Coalgebras and their Modal Logics: Polynomial Functors and Beyond

Part 1: Coalgebraic Modelling of Systems

Helle Hvid Hansen

Workshop on Polynomial Functors, 16 March 2021

University of Groningen, NL

Introduction

Coalgebra: Background and Motivation

Origins and general references

- Non-wellfounded set theory (Aczel'88, Barwise-Moss'96).
  Solving systems of equations, self-referentiality.
- 1990s in Comp.Sci.: systems and data structures as coalgebras.
- J. Rutten. Universal Coalgebra, a theory of systems, 2000.
- B. Jacobs. Introduction to Coalgebra, CUP 2016.

Coalgebra: Background and Motivation

Origins and general references

- Non-wellfounded set theory (Aczel'88, Barwise-Moss'96). Solving systems of equations, self-referentiality.
- 1990s in Comp.Sci.: systems and data structures as coalgebras.
- J. Rutten. Universal Coalgebra, a theory of systems, 2000.
- B. Jacobs. Introduction to Coalgebra, CUP 2016.

Program semantics

- formal descriptions of data and program behaviours
- reasoning (what are useful principles?)

Formal verification

- does system behave as intended?
- we need: formal models of system behaviours
- we need: formal languages for specifying properties
- trade-off: expressiveness & tractability

Overview of Today

Part 1:

1. Introduction

2. Systems as Coalgebras

3. Bisimulation, Coinduction, Behavioural Equivalence

4. Application: Language Semantics of Automata with Branching

Remarks:

- focus on applications and examples in Set.
- only basic category theory.
- polynomial functors: special case
- some pointers to further reading (necessarily incomplete)

Systems as Coalgebras

Big Picture: Algebra vs Coalgebra

Algebra

- construction
- (necessarily) well-founded structures
- induction
- congruence
- compositionality
- universal algebra
- parametric in signature and equations

Coalgebra

- destruction/observation
- (possibly) non-well-founded structures
- coinduction
- bisimulation
- abstraction
- universal coalgebra
- parametric in transitions and observations

cf. [Jacobs & Rutten,1997]

Category of F-Coalgebras

Def. Given $F\colon \mathsf{C} \to \mathsf{C}$, the category $\mathsf{Coalg}(F)$ consists of

- Objects: F-coalgebras $\gamma\colon X \to F(X)$.
- Arrows: F-coalgebra morphisms:

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ {\scriptstyle\gamma}\big\downarrow & & \big\downarrow{\scriptstyle\delta} \\ F(X) & \xrightarrow{F(f)} & F(Y) \end{array}$$

We have:

- general notions of subobject, quotient, …
- all colimits in $\mathsf{Coalg}(F)$ constructed as in C
- limits in $\mathsf{Coalg}(F)$ are non-trivial,
- for $F\colon Set \to Set$, $\mathsf{Coalg}(F)$ is complete and cocomplete

Example: Deterministic systems with output

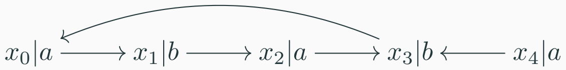
- A deterministic system with output in a set B:

  transition map $t\colon X \to X$

  output map $o\colon X \to B$

  combined $\langle o, t \rangle : X \to B \times X, \quad \langle o, t \rangle(x) = \langle o(x), t(x) \rangle$

  i.e., coalgebra for Set-functor $F(X) = B \times X$.
- Example:

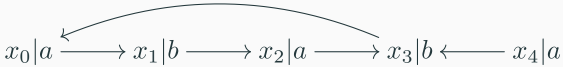


  where $x|a \longrightarrow y|b$ means $o(x) = a$ and $t(x) = y$.

Example: Deterministic systems with output

- A deterministic system with output in a set B:
  - transition map $t\colon X \to X$
  - output map $o\colon X \to B$
  - combined $\langle o, t\rangle : X \to B \times X, \quad \langle o, t\rangle(x) = \langle o(x), t(x)\rangle$

  i.e., coalgebra for Set-functor $F(X) = B \times X$.
- Example:



  where $x|a \longrightarrow y|b$ means $o(x) = a$ and $t(x) = y$.
- Observable behaviour is a stream (infinite sequence):

$$\llbracket x \rrbracket = (o(x), o(t(x)), o(t^2(x)), \ldots)$$

$$\begin{aligned}
\llbracket x_0 \rrbracket &= (a, b, a, b, a, b, a, b, \ldots) = (ab)^\omega \\
\llbracket x_1 \rrbracket &= (b, a, b, a, b, a, b, a, \ldots) = (ba)^\omega \\
\llbracket x_2 \rrbracket &= (a, b, a, b, a, b, a, b, \ldots) = (ab)^\omega
\end{aligned}$$

The Final Deterministic System of Streams

Streams over B: $B^\omega = \{\sigma \colon \mathbb{N} \to B\}$. Write: $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots)$

- "head": $hd(\sigma) = \sigma(0)$, "tail": $tl(\sigma) = (\sigma(1), \sigma(2), \ldots)$
- Deterministic system of streams: $\langle hd, tl \rangle \colon B^\omega \to B \times B^\omega$

The Final Deterministic System of Streams

Streams over B: $B^\omega = \{\sigma\colon \mathbb{N} \to B\}$. Write: $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots)$

- "head": $hd(\sigma) = \sigma(0)$, "tail": $tl(\sigma) = (\sigma(1), \sigma(2), \ldots)$
- Deterministic system of streams: $\langle hd, tl\rangle\colon B^\omega \to B \times B^\omega$

Universal property of $(B^\omega, \langle hd, tl\rangle)$**:**
For all determ. systems $(X, \langle o, t\rangle)$ there is a unique map $[\![-]\!]\colon X \to B^\omega$

such that

$$\begin{array}{rcl} hd([\![x]\!]) & = & o(x) \\ tl([\![x]\!]) & = & [\![t(x)]\!] \end{array}$$

i.e., the following diagram commutes

$$\begin{array}{ccc} X & \xrightarrow{[\![-]\!]} & B^\omega \\ {\scriptstyle \langle o,t\rangle}\downarrow & & \downarrow{\scriptstyle \langle hd,tl\rangle} \\ B \times X & \xrightarrow{id_B \times [\![-]\!]} & B \times B^\omega \end{array}$$

(that is, $[\![-]\!]$ is a morphism of deterministic systems)

The Final Deterministic System of Streams

Streams over B: $B^\omega = \{\sigma\colon \mathbb{N} \to B\}$. Write: $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots)$

- "head": $hd(\sigma) = \sigma(0)$,   "tail": $tl(\sigma) = (\sigma(1), \sigma(2), \ldots)$
- Deterministic system of streams: $\langle hd, tl\rangle \colon B^\omega \to B \times B^\omega$

Universal property of $(B^\omega, \langle hd, tl\rangle)$:
For all determ. systems $(X, \langle o, t\rangle)$ there is a unique map $[\![-]\!]\colon X \to B^\omega$

such that

$$\begin{array}{rcl} hd([\![x]\!]) & = & o(x) \\ tl([\![x]\!]) & = & [\![t(x)]\!] \end{array}$$

i.e., the following diagram commutes

$$\begin{array}{ccc} X & \xrightarrow{[\![-]\!]} & B^\omega \\ {\scriptstyle \langle o,t\rangle}\downarrow & & \downarrow{\scriptstyle \langle hd,tl\rangle} \\ B \times X & \xrightarrow{id_B \times [\![-]\!]} & B \times B^\omega \end{array}$$

(that is, $[\![-]\!]$ is a morphism of deterministic systems)

i.e.,

- $(B^\omega, \langle hd, tl\rangle)$ is a final deterministic system with output in B.
- the map $[\![-]\!]\colon X \to B^\omega$ is defined by coinduction.

Coinduction Proof Principle: Stream Operation Example

Want to define $alt : B^\omega \times B^\omega \to B^\omega$,

$$alt(\sigma, \tau) = (\sigma(0), \tau(1), \sigma(2), \tau(3), \ldots)$$

Coinduction Proof Principle: Stream Operation Example

Want to define $alt : B^\omega \times B^\omega \to B^\omega$,

$$alt(\sigma, \tau) = (\sigma(0), \tau(1), \sigma(2), \tau(3), \ldots)$$

Define deterministic system

$\langle o, t \rangle \colon B^\omega \times B^\omega \to B \times (B^\omega \times B^\omega)$

by

$$\begin{array}{rcl} o(\sigma, \tau) & = & hd(\sigma) \\ t(\sigma, \tau) & = & (tl(\tau), tl(\sigma)) \end{array}$$

$$\begin{array}{ccc} B^\omega \times B^\omega & \xrightarrow{alt} & B^\omega \\ \Big\downarrow{\scriptstyle \langle o,t \rangle} & & \Big\downarrow{\scriptstyle \langle hd,tl \rangle} \\ B \times (B^\omega \times B^\omega) & \xrightarrow{id_B \times alt} & B \times B^\omega \end{array}$$

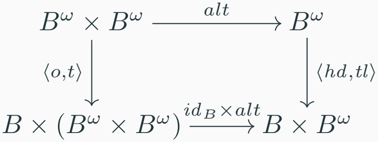
Coinduction Proof Principle: Stream Operation Example

Want to define $alt : B^\omega \times B^\omega \to B^\omega$,

$$alt(\sigma, \tau) = (\sigma(0), \tau(1), \sigma(2), \tau(3), \ldots)$$

Define deterministic system
$\langle o, t\rangle \colon B^\omega \times B^\omega \to B \times (B^\omega \times B^\omega)$
by

$$\begin{array}{rcl} o(\sigma, \tau) & = & hd(\sigma) \\ t(\sigma, \tau) & = & (tl(\tau), tl(\sigma)) \end{array}$$



Or equivalently, by the corecursive equations:

$$\begin{array}{rcl} hd(alt(\sigma, \tau)) & = & hd(\sigma) \\ tl(alt(\sigma, \tau)) & = & alt(tl(\tau), tl(\sigma)) \end{array}$$

or behavioural differential equation (BDE) (derivative $\sigma' = tl(\sigma)$):

$$\begin{array}{rcl} (alt(\sigma, \tau))(0) & = & \sigma(0) \\ (alt(\sigma, \tau))' & = & alt(\tau', \sigma') \end{array}$$

Coinductive Stream Calculus

Let B be a ring, e.g. $B = \mathbb{Z}$ (integers).

- We can define constants, sum, convolution & shuffle product:

$$
\begin{array}{ll}
[r](0) = r, & [r]' = [0] \\
(\sigma + \tau)(0) = \sigma(0) + \tau(0) & (\sigma + \tau)' = \sigma' + \tau' \\
(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0) & (\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau)
\end{array}
$$

…and many other operations on streams.

Let B be a ring, e.g. $B = \mathbb{Z}$ (integers).

- We can define constants, sum, convolution & shuffle product:

$$
\begin{array}{ll}
[r](0) = r, & [r]' = [0] \\
(\sigma + \tau)(0) = \sigma(0) + \tau(0) & (\sigma + \tau)' = \sigma' + \tau' \\
(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0) & (\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau)
\end{array}
$$

  ...and many other operations on streams.

- Linear BDE, example: $\sigma(0) = 0, \quad \sigma' = \tau$

  $\tau(0) = 1, \quad \tau' = \sigma + \tau$

  Solution $\sigma = (0, 1, 1, 2, 3, 5, 8, 13, \ldots)$ (Fibonacci numbers)

Coinductive Stream Calculus

Let B be a ring, e.g. $B = \mathbb{Z}$ (integers).

- We can define constants, sum, convolution & shuffle product:

$$
\begin{array}{ll}
[r](0) = r, & [r]' = [0] \\
(\sigma + \tau)(0) = \sigma(0) + \tau(0) & (\sigma + \tau)' = \sigma' + \tau' \\
(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0) & (\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau)
\end{array}
$$

  ...and many other operations on streams.
- Linear BDE, example: $\sigma(0) = 0, \quad \sigma' = \tau$
  $\tau(0) = 1, \quad \tau' = \sigma + \tau$
  Solution $\sigma = (0, 1, 1, 2, 3, 5, 8, 13, \ldots)$ (Fibonacci numbers)
- A non-linear example: $\sigma(0) = 1, \quad \sigma' = \sigma \times \sigma$
  Solution $\sigma = (1, 1, 2, 5, 14, 42, 132, 429, 1430, \ldots)$ (Catalan numbers)

For more, see e.g. [Rutten'03], [Winter et al.'15], [H et al.'14]

Stream Transforms

Streams form a final system in several different ways. This yields "transforms" (final systems are isomorphic).

Example: Let B be a ring. We define the difference operator:

$$\Delta(\sigma) = \sigma' - \sigma = (\sigma(1) - \sigma(0), \sigma(2) - \sigma(1), \ldots)$$

Then $\langle (-)(0), \Delta \rangle \colon B^\omega \to B \times B^\omega$ is also final, and we get isomorphism:

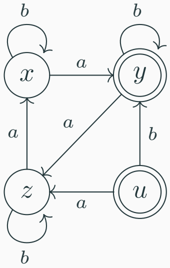
$$\begin{array}{ccc} B^\omega & \xrightarrow[\cong]{\mathcal{N}} & B^\omega \\ {\scriptstyle \langle (-)(0), \Delta \rangle}\big\downarrow & & \big\downarrow{\scriptstyle \langle (-)(0), (-)' \rangle} \\ B \times B^\omega & \xrightarrow{id_B \times \mathcal{N}} & B \times B^\omega \end{array}$$

($\mathcal{N}$ is similar to Newton transform of differentiable functions, when viewing σ as stream of Taylor coefficients.)

For more, see [Pavlovic & Escardo, 1998], [Basold et al.,2017]

Example: Deterministic Automata

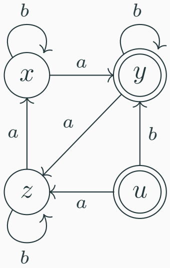
A small example:



Alphabet $A = \{a, b\}$,
State space $X = \{x, y, z, u\}$,
Accepting states $Acc = \{y, u\}$.
A^* = set of all finite sequences (words) over A.
A language is a set of words: $L \subseteq A^*$.
Language accepted at a state consists of all words that label a path to a final state.

Example: Deterministic Automata

A small example:

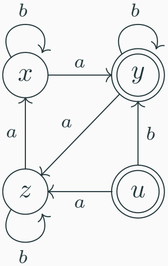


Alphabet $A = \{a, b\}$,
State space $X = \{x, y, z, u\}$,
Accepting states $Acc = \{y, u\}$.
A^* = set of all finite sequences (words) over A.
A language is a set of words: $L \subseteq A^*$.
Language accepted at a state consists of all words that label a path to a final state.

$$L(x) \quad = \quad \{w \in A^* \mid \#_a(w) \equiv 1 \mod 3\} = \{a, ab, ba, abb, bab, ...\}$$

Example: Deterministic Automata

A small example:



Alphabet $A = \{a, b\}$,
State space $X = \{x, y, z, u\}$,
Accepting states $Acc = \{y, u\}$.
A^* = set of all finite sequences (words) over A.
A language is a set of words: $L \subseteq A^*$.
Language accepted at a state consists of all words that label a path to a final state.

$$
\begin{aligned}
L(x) &= \{w \in A^* \mid \#_a(w) \equiv 1 \mod 3\} = \{a, ab, ba, abb, bab, ...\} \\
L(y) &= \{w \in A^* \mid \#_a(w) \equiv 0 \mod 3\} = \{\epsilon, b, bb, ..., aaa, ...\} \\
L(u) &= \{w \in A^* \mid \#_a(w) \equiv 0 \mod 3\} = \{\epsilon, b, bb, ..., aaa, ...\} \\
L(z) &= \{w \in A^* \mid \#_a(w) \equiv 2 \mod 3\} = \{aa, aab, ..., bbabab, ...\}
\end{aligned}
$$

Deterministic Automata as Coalgebra

- A deterministic automaton over alphabet A (omit initial state):
  transition map $\quad t\colon X \to X^A$
  output/acceptance map $\quad o\colon X \to 2 \quad (2 = \{0,1\})$
  combined $\quad \langle o, t \rangle : X \to 2 \times X^A$,
  i.e., coalgebra for Set-functor $F(X) = 2 \times X^A$.
- Morphisms of deterministic automata:

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ {\scriptstyle \langle o,t \rangle}\downarrow & & \downarrow{\scriptstyle \langle p,s \rangle} \\ 2 \times X^A & \xrightarrow{id \times f^A} & 2 \times Y^A \end{array}$$

i.e. $\forall x \in X, \forall a \in A :$

$$\begin{aligned} p(f(x)) &= o(x) \\ s(f(x))(a) &= f(t(x)(a)) \end{aligned}$$

i.e. f preserves output and transitions.

Theorem (Morphisms respect language):
If f is a morphism from $(X, \langle o, t \rangle)$ to $(Y, \langle p, s \rangle)$,
then for all $x \in X$, $L(f(x)) = L(x)$.

The Deterministic Automaton of Languages

Let $\mathcal{L} = \mathcal{P}(A^*) = \{L \subseteq A^*\}$ be the set of all languages over A.
The automaton of languages is the deterministic automaton

$$\langle O, T\rangle \colon \mathcal{L} \to 2 \times \mathcal{L}^A$$

where for all $L \in \mathcal{L}$, all $a \in A$:

$$\begin{array}{lcll} T(L)(a) & = & \{w \in A^* \mid aw \in L\} = L_a & \quad (a\text{-derivative of } L). \\ O(L) & = & 1 \text{ iff } \epsilon \in L & \end{array}$$

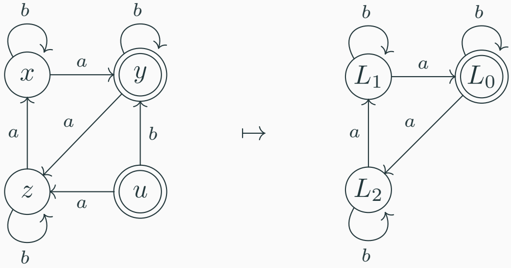
The automaton of languages is a final deterministic automaton, and the unique morphism maps a state to its language:

$$\begin{array}{ccc} X & \xrightarrow{\;L(-)\;} & \mathcal{L} \\ {\scriptstyle \langle o,t\rangle}\downarrow & & \downarrow{\scriptstyle \langle O,T\rangle} \\ 2 \times X^A & \xrightarrow{\;id_B \times L(-)^A\;} & 2 \times \mathcal{L}^A \end{array}$$

$$\begin{array}{lcl} \multicolumn{3}{l}{\forall x \in X, \forall a \in A :} \\ \epsilon \in L(x) & \text{iff} & o(x) = 1 \\ L(x)_a & = & L(t(x)(a)) \end{array}$$

(Observable) behaviour = language. Morphisms preserve behaviour.

Back to Example



where $L_i = \{w \in A^* \mid \#_a(w) \equiv i \mod 3\}$.

In the image of $(X, \langle o, t \rangle)$ under L in the final deterministic automaton, different states accept different languages; it is observable (or minimal, fully abstract).

Behavioural Equivalence and Bisimulation of Det. Automata

Two states in a deterministic automaton are behaviourally equivalent if they accept the same language.

- How can we (effectively) prove that two states are equivalent?

(Note: Languages $L \subseteq \mathcal{P}(A^*)$ are generally infinite.)

Behavioural Equivalence and Bisimulation of Det. Automata

Two states in a deterministic automaton are behaviourally equivalent if they accept the same language.

- How can we (effectively) prove that two states are equivalent?

(Note: Languages $L \subseteq \mathcal{P}(A^*)$ are generally infinite.)

Def. Let $\langle o, t \rangle \colon X \to 2 \times X^A$ be a deterministic automaton.
A relation R on X is a bisimulation if for all states x, y

$$\text{if } (x, y) \in R \text{ then} \quad \begin{array}{ll} (i) & o(x) = o(y) \\ (ii) & \text{for all } a \in A : \langle t(x)(a), t(y)(a) \rangle \in R \end{array}$$

(A bisimulation respects output and is closed under transitions)
Two states x and y are bisimilar if there is a bisimulation R such that $(x, y) \in R$. (Note: If X is finite, then finitely many relations R on X.)

Two states in a deterministic automaton are behaviourally equivalent if they accept the same language.

- How can we (effectively) prove that two states are equivalent?

(Note: Languages $L \subseteq \mathcal{P}(A^*)$ are generally infinite.)

Def. Let $\langle o, t\rangle \colon X \to 2 \times X^A$ be a deterministic automaton.
A relation R on X is a bisimulation if for all states x, y

$$\text{if } (x,y) \in R \text{ then} \quad \begin{array}{ll} (i) & o(x) = o(y) \\ (ii) & \text{for all } a \in A : \langle t(x)(a), t(y)(a)\rangle \in R \end{array}$$

(A bisimulation respects output and is closed under transitions)
Two states x and y are bisimilar if there is a bisimulation R such that $(x, y) \in R$. (Note: If X is finite, then finitely many relations R on X.)

Theorem (Coinduction proof principle):
If x and y are bisimilar, then $L(x) = L(y)$. In particular, if L_1 and L_2 are bisimilar, then $L_1 = L_2$.

Systems as Coalgebras (examples over Set)

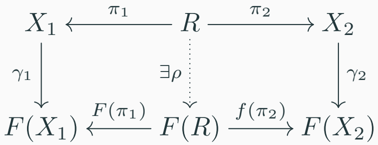
Determ. system with output in B:	$X \to B \times X$
B-labelled, non-wellfounded binary trees :	$X \to B \times X \times X$
B-labelled, possibly non-wellfnd binary trees :	$X \to 1 + B \times X \times X$
Determ. automaton on alphabet A:	$X \to 2 \times X^A$
Moore machines with input in A and output in B:	$X \to B \times X^A$
Mealy machines with input in A and output in B:	$X \to (B \times X)^A$

Systems as Coalgebras (examples over Set)

Determ. system with output in B:	$X \to B \times X$
B-labelled, non-wellfounded binary trees :	$X \to B \times X \times X$
B-labelled, possibly non-wellfnd binary trees :	$X \to 1 + B \times X \times X$
Determ. automaton on alphabet A:	$X \to 2 \times X^A$
Moore machines with input in A and output in B:	$X \to B \times X^A$
Mealy machines with input in A and output in B:	$X \to (B \times X)^A$
Nondeterministic automaton on alphabet A:	$X \to 2 \times \mathcal{P}(X)^A$
Alternating automaton on alphabet A:	$X \to 2 \times (\mathcal{Q}\,\mathcal{Q}\,X)^A$
A-labelled transition system:	$X \to \mathcal{P}(X)^A$
Markov chains ($\mathcal{D}$ is distribution monad):	$X \to \mathcal{D}(X)$
Markov decision process:	$X \to \mathbb{R} \times \mathcal{D}(X)^A$
Linear weighted automata:	$X \to \mathbb{R} \times (\mathbb{R}^X)^A$
...	...
F-coalgebra :	$X \to F(X)$

Bisimulation, Coinduction, Behavioural Equivalence

Def. A relation $R \subseteq X_1 \times X_2$ is an F-bisimulation if there is a $\rho\colon R \to F(R)$ such that projections are F-coalgebra morphisms:



Two states are F-bisimilar (notation: $x_1 \leftrightarroweq x_2$) if $(x_1, x_2) \in Z$ for some F-bisimulation Z.

Bisimulations in $\mathsf{Coalg}(F)$

Def. A relation $R \subseteq X_1 \times X_2$ is an F-bisimulation if there is a $\rho\colon R \to F(R)$ such that projections are F-coalgebra morphisms:

$$\begin{array}{ccccc} X_1 & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & X_2 \\ \downarrow{\scriptstyle\gamma_1} & & \vdots{\scriptstyle\exists\rho} & & \downarrow{\scriptstyle\gamma_2} \\ F(X_1) & \xleftarrow{F(\pi_1)} & F(R) & \xrightarrow{f(\pi_2)} & F(X_2) \end{array}$$

Two states are F-bisimilar (notation: $x_1 \Leftrightarrow x_2$) if $(x_1, x_2) \in Z$ for some F-bisimulation Z.

Equivalently (via relation lifting): R is an F-bisimulation if $R \subseteq (\gamma_1 \times \gamma_2)^{-1}(\overline{F}(R))$ where $\overline{F}\colon \mathsf{Rel} \to \mathsf{Rel}$ is:

$$\overline{F}(R) = \{\langle F(\pi_1)(u), F(\pi_2)(u)\rangle \mid u \in F(R)\} \subseteq F(X_1) \times F(X_2)$$

F-bisimilarity is the greatest fixpoint of $(\gamma_1 \times \gamma_2)^{-1}(\overline{F}(-))$.

Bisimulations in $\mathsf{Coalg}(F)$

Def. A relation $R \subseteq X_1 \times X_2$ is an F-bisimulation if there is a $\rho\colon R \to F(R)$ such that projections are F-coalgebra morphisms:

$$\begin{array}{ccccc}
X_1 & \xleftarrow{\pi_1} & R & \xrightarrow{\pi_2} & X_2 \\
{\scriptstyle\gamma_1}\downarrow & & \vdots\,{\scriptstyle\exists\rho} & & \downarrow{\scriptstyle\gamma_2} \\
F(X_1) & \xleftarrow{F(\pi_1)} & F(R) & \xrightarrow{f(\pi_2)} & F(X_2)
\end{array}$$

Two states are F-bisimilar (notation: $x_1 \ {\leftrightarrows}\ x_2$) if $(x_1, x_2) \in Z$ for some F-bisimulation Z.

Equivalently (via relation lifting): R is an F-bisimulation if $R \subseteq (\gamma_1 \times \gamma_2)^{-1}(\overline{F}(R))$ where $\overline{F}\colon \mathsf{Rel} \to \mathsf{Rel}$ is:

$$\overline{F}(R) = \{\langle F(\pi_1)(u), F(\pi_2)(u)\rangle \mid u \in F(R)\} \subseteq F(X_1) \times F(X_2)$$

F-bisimilarity is the greatest fixpoint of $(\gamma_1 \times \gamma_2)^{-1}(\overline{F}(-))$.

Coinduction proof principle:

Theorem: In final F-coalgebra (Z, ζ), bisimilarity implies equality.

Proof: If $(R, \rho) \underset{\pi_2}{\overset{\pi_1}{\rightrightarrows}} (Z, \zeta)$ then $\pi_1 = \pi_2$, hence $R \subseteq \{\langle z, z\rangle \mid z \in Z\}$.

Behavioural Equivalence in $\mathsf{Coalg}(F)$

Basic idea: Behaviour is invariant under coalgebra morphisms.

Behavioural Equivalence in Coalg(F)

Basic idea: Behaviour is invariant under coalgebra morphisms.

Let (X_1, γ_1) and (X_2, γ_2) be F-coalgebras.

Def. Two states $x_1 \in X_1$ and $x_2 \in X_2$ are behaviourally equivalent (notation: $x_1 \sim x_2$) if there exist F-coalgebra morphisms $f_i \colon (X_i, \gamma_i) \to (Y, \delta)$ such that $f_1(x_1) = f_2(x_2)$.

$$
\begin{array}{ccccc}
X_1 & \xrightarrow{f_1} & Y & \xleftarrow{f_2} & X_2 \\
\downarrow{\scriptstyle \gamma_1} & & \downarrow{\scriptstyle \delta} & & \downarrow{\scriptstyle \gamma_2} \\
F(X_1) & \xrightarrow{F(f_1)} & F(Y) & \xleftarrow{F(f_2)} & F(X_2)
\end{array}
$$

(cospan/cocongruence)

Behavioural Equivalence in $\mathsf{Coalg}(F)$

Basic idea: Behaviour is invariant under coalgebra morphisms.

Let (X_1, γ_1) and (X_2, γ_2) be F-coalgebras.

Def. Two states $x_1 \in X_1$ and $x_2 \in X_2$ are behaviourally equivalent (notation: $x_1 \sim x_2$) if there exist F-coalgebra morphisms $f_i \colon (X_i, \gamma_i) \to (Y, \delta)$ such that $f_1(x_1) = f_2(x_2)$.

$$\begin{array}{ccccc} X_1 & \xrightarrow{f_1} & Y & \xleftarrow{f_2} & X_2 \\ \downarrow{\scriptstyle \gamma_1} & & \downarrow{\scriptstyle \delta} & & \downarrow{\scriptstyle \gamma_2} \\ F(X_1) & \xrightarrow{F(f_1)} & F(Y) & \xleftarrow{F(f_2)} & F(X_2) \end{array}$$

(cospan/cocongruence)

Some basic facts:

- If final F-coalgebra exists, then
$$[\![x_1]\!] = [\![x_2]\!] \quad \Longleftrightarrow \quad x_1 \sim x_2.$$
- For all F-coalgebras: $x_1 \leftrightarroweq x_2$ implies $x_1 \sim x_2$.
- If F preserves weak pullbacks, then $x_1 \sim x_2$ implies $x_1 \leftrightarroweq x_2$. (Includes all polynomial Set-functors.)

Existence of Final F-Coalgebra

Final F-coalgebra provides coinductive definition and proof principle, but they do not always exist.
By Lambek's Lemma, if (Z, ζ) is final F-coalgebra, then $Z \cong F(Z)$.
(So powerset functor $\mathcal{P}$ has no final coalgebra.)

When do we have a final F-coalgebra, and how to obtain it?

Existence of Final F-Coalgebra

Final F-coalgebra provides coinductive definition and proof principle, but they do not always exist.
By Lambek's Lemma, if (Z, ζ) is final F-coalgebra, then $Z \cong F(Z)$.
(So powerset functor $\mathcal{P}$ has no final coalgebra.)

When do we have a final F-coalgebra, and how to obtain it?

- If F is ω^{op}-continuous (includes all polynomial Set-functors), as limit of final sequence:

$$1 \xleftarrow{\;!\;} F(1) \xleftarrow{F(!)} F^2(1) \xleftarrow{F^2(!)} F^3(1) \xleftarrow{F^3(!)} \cdots$$

- If F is κ-accessible (κ regular cardinal), as the $(\kappa + \kappa)$'th element of the final sequence [Worrell, 2005].
  (Includes e.g. finitary powerset $\mathcal{P}_\omega$.)

Application: Language Semantics of Automata with Branching

Automata with Branching

Examples of branching automata (let A be alphabet):

Nondeterministic automaton: $X \to 2 \times (\mathcal{P}X)^A$

Weighted automaton (over semiring/rig S): $X \to S \times (\mathcal{M}_S X)^A$

Probabilistic automaton: $X \to [0,1] \times (\mathcal{D}X)^A$

(where $\mathcal{M}_S(X) = \{f : X \to S \mid f \text{ has finite support}\}$)

Automata with Branching

Examples of branching automata (let A be alphabet):

Nondeterministic automaton:	$X \to 2 \times (\mathcal{P}X)^A$
Weighted automaton (over semiring/rig S):	$X \to S \times (\mathcal{M}_S X)^A$
Probabilistic automaton:	$X \to [0,1] \times (\mathcal{D}X)^A$

(where $\mathcal{M}_S(X) = \{f : X \to S \mid f \text{ has finite support}\}$)

General form: $X \to B \times (TX)^A$, i.e., FT-coalgebras where

- $F(X) = B \times X^A$,
- T is Set-monad (T, η, μ)
- B is (carrier of) Eilenberg-Moore algebra for T.

FT-behaviours are "branching behaviours". E.g. for NDA, bisimilarity is stronger than language equivalence.

Often, we are interested in (weighted/probabilistic) language semantics: $[\![x]\!] : A^* \to B$.

Language Semantics for Automata with Branching

We have a distributive law $\lambda : TF \Rightarrow FT$ of monad (T, η, μ) over functor F.

$$T(B \times X^A) \xrightarrow{\langle T\pi_1, T\pi_2 \rangle} TB \times T(X^A) \xrightarrow{\beta \times str} B \times (TX)^A$$

We obtain "determinization" functor $(-)^\sharp \colon \mathsf{Coalg}_{Set}(FT) \to \mathsf{Coalg}_{EM(T)}(F_\lambda)$ where $F_\lambda \colon EM(T) \to EM(T)$ is $F_\lambda(Y, \alpha : TY \to Y) = (FY, F\alpha \circ \lambda_Y)$.

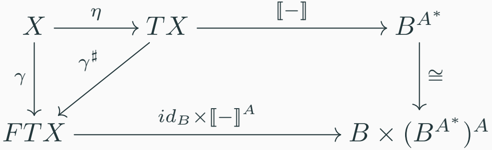
Language Semantics for Automata with Branching

We have a distributive law $\lambda : TF \Rightarrow FT$ of monad (T, η, μ) over functor F.

$$T(B \times X^A) \xrightarrow{\langle T\pi_1, T\pi_2 \rangle} TB \times T(X^A) \xrightarrow{\beta \times str} B \times (TX)^A$$

We obtain "determinization" functor $(-)^\sharp \colon \mathsf{Coalg}_{Set}(FT) \to \mathsf{Coalg}_{EM(T)}(F_\lambda)$ where $F_\lambda \colon EM(T) \to EM(T)$ is $F_\lambda(Y, \alpha : TY \to Y) = (FY, F\alpha \circ \lambda_Y)$.

The final F-coalgebra of languages lifts to final F_λ-coalgebra, yielding language semantics for FT-coalgebras:



(cf. [Bartels'03], [Jacobs'06], [Silva et al.'13], [Jacobs et al.'15])

Concluding Part 1

Summary: Universal Coalgebra

- Unifying theory of state-based systems (black-box view, observable behaviour).
- Includes many familiar system types (streams, trees, automata, Markov decision processes,...)
- Developed parametric in system type $F\colon \mathsf{C} \to \mathsf{C}$
- A coalgebra $X \to F(X)$ specifies (local) one-step behavior
- Coinductive proof and definition principle

Current coalgebra research (cf. conferences CMCS, CALCO)

- automata and formal language theory
- concurrency
- modular verification tools
- coalgebraic logic
- algebra and coalgebra

Part 2: Modal logics for coalgebras.