Coalgebra

Luís Soares Barbosa

Universidade do Minho
HASLab HIGH-ASSURANCE SOFTWARE LABORATORY
UNITED NATIONS UNIVERSITY UNU-EGOV

Algebraic and Coalgebraic Methods in Software Development

MAP-i, 23.X.2017

Reactive systems

Systems whose internal configurations are only partially accessible, and are, therefore, characterised by their emergent behaviour which encodes a continued interaction with their environment

> "From being a prescription for how to do something – in Turing's terms a 'list of instructions', software becomes much more akin to a description of behaviour, not only programmed on a computer, but also occurring by hap or design inside or outside it."

[Robin Milner, *Turing Award Lecture*, 1991]

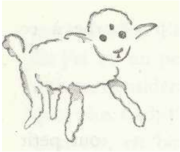
Why coalgebra matters?

Why coalgebra matters?

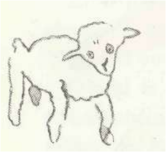
'très malade'

Why coalgebra matters?

'très malade'

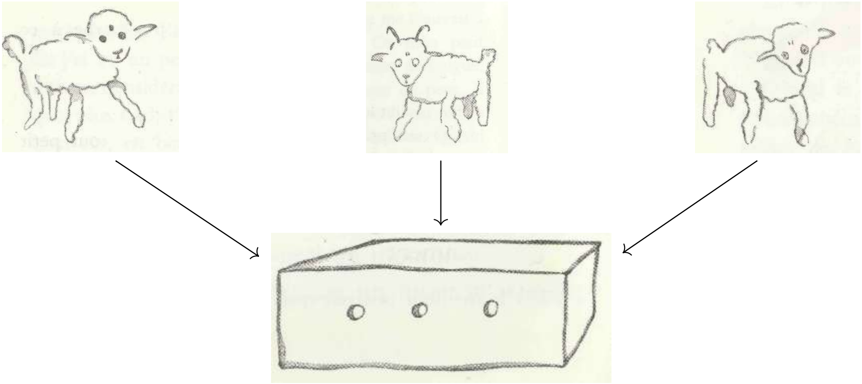


'pas un mouton!"



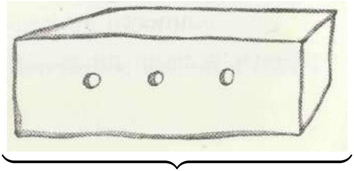
Why coalgebra matters?

'très malade' *'pas un mouton!'* *'trop vieux'*



[Antoine de Saint-Exupéry, *Le Petit Prince*, 1943]

Why coalgebra matters?



an observation shape, a transition type, an interface

Construction vs observation

Models are typically arrows from/to structured objects.
For example, to build an (inductive) data structure one specifies:

an assembly process

$$a : \underbrace{\mathbf{1} + D \times U}_{\text{a tool box}} \longrightarrow U$$

a tool box

the artifact

The structured domain captures a signature of constructors composed additively: $a = [nil, cons]$

Construction vs observation

Reversing the arrow swaps structure from its domain to the codomain, specifying

an observation process

$$p : U \longrightarrow \underbrace{\mathbf{1} + D \times U}$$

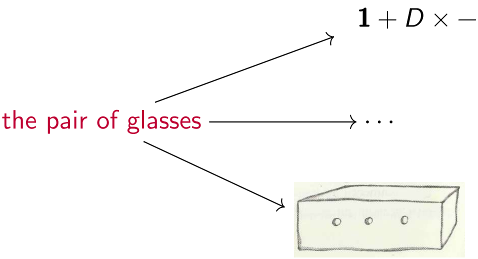
the state space

a pair of glasses

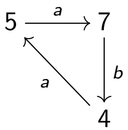
observers are paired in parallel: $p = \underline{*} \lhd \mathit{empty?} \rhd \langle \mathit{head}, \mathit{tail} \rangle$

Construction vs observation

- U can be thought as a state space
- its elements are no longer distinguished by construction, but rather identified when generating the same behaviour;
- finiteness is no longer required;
- the observation shape reveals the type of an underlying transition structure

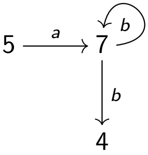


Different glasses capture different transition strucutres



$$p : U \longrightarrow D \times U$$

$$p \;=\; \{5 \mapsto (a,7), 7 \mapsto (b,4), 4 \mapsto (a,7)\}$$



$$p : U \longrightarrow \mathcal{P}(D \times U)$$

$$p \;=\; \{5 \mapsto \{(a,7)\}, 7 \mapsto \{(b,4),(b,7)\}, 4 \mapsto \emptyset\}$$

Different glasses capture different transition strucutres

Recall the models of reactive systems we met so far:

$\alpha : S \longrightarrow \mathcal{P}(S)$	unlabelled TS
$\alpha : S \longrightarrow \mathbb{N} \times S + \mathbf{1}$	partial LTS (generative)
$\alpha : S \longrightarrow (S + \mathbf{1})^{\mathbb{N}}$	partial LTS (reactive)
$\alpha : S \longrightarrow \mathcal{P}(\mathbb{N} \times S)$	non deterministic LTS (generative)
$\alpha : S \longrightarrow \mathcal{P}(S)^{\mathbb{N}}$	non deterministic LTS (reactive)
$\alpha : S \longrightarrow \mathcal{D}S$	simple PTS (Markov chain)
$\alpha : S \longrightarrow \mathcal{D}\mathbb{N} \times S + \mathbf{1}$	generative PTS
$\alpha : S \longrightarrow (\mathcal{D}S + \mathbf{1})^{\mathbb{N}}$	reactive PTS
$\alpha : S \longrightarrow \mathcal{D}S + (\mathbb{N} \times S) + \mathbf{1}$	stratified PTS
$\alpha : S \longrightarrow \mathcal{P}(\mathcal{D}\mathbb{N} \times S)$	strict Segala PTS
$\alpha : S \longrightarrow \mathcal{P}(\mathbb{N} \times \mathcal{D}S)$	simple Segala PTS
$\alpha : S \longrightarrow \mathcal{P}(\mathcal{D}\mathcal{P}(\mathbb{N} \times S))$	Pnueli-Zuck PTS

The general pattern

$$\boxed{\alpha : S \longrightarrow \mathcal{T}\, S}$$

In general, we need

a **lens**: 

an **observation tool**: state space $\xrightarrow{\alpha}$  state space

Models of reactive systems are coalgebras

- $\mathcal{T}$ is the shape of the behaviour (mathematically a functor)
- abstract behavioural structures are (final) coalgebras

The general pattern

$$\boxed{\alpha : S \longrightarrow \mathcal{T}\, S}$$

In general, we need

a lens: 

an observation tool: state space $\xrightarrow{\alpha}$  state space

Models of reactive systems are coalgebras

- $\mathcal{T}$ is the shape of the behaviour (mathematically a functor)
- abstract behavioural structures are (final) coalgebras

What coalgebra brings?

functor	$\mathcal{F}$	transition type / observation shape
coalgebra	$p : U \longrightarrow \mathcal{F}(U)$	generic transition system
morphism	$\begin{array}{ccc} U & \xrightarrow{p} & \mathcal{F}(U) \\ {\scriptstyle h}\uparrow & & \uparrow{\scriptstyle \mathcal{F}(h)} \\ V & \xrightarrow{q} & \mathcal{F}(V) \end{array}$	behaviour preserving map
finality	$\begin{array}{ccc} \Omega_{\mathcal{F}} & \xrightarrow{\omega_{\mathcal{F}}} & \mathcal{F}(\Omega_{\mathcal{F}}) \\ {\scriptstyle [\![p]\!]}\uparrow & & \uparrow{\scriptstyle \mathcal{F}([\![p]\!])} \\ U & \xrightarrow{p} & \mathcal{F}(U) \end{array}$	behaviour

What coalgebra brings?

equivalence	$u \equiv_{\mathcal{F}} v \Leftrightarrow [\![(p)]\!] u = [\![(q)]\!] v$	observational reasoning
operators	the structure of $C_{\mathcal{F}}$	composition
modality	$\Box\phi \ \widehat{=} \ p^{\circ} \cdot \mathcal{F}(\phi) \cdot p$	generic modal logic

- a conceptual tool for the working software engineer to deal with the (emergent) behaviour of computing systems, in a compositional and uniform way;
- an intuitive symmetry made into a mathematical duality.

Automata

state space	U
transition function	$m : U \longrightarrow U$
attribute (or label)	$at : U \longrightarrow B$

i.e.,

$$p = \langle at, m \rangle : U \longrightarrow B \times U$$

Automata

state space	U
transition function	$m : U \longrightarrow U$
attribute (or label)	$at : U \longrightarrow B$

i.e.,

$$p = \langle at, m \rangle : U \longrightarrow B \times U$$

Automata

The behaviour of p at (from) a state $u \in U$ is revealed by successive observations (experiments):

$$\llbracket p \rrbracket\ u \quad = \quad [at\ u, at\ (m\ u), at\ (m\ (m\ u)), \ldots]$$
$$\llbracket p \rrbracket \quad = \quad cons \cdot \langle at, \llbracket p \rrbracket \cdot m \rangle$$

which means that

Automata behaviours are elements of B^ω (*i.e.*, streams)

Automata

The behaviour of p at (from) a state $u \in U$ is revealed by successive observations (experiments):

$$\begin{aligned} [\![p]\!]\, u &= [at\ u, at\ (m\ u), at\ (m\ (m\ u)), \ldots] \\ [\![p]\!] &= cons \cdot \langle at, [\![p]\!] \cdot m \rangle \end{aligned}$$

which means that

Automata behaviours are elements of B^{ω} (*i.e.*, streams)

Automata

Example: A twist automata

state space	$U = \mathbb{N} \times \mathbb{N}$
transition function	$m\,(n, n') \;=\; (n', n)$
attribute	$at\,(n, n') \;=\; n$

i.e.,

$$twist \;=\; \langle \pi_1, s \rangle$$

Automata

Example: A stream automata

state space	$U = B^{\omega}$
transition function	$m\, s \;=\; tail\, s$
attribute	$at\, s \;=\; head\, s$

i.e.,

$$\omega \;=\; \langle hd, tl \rangle$$

Automata behaviours form themselves an automata

Automata

Example: A stream automata

state space	$U = B^\omega$
transition function	$m\,s \;=\; tail\,s$
attribute	$at\,s \;=\; head\,s$

i.e.,

$$\omega \;=\; \langle hd, tl \rangle$$

Automata behaviours form themselves an automata

Automata morphisms

A morphism

$$h : p \longrightarrow q$$

where

$$\begin{aligned} p &= \langle at, m \rangle : U \longrightarrow B \times U \\ q &= \langle at', m' \rangle : V \longrightarrow B \times V \end{aligned}$$

is a function $h : U \longrightarrow V$ such that

$$\begin{array}{ccc} U & \xrightarrow{p} & B \times U \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle id \times h} \\ V & \xrightarrow{q} & B \times V \end{array}$$

i.e.,

$$at = at' \cdot h \quad \text{and} \quad h \cdot m = m' \cdot h$$

Behaviour as a morphism

Th: Behaviour $[\![p]\!]$ is an automata morphism from p to ω

because

$$
\begin{aligned}
& at = hd \cdot cons \cdot \langle at, [\![p]\!] \cdot m \rangle \\
= & \quad \{ \; hd \cdot cons = \pi_1 \; \} \\
& at = \pi_1 \cdot \langle at, [\![p]\!] \cdot m \rangle \\
= & \quad \{ \; \times \text{ cancellation } \} \\
& at = at
\end{aligned}
$$

and

$$
\begin{aligned}
& [\![p]\!] \cdot m = tl \cdot cons \cdot \langle at, [\![p]\!] \cdot m \rangle \\
= & \quad \{ \; tl \cdot cons = \pi_2 \; \} \\
& [\![p]\!] \cdot m = \pi_2 \cdot \langle at, [\![p]\!] \cdot m \rangle \\
= & \quad \{ \; \times \text{ cancellation } \} \\
& [\![p]\!] \cdot m = [\![p]\!] \cdot m
\end{aligned}
$$

The final automata

Automata behaviours form themselves an automata ω with a particular characteristic: from each other automata p there is one and only one morphism

$$[\![p]\!] : p \longrightarrow \omega$$

Automata ω is the final automata, i.e., the universal one in the category of automata and automata morphisms.

Question

How to reason about automata behaviours?

Induction & Coinduction

Reasoning about B^*

$$len(map\, f\, l) = len\, l$$

where functions are defined inductively by their effect on B^* constructors

$$\begin{aligned} len\, [] &= 0 \\ len(h : t) &= 1 + len\, t \end{aligned}$$

$$\begin{aligned} map\, f\, [] &= [] \\ map\, f\, (h : t) &= f(h) : map\, f\, t \end{aligned}$$

These equations are indeed a **functional program** ...

Induction & Coinduction

Proof (by structural induction).

Base case is trivial. Then,

$$
\begin{aligned}
& len(map\, f\, (h : t)) \\
= & \quad \{\ map\, f \text{ definition } \} \\
& len(f\,(h) : map\, f\ t) \\
= & \quad \{\ len \text{ definition } \} \\
& 1 + len(map\, f\ t) \\
= & \quad \{\ \text{induction hypothesis } \} \\
& 1 + len\, t \\
= & \quad \{\ len \text{ definition } \} \\
& len(h : t)
\end{aligned}
$$

Induction & Coinduction

Inductive reasoning requires that, by repeatedly unfolding the definition, arguments become smaller, *i.e.*, closer to the elementary constructors

... but what happens if this unfolding process does not terminate?

Induction & Coinduction

Consider

$$\begin{aligned} map\, f\, (h : t) &= (f\ h) : map\, f\ t \\ gen\, f\, x &= x : gen\, f\, (f\, x) \end{aligned}$$

- definition unfolding does not terminate but ...
- ... reveals longer and longer prefixes of the result: every element in the result gets uniquely determined along this process

Strategy

To reason about circular definitions over infinite structures, our attention shifts from argument's structural shrinking to the progressive construction of the result which becomes richer in informational contents.

Induction & Coinduction

Reasoning about B^ω: the local view

Two streams s and r are observationally the same if

- they have identical head observations: $head\ \ s = head\ \ r$,
- and their tails — $tail\ \ s$ and $tail\ \ r$ — support a similar verification.

Relation $R : B^\omega \longrightarrow B^\omega$ is a (stream) bisimulation iff

$$\langle x, y \rangle \in R \Rightarrow head\ \ x = head\ \ y \wedge \langle tail\ \ x, tail\ \ y \rangle \in R$$

(i.e., R is closed under the computational dynamics)

Induction & Coinduction

Reasoning about B^ω: the local view

Two streams s and r are observationally the same if

- they have identical head observations: $head\ \ s = head\ \ r$,
- and their tails — $tail\ \ s$ and $tail\ \ r$ — support a similar verification.

Relation $R : B^\omega \longrightarrow B^\omega$ is a (stream) bisimulation iff

$$\langle x, y\rangle \in R \ \Rightarrow\ head\ \ x = head\ \ y\ \wedge\ \langle tail\ \ x, tail\ \ y\rangle \in R$$

(i.e., R is closed under the computational dynamics)

Induction & Coinduction

Coinduction as a proof principle:

- a systematic way of strengthening the statement to prove: from equality $s = r$ to a larger set R which contains pair $\langle s, r\rangle$
- ensuring that such a set is a bisimulation, *i.e.*, the closure of the original set under taking derivatives
- moreover, as a proof principle, it generalises from streams to a large class of behaviour types

Induction & Coinduction

$$map_f \cdot gen_f \;=\; gen_{f.f}$$

Check that R below is a bisimulation

$$R \;=\; \{\langle map\, f\, (gen\, f\, x)\,,\, gen\, f\, (f\, x)\rangle |\; x \in ..., f \in ...\}$$

- $head\;(map\, f\, (gen\, f\, x)) \;=\; f\, x \;=\; head\;(gen\, f\, (f\, x))$
- $tail\;(map\, f\, (gen\, f\, x)) \;=\; map\, f\;\; tail\;\; (gen\, f\, x) \;=\; map\, f\, (gen\, f\, f\, x)\;\; tail\;\; (gen\, f\, (f\, x)) \;=\; gen\, f\, (f\, f\, x)$. Thus,

$$\langle tail\;(map\, f\, (gen\, f\, x)), tail\;(gen\, f\, (f\, x))\rangle \in R$$

Remark:
In general, however, much larger relations have to be considered and the construction of bisimulations is not trivial

Induction & Coinduction

$$map_f \cdot gen_f \;=\; gen_{f.f}$$

Check that R below is a bisimulation

$$R \;=\; \{\langle map\, f\, (gen\, f\, x)\,,\, gen\, f\, (f\, x)\rangle |\, x \in ..., f \in ...\}$$

- $head\;(map\, f\,(gen\, f\, x)) \;=\; f\, x \;=\; head\;(gen\, f\,(f\, x))$
- $tail\;(map\, f\,(gen\, f\, x)) \;=\; map\, f\;\; tail\;\;(gen\, f\, x) \;=\; map\, f\,(gen\, f\; f\, x)\;\; tail\;(gen\, f\,(f\, x)) \;=\; gen\, f\,(f\; f\, x)$. Thus,

$$\langle tail\;(map\, f\,(gen\, f\, x)), tail\;(gen\, f\,(f\, x))\rangle \in R$$

Remark:
In general, however, much larger relations have to be considered and the construction of bisimulations is not trivial

Example: $\mathcal{F}\,X = X^A \times B$

Objects are Moore machines

$$p \;\widehat{=}\; \langle \overline{m}, at \rangle : U \longrightarrow U^A \times B$$

$$\begin{array}{ccc} U & \xrightarrow{\;p\;} & U^A \times B \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle h^A \times id} \\ V & \xrightarrow{\;p'\;} & V^A \times B \end{array} \quad \text{boiling down to} \quad \begin{array}{ccc} U & \xrightarrow{\;at\;} & B \\ {\scriptstyle h}\downarrow & & \downarrow{\scriptstyle id} \\ V & \xrightarrow{\;at'\;} & B \end{array} \qquad \begin{array}{ccc} U \times A & \xrightarrow{\;m\;} & U \\ {\scriptstyle h \times id}\downarrow & & \downarrow{\scriptstyle h} \\ V \times A & \xrightarrow{\;m'\;} & V \end{array}$$

$$m' \cdot (h \times id) \;=\; h \cdot m \;\;\wedge\;\; at' \cdot h \;=\; at$$

Example: Moore behaviours

Triggered by input sequences $s = [a_0, a_1, \ldots]$ in A^*, the behaviour of p is revealed by successive observations:

$$at\ u,\ at\ (\overline{m}\ u\ a_0),\ at\ (\overline{m}\ (\overline{m}\ u\ a_0)\ a_1), \ldots$$

$$[\![p]\!]\, u\, \underline{nil} \,\widehat{=}\, at\ u \quad \text{and} \quad [\![p]\!]\, u\, (cons\, \langle a, t\rangle) \,\widehat{=}\, [\![p]\!]\, (m\, \langle u, a\rangle)\, t$$

behaviours organise themselves into a Moore machine over B^{A^*}:

$$\omega_{\mathcal{F}} \,\widehat{=}\, \langle \overline{m}_\omega, at_\omega \rangle : B^{A^*} \longrightarrow (B^{A^*})^A \times B$$

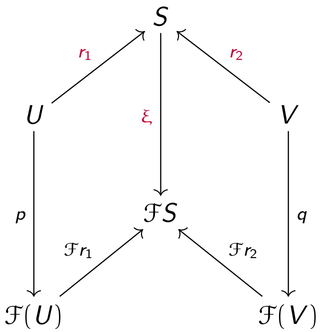
where

$at_\omega\ f$	$\widehat{=}$	$f\ \ nil$	the attribute before any input
$\overline{m}_\omega\ f\ a$	$\widehat{=}$	$\lambda\, s\, .\ \ f(cons\langle a, s\rangle)$	input determines subsequent evolution

Observational equivalence

$$u \equiv_{\mathcal{F}} v \quad \Leftrightarrow \quad [\![p]\!] u = [\![q]\!] v$$

In general, seek for a cocongruence, i.e.



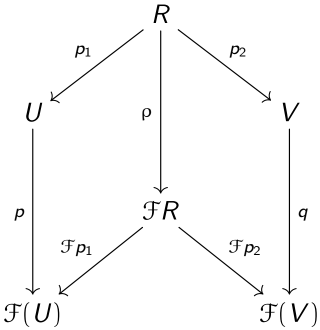
even if $\mathcal{F}$ does not admit a final coalgebra $\omega_{\mathcal{F}}$

▷ Looking for duals: congruent terms vs cocongruent behaviours

Bisimulation

$$u \sim_{\mathcal{F}} v \quad \Leftrightarrow \quad \exists_{\text{bisimulation } R} \, . \quad u = p_1 t \; \text{ and } \; v = p_2 t \; \text{ for a } \; t \in R$$

Bisimulation: a (monic) span $p \xleftarrow{p_1} \rho \xrightarrow{p_2} q$ in $C_{\mathcal{F}}$



▷ *analogue* but not *dual* to a compatible relation

Bisimilarity vs observational equivalence

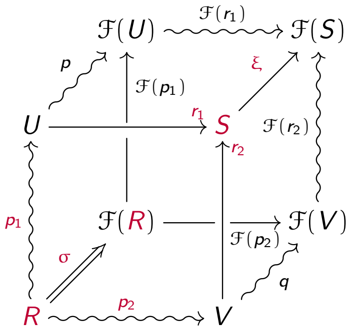
An example: bisimulation for Moore machines

$$\langle u, v\rangle \in R \;\Rightarrow\; at_p\, u = at_q\, v \;\text{ and }\; \langle \overline{m}_p\, u\, a, \overline{m}_q\, v\, a\rangle \in R, \;\text{ for all } a \in A\,.$$

- Bisimilarity is amenable to automation; efficient, iterative algorithms.
- Provides a a technique for coinductive proofs: from argument's structural shrinking to the progressive construction of the behaviour which becomes richer in informational contents.

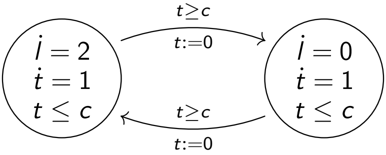
▷ $\sim_{\mathcal{F}}$ and $\equiv_{\mathcal{F}}$ coincide for most functors of interest in SE

Bisimilarity vs observational equivalence



▷ No need for σ to be unique: $\mathcal{F}$ must only preserve weak pullbacks.

Illustration: Hybrid automata



. . . models capturing the interaction of discrete (computational) systems with continuous (physical) processes . . .

$$p : U \longrightarrow \mathcal{G}(U) \times \mathcal{H}(O)$$

where $\mathcal{H}$ captures the continuous evolution of a quantity O over time.

$\mathcal{H}(X) \;\widehat{=}\; \{\langle f, d\rangle \in X^{\mathsf{T}} \times [0,\infty] \mid f \cdot \lambda_d = f\}$ and $\mathcal{H}(h) \;\widehat{=}\; h^{\mathsf{T}} \times id$

▷ Renato Neves's forthcoming PhD thesis

Illustration: Hybrid automata

$$b : V \times P \longrightarrow (V \times P) \times \mathcal{H}(P) \mathrel{\widehat{=}} \langle b_d, b_c \rangle$$

$$b_d \, \langle v, p \rangle \mathrel{\widehat{=}} \langle vel_g \langle v, zpos_g \langle v, p \rangle \rangle \times -0.5,\ 0 \rangle$$
$$b_c \mathrel{\widehat{=}} \langle pos_g (v, p), zpos_g \rangle$$

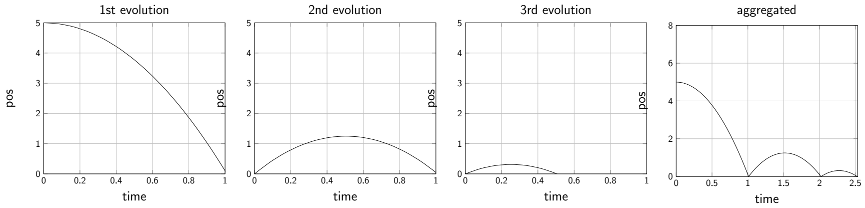


Illustration: Hybrid automata

$$p : U \longrightarrow \mathcal{G}(U) \times \mathcal{H}(O)$$

coalgebra p	functor $\mathcal{G}$
$U \to (U \times \mathcal{H}O)^I$	$Id\, X = X$
$U \to (\Delta U \times \mathcal{H}O)$	$\Delta\, X = X \times X$
$U \to (\mathcal{P}U \times \mathcal{H}O)$	$\mathcal{P}\, X = \{A \subseteq X\}$
$U \to (\mathcal{D}U \times \mathcal{H}O)$	$\mathcal{D}\, X = \{\mu \in [0,1]^X \mid \mu[X] = 1\}$
$U \to (\mathcal{P}\mathcal{D}U \times \mathcal{H}O)$	$\mathcal{P}\mathcal{D}$

- ‘black-box’ view: discrete transitions are kept internal; continuous evolutions make up the observable behaviour.

Properties

Modal assertions, *i.e.* properties to be interpreted across a transition system capturing its dynamics, are pervasive in Software Engineering.

Modalities in Coalgebra also acquire a shape

i.e. their definition becomes parametric on whatever type of behaviour seems appropriate for addressing the problem at hand.

Example: invariants

Predicates preserved along the system's evolution:

$$\forall_{u \in U} \,.\; u \; \phi \; u \; \Rightarrow \; (p \; u) \, \mathcal{F}(\phi) \, (p \; u)$$

which, by eliminating variables, is equivalent to

$$\phi \; \subseteq \; \underbrace{p^{\circ} \cdot \mathcal{F}(\phi) \cdot p}_{\Box \phi}$$

▷ regarding ϕ as a coreflexive relation and $\mathcal{F}$ as a relator

$\Box$ acquires a shape

Example: $\mathcal{F}(X) = \mathcal{P}(X)$

$$\Box\phi \;=\; \{u \in U|\; (p\; u)\; \mathcal{P}(\phi)\; (p\; u)\} \;=\; \{u \in U|\; p\; u \subseteq \phi\}$$

i.e. the standard interpretation of the $\Box$ modality in Kripke semantics

Example: $\mathcal{F}(X) = \mathbf{1} + X$

$$\Box\phi \;=\; \{u \in U|\; p\; u = \iota_2\; u' \;\Rightarrow\; u' \in \phi\}$$

Going generic: Coalgebraic logic

$\Box$ is relative to the 'global' dynamics of p.

However, depending on applications one may be interested in other types of modalities:

- For $\mathcal{F}(X) = A \times X \times X$, follow right or left successors.
- For $\mathcal{F}(X) = (\mathcal{P}X)^A$, define one 'box' operator per each action $a \in A$.

Going generic

$\mathcal{F}$-coalgebras generate modalities by predicate lifting

$$\Box \ \widehat{=}\ \mathbf{2}^U \xrightarrow{\gamma_U} \mathbf{2}^{\mathcal{F}(U)} \xrightarrow{p^{-1}} \mathbf{2}^U$$

$$\Box \phi \ = \ \{u \in U |\ p\ u \in \gamma_U \phi\}$$

Example

- A family $\{\gamma^a : \mathbf{2}^- \Longrightarrow \mathbf{2}^{\mathcal{P}(-)^A} |\ a \in A\}$ of predicate liftings

$$\gamma^a_U\ \phi\ \widehat{=}\ \{s \in \mathcal{P}(U)^A |\ s\ a \subseteq \phi\}$$

induces the indexed modalities of Hennessy–Milner logic:

$$[a]\phi = \{u \in U |\ (p\ u)\ a \subseteq \phi\}$$

Why coalgebra matters?

The message

Coalgebra is the mathematics for dynamical, state-based systems

Why coalgebra matters?

The message

Coalgebra is the mathematics for dynamical, state-based systems

The method

From a suitable characterisation of the type of a system's dynamics, canonical notions of behaviour, observational reasoning (equational and inequational), composition and modality can be derived in a uniform way.

Why coalgebra matters?

The message

Coalgebra is the mathematics for dynamical, state-based systems

The method

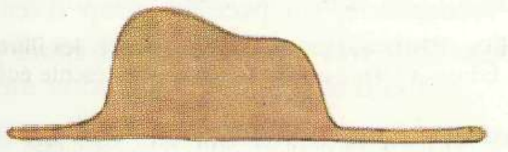
From a suitable characterisation of the type of a system's dynamics, canonical notions of behaviour, observational reasoning (equational and inequational), composition and modality can be derived in a uniform way.

The crucial design choice

The type of a system's dynamics is the pair of glasses through which it is observed

Which pair of glasses?

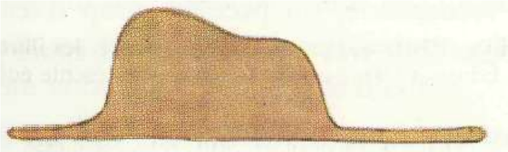
From the coarsest . . .



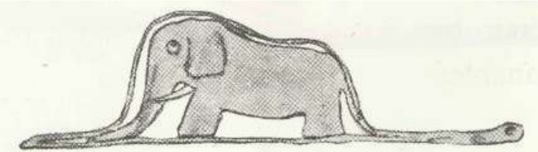
[Antoine de Saint-Exupéry, *Le Petit Prince*, 1943)]

Which pair of glasses?

From the coarsest . . .



. . . to the most detailed



[Antoine de Saint-Exupéry, *Le Petit Prince*, 1943)]

Coalgebra for the working software engineer

$$\textbf{Software Engineering} \left\{ \begin{array}{l} \text{modelling complex systems} \\ \text{architecting their composition} \\ \text{reasoning about their behaviour} \end{array} \right\} \textbf{Coalgebra}$$

Epilogue

$$\textbf{Engineering} \left\{ \text{HOW} \ldots\ldots \text{WHAT} \right\} \textbf{Mathematics}$$

Doing Software Engineering in lighter, more informal ways, is like talking about electricity without using calculus: Good enough to replace a fuse, not enough to design an amplifier.

[attributed to Vlad Patryshev]