MGS 2013. Coalgebras and infinite data structures

Venanzio Capretta

Lecture 2: Coalgebras and Bisimulation

The notion of coalgebra can be generalized to any functor on any category.

Assume we work on a category $\mathcal{C}$ (if you don't know category theory, just interpret everything in Set: objects are sets, morphisms are functions between sets)

An endofunctor on $\mathcal{C}$ is a mapping $F : \mathcal{C} \longrightarrow \mathcal{C}$ operating both on objects and morphisms

such that: $F(id_X) = id_{FX}$ preserves identities

$F(g \circ f) = F(g) \circ F(f)$ preserves composition

Examples of functors (on Set):

- Identity functor: $Id(X) = X$, $Id(f) = f$
- $FX = 1 + X$

  If X is a set, FX contains copies of all elements of X ($inr(x)$ for $x : X$) plus a new element $inl(\bullet)$.

  If $f : X \longrightarrow Y$, $Ff : FX \longrightarrow FY$

  $$inl(\bullet) \longmapsto inl(\bullet)$$
  $$inr(x) \longmapsto inr(fx)$$

- Given a fixed set A.

  $FX = A \times X$ is the functor we used for streams

- $FX = A \times X^2$ is the functor we used for binary trees.

- $\mathcal{P}$ the powerset functor

  $\mathcal{P}X$ = subsets of X

  if $f : X \longrightarrow Y$, $\mathcal{P}f : \mathcal{P}X \longrightarrow \mathcal{P}Y$

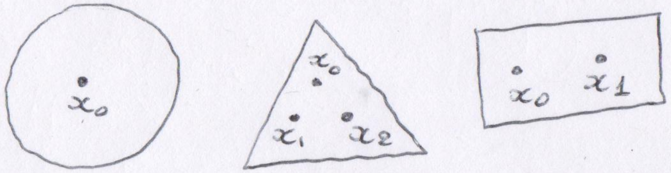
  $$U \longmapsto \{fx \mid x \in U\}$$

Containers

Some functors have a nice intuitive explanation and good formal properties.

An element of FX is thought of as a structure with a shape into which elements of X are inserted at some positions.

For example, we could have three shapes



The container is characterized by the set of shapes:

$$S = \{\bigcirc, \triangle, \square\}$$

And, for every shape, the set of its positions

$$P\bigcirc = \{0\}$$
$$P\triangle = \{0, 1, 2\}$$
$$P\square = \{0, 1\}$$

A container is a pair (S, P) wher

$S : \mathrm{Set}$ ~~posit~~ set of shapes

$P : S \longrightarrow \mathrm{Set}$ set of positions for every shape

The functor associated with a container is

$$F_{(S,P)}\, X = \sum s : S.\ P_s \longrightarrow X$$

$\uparrow$

Σ type: dependent pairs

An element of $F_{(S,P)} X$ is a pair

$\langle s, f \rangle$ where $s : S$

$f : P_s \longrightarrow X$

s is the shape of the element

f assigns an element of X to every position in the shape s.

Many of the functors we have seen are containers

- $FX = 1 + X$

  shapes: $S = \{L, R\}$

  positions: $PL = \emptyset$, $PR = \{\bullet\}$

  no positions: the element $\mathrm{inl}(\bullet)$ doesn't contain any value from X

  one position: the element $\mathrm{inr}(x)$ contains one value $x \in X$

- $FX = A \times X$

  shapes: $S = A$

  positions: $Pa = \{\bullet\}$

- $FX = A \times X^2$

  shapes: $S = A$

  positions: $Pa = \{L, R\}$

The powerset functor $\mathcal{P}$ is not a container

Other examples of containers

- The List functor is a container

  shapes: $S = \mathbb{N}$ (length of the list)

  positions: $Pn = \{0, 1, \ldots, n-1\}$ (indices of the elements)

So $\mathrm{List}\, A = \Sigma n : S.\, Pn \longrightarrow A$

The list $[a_0, a_1, a_2]$ is represented by

$\langle 3, f \rangle$ where $f : \{0,1,2\} \longrightarrow A$
$f\, i = a_i$

- The stream functor $\mathbb{S}$ is a container

  shapes: $S = \{\bullet\}$

  positions: $P\bullet = \mathbb{N}$

Exercise: Find a container representation for the functor $\mathbb{T}$ of infinite binary trees.

Coalgebra

A coalgebra for a functor F is a pair (X, α) where

X is an object

$\alpha : X \longrightarrow FX$ is a morphism

Final coalgebra

A coalgebra (A, α) is final if for every coalgebra (X, γ) there is a unique morphism $\hat{\gamma} : X \longrightarrow A$ making the following diagram commute:

$$
\begin{tikzcd}
A \arrow[r, "\alpha"] & FA \\
X \arrow[u, dashed, "\hat{\gamma}"] \arrow[r, "\gamma"'] & FX \arrow[u, "F\hat{\gamma}"']
\end{tikzcd}
$$

$\hat{\gamma}$ is the anamorphism associated to (X, γ)

If $\hat{\gamma}$ always exists but is not unique we call (A, α) a <u>weakly final coalgebra</u>.

Lambek's Lemma:

Every final coalgebra is an isomorphism.

Proof: Let (A, α) be a final coalgebra

$F\alpha : FA \longrightarrow F^2A$ is also a coalgebra.

So there is a unique anamorphism

$\bar{\alpha} : FA \longrightarrow A$.

We can show that $\bar{\alpha}$ is the inverse of α.

$$
\begin{tikzcd}
A \arrow[r, "\alpha"] & FA \\
FA \arrow[u, dashed, "\bar{\alpha}"] \arrow[r, "F\alpha"] & F^2A \arrow[u, "F\bar{\alpha}"'] \\
A \arrow[u, "\alpha"] \arrow[r, "\alpha"] \arrow[uu, dashed, bend left=60, "id_A"] & FA \arrow[u, "F\alpha"'] \arrow[uu, bend right=60, "Fid_A = id_{FA}"']
\end{tikzcd}
$$

There is a unique anamorphism from (A, α) to itself:

$\bar{\alpha} \circ \alpha = id_A$

Then the commutativity of the upper rectangle gives:

$\alpha \circ \bar{\alpha} = F\bar{\alpha} \circ F\alpha = F(\bar{\alpha} \circ \alpha) = Fid_A = id_{FA}$ ∎

$\mathcal{P}$ doesn't have a final coalgebra

A final coalgebra (A, α) woud be an isomorphism $A \cong \mathcal{P}A$ by Lambek.
But this is impossible by Cantor.

Equality between elements of a final coalgebra.

Final coalgebras model coinductive types whose elements have infinite structure.

How can we prove that two infinite things are equal?

<u>Bisimulation</u>

Coq programming and proving
Streams and Trees

Equality of streams:

Two streams are equal if all their elements are the same.

How can we prove that they are equal without checking an infinite number of equalities?

<u>Bisimulation</u>

A bisimulation of streams is a relation R on $\mathbb{S}_A$ such that

$$s_1 \, R \, s_2 \Rightarrow {}^h s_1 = {}^h s_2 \quad \wedge \quad {}^t s_1 \, R \, {}^t s_2$$

Coinduction Principle:

If R is a bisimulation

$$s_1 \, R \, s_2 \Rightarrow s_1 = s_2$$

Generalization: Bisimulation between coalgebras

Let $(X, \alpha : X \to A \times X)$, $(Y, \beta : Y \to A \times Y)$ be two coalgebras.

A <u>bisimulation</u> between (X, α) and (Y, β) is a relation R between X an Y such that

$$x R y \Rightarrow \pi_1(\alpha x) = \pi_1(\beta y) \wedge \pi_2(\alpha x) \, R \, \pi_2(\beta y)$$

or, with different notation

$$x R y \Rightarrow h_\alpha x = h_\alpha y \wedge t_\alpha x \, R \, t_\beta y$$

Coinduction Principle

If R is a bisimulation between (X, α) and (Y, β)

$$x R y \Rightarrow \hat{\alpha}(x) = \hat{\beta}(y)$$

In intensional type theory (Coq)

coinductive types are <u>weak</u> final coalgebras.

So the Coinduction Principle doesn't hold.

But we can define the appropriate equivalence:

<u>Bisimilarity</u>:

Two streams s_1 and s_2 are bisimilar if there exists a bisimulation R such that $s_1 R s_2$.

In that case we write $s_1 \sim s_2$.

A weakly final (all-encompassing) coalgebra is (strongly) final $\Leftrightarrow$ $\forall x, y. \, x \sim y \to x = y$.

Bisimulation for trees

A bisimulation on T_A is a relation R on T_A such that

$$t_1 \mathrel{R} t_2 \Rightarrow \begin{array}{l} \mathrm{label}(t_1) = \mathrm{label}(t_2) \wedge \\ \mathrm{left}(t_1) \mathrel{R} \mathrm{left}(t_2) \wedge \\ \mathrm{right}(t_1) \mathrel{R} \mathrm{right}(t_2). \end{array}$$

As for streams, we can generalize the definition to any coalgebras.

$t_1 \sim t_2$ if there exists a bisimulation

bisimilarity R such that $t_1 \mathrel{R} t_2$

The coinduction principle holds for trees.

Bisimulations for a generic functor.

In a category $\mathcal{C}$, a relation between objects A and B is represented by a span:

$$\begin{tikzcd} & A \\ R \arrow[ur, "p_1"] \arrow[dr, "p_2"'] & \\ & B \end{tikzcd}$$

idea: R is the set of pairs that are related, p_1 and p_2 give the two related elements.

more general idea:

R is the set of proofs of the relation, p_1 and p_2 give the elements related by the proof.

A relation/span can be lifted to a functor:

$$
\begin{tikzcd}
& A \\
R \arrow[ur, "p_1"] \arrow[dr, "p_2"'] & \\
& B
\end{tikzcd}
\quad \Longrightarrow \quad
\begin{tikzcd}
& FA \\
FR \arrow[ur, "Fp_1"] \arrow[dr, "Fp_2"'] & \\
& FB
\end{tikzcd}
$$

Intuition for containers:

Two elements $x : FA$ and $y : FB$ are related by FR if

- they have the same shape
- components in the same position are related by R.

A bisimulation between coalgebras (A, α), (B, β) is a span (R, p_1, p_2) that "implies" its lifting through the coalgebras.

This means:

There is a coalgebra structure on R $\rho : R \longrightarrow FR$ that makes this diagram commute:

$$
\begin{tikzcd}
& A \arrow[rr, "\alpha"] & & FA \\
R \arrow[ur, "p_1"] \arrow[rr, "\rho"] \arrow[dr, "p_2"'] & & FR \arrow[ur, "Fp_1"'] \arrow[dr, "Fp_2"] & \\
& B \arrow[rr, "\beta"'] & & FB
\end{tikzcd}
$$

So a bisimulation is just a coalgebra (R, ρ) with coalgebra morphisms to (A, α) and (B, β)

Exercise:

Verify that when you instantiate this definition for the functors

$$FX = A \times X \quad \text{and} \quad FX = A \times X^2$$

you get the notion of bisimulation for streams and binary trees

Bisimilarity can itself be define by coinductive means.

For streams:

Bisimilarity ($\sim$) is the relation on $\mathbb{S}_A$ coinductively defined by the rule

$$\frac{a : A \qquad s_1, s_2 : \mathbb{S}_A \qquad s_1 \sim s_2}{(a \lhd s_1) \sim (a \lhd s_2)}$$

A proof of $s_1 \sim s_2$ then requires that $h_{s_1} = h_{s_2}$ and that we have a proof that $t_{s_1} \sim t_{s_2}$ which is in turn constructed by the rule in an infinite regression.

We can define such a proof by a guarded fixpoints, in the same way as we define elements of a coinductive type

Coq allows us to define coinductive predicates and relations and to construct infinite proofs.

Coq definition of bisimilarity and coinductive predicates.