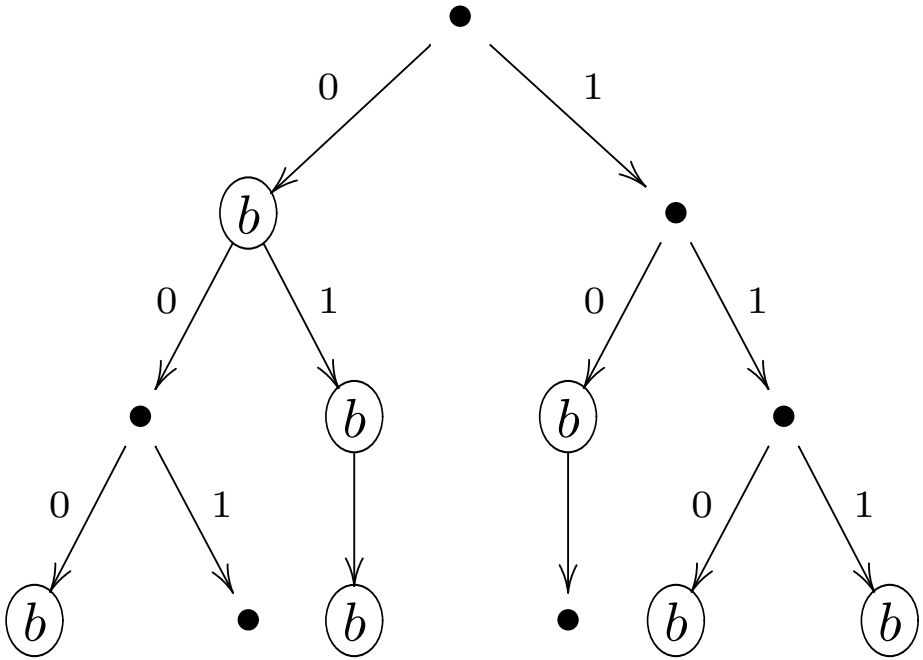
$$\begin{aligned}
\chi : & P_B(C) \times P_A(B) \to P_A(C) \\
& \langle post, \quad pre \rangle \quad \mapsto (\mathsf{unfold}\, \gamma) \langle \mathsf{out}\, post, \mathsf{out}\, pre \rangle
\end{aligned}$$

represents composition.

This is *output centered* – alternatives are possible.

Higher-Order Functions

Observation. First-Order Functions $A^\omega \to B^\omega$ are *trees*



Idea.

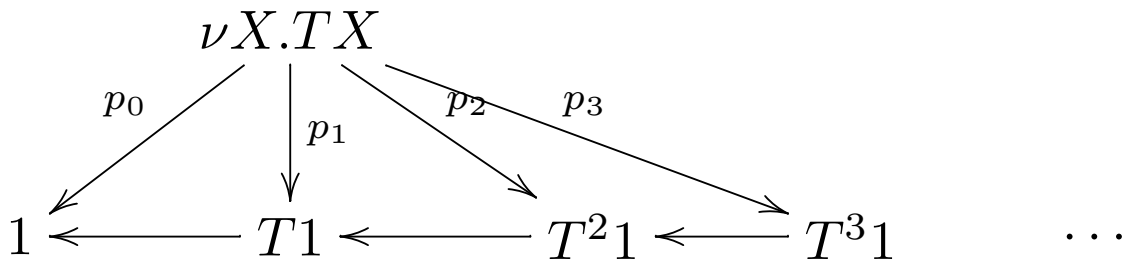
- represent higher-order functions as functions on trees
- *but:* domain doesn't fit into $\nu Z.(S \lhd P)Z = \nu Z.\sum_{s \in S} Z^{P(s)}$

Topological Excursion

Question. What's the natural topology on $A^\omega \to B^\omega$?

Topology on Representatives $R = \nu X.\mu Y.B \times X + (A \to Y)$.

- consider $TX = \mu Y.B \times X + (A \to Y)$ and $\sigma : R \to TR$
- topology given by the inverse limit



where $p_{i+1} = Tp_i \circ \sigma$. Topology generated by $p_i^{-1}(o)$, $o \subseteq T^i 1$ open

Induced Topology on $(A^\omega \to B^\omega)$ is compact-open:

- elements of $T^n 1$ are layers of A-branching trees with labels in B
- single trees define compact-open constraints

Container Magic

Summary so far.

- have representation of functions $\nu Z.(S \lhd P)Z \to X$
- want: representations of $\nu X.\mu Y.(B \times X) + (A \to Y) \to X$

Container Translation. Representations for free – if we solve

$$\mu Y.(B \times X) + (A \to Y) = (S \lhd P)X$$

Theorem. (Abbot/Alternkirch/Ghani) Containers are closed under μ, ν.

More precisely: for every n-ary container

$$C(X_1, \ldots, X_n) = \sum_{s \in S} X_1^{P_1(s)} \times \cdots \times X_n^{P_n(s)}$$

there is an $n-1$-ary container $D(X_1, \ldots, X_{n-1})$ that satisfies

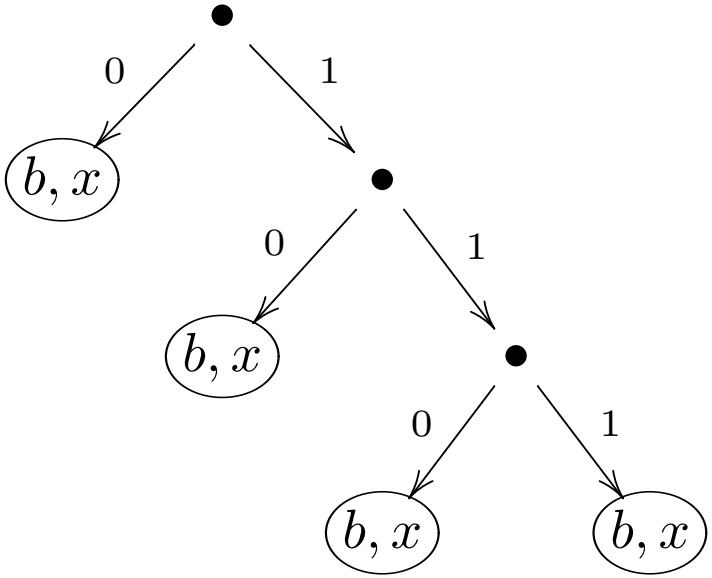
$$D(X_1, \ldots, X_{n-1}) = \mu X_n.C(X_1, \ldots, X_n)$$

Container Translation by Example

Wanted. Solutions of

$$\mu Y.B \times X + (A \to Y) \cong (S \lhd P)X = \sum_{s \in S} X^{P(s)}$$

Observation. We see *trees* with payload at the leaves.



Shapes.

$$S = \mu X.B + (A \to X)$$

Positions.

$$P(s) = \{ \text{ paths in } S \text{ from root to leaves} \}$$

Order-Two Example

Representatives. (Recall: $S = \mu X.B + (A \to X)$ and $P(s) =$ paths)

$$R = \nu F.\mu G.\Lambda I.\, C \times F(I) + I \times \prod_{s \in S} (I + P(s))$$

Unfolding Isomorphisms. (Recall: $R(I)$ represents $(A^\omega \to B^\omega)^I \to C^\omega$)

$$R(I) \cong C \times R(I) + I \times \prod_{s \in S} R(I + P(s))$$

Induced Representation.

$$\begin{array}{llll} \mathsf{eat}(I): & R & \to T^I & \to \nu Z.C \times Z \\ \mathsf{eat}(I) & (\mathtt{Ret}(c, r)) & (\phi) & = c : (\mathsf{eat}\ r\ \phi) \\ \mathsf{eat}(I) & (\mathtt{Rd}\,(i, f)) & (\phi) & = \mathsf{eat}\ (f(\mathsf{root}\,\phi(i)))\ [\phi, \mathsf{debris}(\phi(i))] \end{array}$$

Notation. For $t = (r, d) \in T = \nu X.(S \lhd P)X \cong \sum_{s \in S} T^{P(s)}$

$$\mathsf{root}(r, d) = r \quad \text{and} \quad \mathsf{debris}(r, d) = d$$

Conclusions

Tree Eating.

- linear structures (streams) $\rightsquigarrow$ family of inductive types
- nonlinear structures (trees) $\rightsquigarrow$ inductive families of types
- in both cases: sound and complete representation of continuous functions

Higher Order Functions.

- reducible to tree case – but with coding
- possibly very inefficient in practice – try out

Open Questions.

- more combinators (e.g. buffering, currying)
- concrete case studies – in particular integration
- complexity of (higher order) stream functions?