Streams and Coalgebra
Lecture 1

Helle Hvid Hansen and Jan Rutten

Radboud University Nijmegen & CWI Amsterdam

Representing Streams II, Lorentz Center, Leiden, January 2014

Tutorial Overview

Lecture 1 (Hansen): Coalgebra as a Unifying Theory of Systems

- Introduction and motivation
- Examples of systems.
- Universal coalgebra: basic concepts

Lecture 2 (Hansen): The Coinduction Principle

- Coinductive proofs: Bisimulations
- Coinductive definitions: Behavioural differential equations
- Definition format: Syntactic method.

Exercises (Hansen/Rutten).

Lecture 3 (Rutten): Newton Series and Shuffle Product.

Coalgebra: Historical background

- Non-wellfounded set theory (Aczel'88, Barwise-Moss'96). Solving systems of equations, self-referentiality.
  E.g. $$\begin{aligned} x &= \{a, y\} \\ y &= \{x\} \end{aligned}$$
- Program semantics: solving recursive domain equations $X \cong F(X)$ (ordered spaces, metric spaces)
- Transition systems as coalgebras (Rutten'95), precursor of J. Rutten. Universal Coalgebra, a theory of systems, 2000.
- Growing community. Established subdiscipline of theoretical computer science: automata theory, program semantics, modal logic, ...
  - Coalgebraic Methods in Computer Science (CMCS).
  - Conf. on Algebra and Coalgebra in Comp. Science (CALCO)
  - also: LICS, FoSSaCS, ICALP, MFPS, POPL, ...

Coalgebra

- General theory of state-based systems (black-box view).
- Abstract notion of observable behaviour.
- Developed parametric in system type.
- General definitions of morphism, behavioural equivalence, bisimulation, ... (parametric in system type).
- Includes many familiar systems (streams, trees, automata, relations,...)

Algebra vs Coalgebra

Coalgebra is dual to algebra, see e.g. B. Jacobs and J.J.M.M. Rutten. An introduction to (co)algebras and (co)induction, 2011 (1997).

Algebra vs Coalgebra

Coalgebra is dual to algebra, see e.g. B. Jacobs and J.J.M.M. Rutten. An introduction to (co)algebras and (co)induction, 2011 (1997).

algebra	$\leftrightsquigarrow$	**coalgebra**
well-founded structures		non-well-founded structures
construction		observation
syntax		behaviour

Algebra vs Coalgebra

Coalgebra is dual to algebra, see e.g. B. Jacobs and J.J.M.M. Rutten. An introduction to (co)algebras and (co)induction, 2011 (1997).

algebra	$\leftrightsquigarrow$	**coalgebra**
well-founded structures		non-well-founded structures
construction		observation
syntax		behaviour
congruence		bisimulation
initiality		finality
induction		coinduction

Algebra vs Coalgebra

Coalgebra is dual to algebra, see e.g. B. Jacobs and J.J.M.M. Rutten. An introduction to (co)algebras and (co)induction, 2011 (1997).

algebra	$\leftrightsquigarrow$	**coalgebra**
well-founded structures		non-well-founded structures
construction		observation
syntax		behaviour
congruence		bisimulation
initiality		finality
induction		coinduction

Algebra and coalgebra both fundamental to theoretical computer science, especially their interaction.

Motivation

What can coalgebra do for you?

- Unifying theory: reveal common mathematical structure.
- New perspective on existing results.
- New results/generalisations via transfer using general perspective.
- Theorems and proofs for many system types “in one go”.
- Mathematical tool box: morphism, bisimulation, equivalence, modal logics, ...

Some Examples

- Deterministic Systems with Output (DSO)
- Deterministic Automata with Output (DAO)

(On the board)

Remarks:

- State space can be infinite
- No initial state

Systems as Coalgebras

DSO with output in B:	$X \to B \times X$
DAO with output in B on alphabet A:	$X \to B \times X^A$
Deterministic automaton on alphabet A:	$X \to 2 \times X^A$

Systems as Coalgebras

DSO with output in B:	$X \to B \times X$
DAO with output in B on alphabet A:	$X \to B \times X^A$
Deterministic automaton on alphabet A:	$X \to 2 \times X^A$
Kripke frame (relation):	$X \to \mathcal{P}(X)$

Systems as Coalgebras

DSO with output in B:	$X \to B \times X$
DAO with output in B on alphabet A:	$X \to B \times X^A$
Deterministic automaton on alphabet A:	$X \to 2 \times X^A$
Kripke frame (relation):	$X \to \mathcal{P}(X)$
A-labelled transition system:	$X \to \mathcal{P}(X)^A$
Nondeterministic automaton on alphabet A:	$X \to 2 \times \mathcal{P}(X)^A$

Systems as Coalgebras

DSO with output in B:	$X \to B \times X$
DAO with output in B on alphabet A:	$X \to B \times X^A$
Deterministic automaton on alphabet A:	$X \to 2 \times X^A$
Kripke frame (relation):	$X \to \mathcal{P}(X)$
A-labelled transition system:	$X \to \mathcal{P}(X)^A$
Nondeterministic automaton on alphabet A:	$X \to 2 \times \mathcal{P}(X)^A$
Markov chains:	$X \to \mathcal{D}(X)$
...	

Systems as Coalgebras

DSO with output in B:	$X \to B \times X$
DAO with output in B on alphabet A:	$X \to B \times X^A$
Deterministic automaton on alphabet A:	$X \to 2 \times X^A$
Kripke frame (relation):	$X \to \mathcal{P}(X)$
A-labelled transition system:	$X \to \mathcal{P}(X)^A$
Nondeterministic automaton on alphabet A:	$X \to 2 \times \mathcal{P}(X)^A$
Markov chains:	$X \to \mathcal{D}(X)$
...	
F-coalgebra:	$X \to F(X)$

- F defines the system type
- Formally, F is a functor on a category...

Categories

A *category* $\mathcal{C}$ consists of:

- a collection $Ob(\mathcal{C})$ of *objects*, and
- a collection $Ar(\mathcal{C})$ of *arrows* between objects:

  write: $f\colon X \to Y$ or $X \xrightarrow{f} Y$
  for "f is an arrow from X to Y" where $X, Y \in Ob(\mathcal{C})$.

  $Hom(X, Y)$ is the collection of all arrows from X to Y.
- a *composition* $\circ$ *of arrows*
  $Hom(X, Y) \times Hom(Y, Z) \xrightarrow{\circ} Hom(X, Z)$, written $g \circ f$ for $f\colon X \to Y, g\colon Y \to Z$, such that

  (assoc) for all $f\colon X \to Y, g\colon Y \to Z, h\colon Z \to U$:
  $h \circ (g \circ f) = (h \circ g) \circ f$

  (id) for all $X \in Ob(\mathcal{C})$ there is an identity arrow $id_X\colon X \to X$
  such that for all $f\colon X \to Y$: $f \circ id_X = id_Y \circ f$.

Categories

A *category* $\mathcal{C}$ consists of:

- a collection $Ob(\mathcal{C})$ of *objects*, and
- a collection $Ar(\mathcal{C})$ of *arrows* between objects:

  write: $f\colon X \to Y$ or $X \xrightarrow{f} Y$

  for "f is an arrow from X to Y" where $X, Y \in Ob(\mathcal{C})$.

  $Hom(X, Y)$ is the collection of all arrows from X to Y.
- a *composition* $\circ$ *of arrows*

  $Hom(X, Y) \times Hom(Y, Z) \xrightarrow{\circ} Hom(X, Z)$, written $g \circ f$ for $f\colon X \to Y, g\colon Y \to Z$, such that

  (assoc) for all $f\colon X \to Y, g\colon Y \to Z, h\colon Z \to U$:
  $h \circ (g \circ f) = (h \circ g) \circ f$

  (id) for all $X \in Ob(\mathcal{C})$ there is an identity arrow $id_X\colon X \to X$ such that for all $f\colon X \to Y$: $f \circ id_X = id_Y \circ f$.

Categories

Examples:

- Set = sets and functions

Categories

Examples:

- *Set* = sets and functions
- Mon = monoids and monoid morphisms
- Ring = rings and ring homomorphisms
- Top = topological spaces and continuous maps.

Categories

Examples:

- *Set* = sets and functions
- Mon = monoids and monoid morphisms
- Ring = rings and ring homomorphisms
- Top = topological spaces and continuous maps.
- *Rel* = sets and relations, i.e., $f : X \to Y$ means $f \subseteq X \times Y$.
- $(P, \leq)$ partially ordered set P:
  objects are elements $x, y, ... \in P$,
  arrows: $x \to y$ iff $x \leq y$ (arrows are unique).
- ...

Exercise: Show that these are indeed categories.

Functors between Categories

A *functor F* from a category $\mathcal{C}$ to a category $\mathcal{D}$ ($F\colon \mathcal{C} \to \mathcal{D}$) associates objects and arrows from $\mathcal{C}$ with objects and arrows from $\mathcal{D}$:

- (objects) $X \in Ob(\mathcal{C}) \mapsto F(X) \in Ob(\mathcal{D})$,
- (arrows)
  $f\colon X \to Y \in Ar(\mathcal{C}) \mapsto F(f)\colon F(X) \to F(Y) \in Ar(\mathcal{D})$ s.t.:
  (id) $F(id_X) = id_{F(X)}$ for all $X \in Ob(\mathcal{C})$ and
  (comp) $F(g \circ f) = F(g) \circ F(f)$
  for all $f\colon X \to Y, g\colon Y \to Z \in Ar(\mathcal{D})$.

An *(endo)functor on $\mathcal{C}$* (or *$\mathcal{C}$-functor*) is a functor $F\colon \mathcal{C} \to \mathcal{C}$.

Functors between Categories

Examples:

- Identity functor on $\mathcal{C}$: $X \mapsto X$, $f \mapsto f$.
- Constant functor on $\mathcal{C}$ for some $A \in Ob(\mathcal{C})$: $X \mapsto A$, $f \mapsto id_A$.

Functors between Categories

Examples:

- Identity functor on $\mathcal{C}$: $X \mapsto X$, $f \mapsto f$.
- Constant functor on $\mathcal{C}$ for some $A \in Ob(\mathcal{C})$: $X \mapsto A$, $f \mapsto id_A$.
- Powerset functor on Set:
  $X \mapsto \mathcal{P}(X)$, $f \mapsto f[-]$ (direct image).

Functors between Categories

Examples:

- Identity functor on $\mathcal{C}$: $X \mapsto X$, $f \mapsto f$.
- Constant functor on $\mathcal{C}$ for some $A \in Ob(\mathcal{C})$: $X \mapsto A$, $f \mapsto id_A$.
- Powerset functor on *Set*:
  $X \mapsto \mathcal{P}(X)$, $f \mapsto f[-]$ (direct image).
- Forgetful functor from *Mon* to *Set*:
  $(X, \cdot, 1) \mapsto X$, $f \mapsto f$ (underlying function).
- Free functor from *Set* to *Mon*: $X \mapsto X^*$, $f : X \to Y \mapsto$ unique morphism $f^* : X^* \to Y^*$ extending f.
- Monotonic maps on a partially ordered set $(P, \leq)$.

Exercise: Show that these are indeed functors.

Category of F-coalgebras

Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor on $\mathcal{C}$.

Category of F-coalgebras

Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor on $\mathcal{C}$.

- An *F-coalgebra* is an arrow $\gamma\colon X \to F(X)$ in $Ar(\mathcal{C})$, also written as (X, γ).
- An *F-coalgebra morphism* $f\colon (X,\gamma) \to (Y,\delta)$ is an arrow $f\colon X \to Y$ in $Ar(\mathcal{C})$ such that $F(f) \circ \gamma = \delta \circ f$, i.e., the following diagram commutes:

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "\gamma"'] & Y \arrow[d, "\delta"] \\ F(X) \arrow[r, "F(f)"] & F(Y) \end{tikzcd}$$

- Coalg(F) denotes the category of F-coalgebras and F-coalgebra morphisms.

Exercise: Verify that Coalg(F) is indeed a category.

Basic Constructions

Assume $F\colon Set \to Set$. (All notions generalise).

- (X, γ) is a subcoalgebra of (Y, δ) if $X \subseteq Y$ and inclusion map $X \rightarrowtail Y$ is coalgebra morphism, i.e., $(X, \gamma) \rightarrowtail (Y, \delta)$.
- (Y, δ) is a quotient of (X, γ) if there is $f\colon (X, \gamma) \twoheadrightarrow (Y, \delta)$.
- the coproduct of (X, γ) and (Y, γ) is $(X + Y, \phi)$ where

$$\begin{tikzcd}
X \arrow[r, "\kappa_1"] \arrow[d, "\gamma"'] & X + Y \arrow[d, dashed, "\phi"] & Y \arrow[l, "\kappa_2"'] \arrow[d, "\delta"] \\
F(X) \arrow[r, "F\kappa_1"] & F(X + Y) & F(Y) \arrow[l, "F\kappa_2"']
\end{tikzcd}$$

where $\phi = [F\kappa_1 \circ \gamma, F\kappa_2 \circ \delta]$.

Coalgebraic Modelling of DSOs

Deterministic systems with output in B are F-coalgebras for *Set*-functor $F(X) = B \times X$. $\langle o, d \rangle : X \to B \times X$

Coalgebra morphisms:

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "\langle o_\gamma, d_\gamma \rangle"'] & Y \arrow[d, "\langle o_\delta, d_\delta \rangle"] \\ B \times X \arrow[r, "id_B \times f"] & B \times Y \end{tikzcd}$$

$$\begin{array}{rcl} \forall x \in X : o_\delta(f(x)) & = & o_\gamma(x) \\ d_\delta(f(x)) & = & f(d_\gamma(x)) \end{array}$$

Coalgebraic Modelling of DSOs

Deterministic systems with output in B are F-coalgebras for *Set*-functor $F(X) = B \times X$. $\langle o, d\rangle \colon X \to B \times X$

Coalgebra morphisms:

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "\langle o_\gamma, d_\gamma\rangle"'] & Y \arrow[d, "\langle o_\delta, d_\delta\rangle"] \\ B \times X \arrow[r, "id_B \times f"] & B \times Y \end{tikzcd}$$

$$\forall x \in X : \begin{array}{rcl} o_\delta(f(x)) & = & o_\gamma(x) \\ d_\delta(f(x)) & = & f(d_\gamma(x)) \end{array}$$

- morphisms preserve output and transitions.
- subsystems are transition closed subsets.
- quotients identify states that are "equivalent".
- coproducts are disjoint unions.
- observable behaviours are streams B^ω.

Coalgebraic Modelling of DAOs

DAOs (with output in B, alphabet A) are F-coalgebras for Set-functor $F(X) = B \times X^A$. Morphisms:

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "\langle o_\gamma, d_\gamma \rangle"'] & Y \arrow[d, "\langle o_\delta, d_\delta \rangle"] \\ B \times X^A \arrow[r, "id_B \times f^A"] & B \times Y^A \end{tikzcd}$$

$$\forall x \in X \; \forall a \in A : \quad \begin{array}{rcl} o_\delta(f(x)) & = & o_\gamma(x) \\ d_\delta(f(x))(a) & = & f(d_\gamma(x)(a)) \end{array}$$

where $\quad f^A \;:\; \begin{array}{rcl} X^A & \to & Y^A \\ h & \mapsto & f \circ h \end{array} \qquad (G(X) = X^A$ is a functor$)$

Coalgebraic Modelling of DAOs

DAOs (with output in B, alphabet A) are F-coalgebras for *Set*-functor $F(X) = B \times X^A$. Morphisms:

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[d, "\langle o_\gamma, d_\gamma \rangle"'] & Y \arrow[d, "\langle o_\delta, d_\delta \rangle"] \\ B \times X^A \arrow[r, "id_B \times f^A"] & B \times Y^A \end{tikzcd}$$

$$\forall x \in X \; \forall a \in A : \quad \begin{array}{rcl} o_\delta(f(x)) & = & o_\gamma(x) \\ d_\delta(f(x))(a) & = & f(d_\gamma(x)(a)) \end{array}$$

where $f^A \;:\; \begin{array}{ccc} X^A & \to & Y^A \\ h & \mapsto & f \circ h \end{array}$ $\quad (G(X) = X^A$ is a functor)

- morphisms preserve output and transitions.
- subsystems are transition closed subsets.
- quotients identify states that are "equivalent".
- coproducts are disjoint unions.
- observable behaviours are B-valued languages B^{A^*}.

Final F-Coalgebra

- An F-coalgebra (Z, ζ) is *final* if for all F-coalgebras (X, γ) there is a unique F-coalgebra morphism $h\colon (X, \gamma) \to (Z, \zeta)$:

$$\begin{tikzcd} X \arrow[r, dashed, "\exists! h"] \arrow[d, "\forall\gamma"'] & Z \arrow[d, "\zeta"] \\ F(X) \arrow[r, dashed, "F(h)"] & F(Z) \end{tikzcd}$$

- The unique morphism $h\colon (X, \gamma) \to (Z, \zeta)$ is called the *behaviour map*. We often write $[\![-]\!] = h$.
- Z is the collection of all behaviours of F-coalgebras.

Final F-Coalgebra Facts

- Final F-coalgebra is unique up to isomorphism (in Coalg(F)). (We speak about "the" final F-coalgebra.)
- If (Z, ζ) is final, then $\zeta \colon Z \xrightarrow{\cong} F(Z)$ is an isomorphism in $\mathcal{C}$ (Lambek's Lemma).
- Final F-coalgebras do not always exist.
  E.g. $X \ncong \mathcal{P}(X)$ for all sets X.
  But $\mathcal{P}_\omega(X) = \{U \subseteq X \mid card(U) \text{ finite}\}$ does have final coalgebra.

The Final DSO of Streams

Streams $B^\omega = \{\sigma \colon \mathbb{N} \to B\}$ are a DSO with output in B:

$$\begin{array}{rcl} B^\omega & \to & B \times B^\omega \\ \sigma & \mapsto & \langle \sigma(0), \sigma' \rangle \end{array}$$

where $\sigma(0)$ is *initial value*, *head* of σ,
$\sigma' = (\sigma(1), \sigma(2), \sigma(3), \dots)$ is *stream derivative*, *tail* of σ.

The Final DSO of Streams

Streams $B^\omega = \{\sigma \colon \mathbb{N} \to B\}$ are a DSO with output in B:

$$\begin{array}{rcl} B^\omega & \to & B \times B^\omega \\ \sigma & \mapsto & \langle \sigma(0), \sigma' \rangle \end{array}$$

where $\sigma(0)$ is *initial value*, *head* of σ,
$\sigma' = (\sigma(1), \sigma(2), \sigma(3), \ldots)$ is *stream derivative*, *tail* of σ.

$$\begin{tikzcd} X \arrow[r, "{[\![-]\!]}"] \arrow[d, "{\langle o,d \rangle}"'] & B^\omega \arrow[d, "{\langle (-)(0), (-)' \rangle}"] \\ B \times X \arrow[r, "{id_B \times [\![-]\!]}"] & B \times B^\omega \end{tikzcd}$$

$\forall x \in X$:

$$\begin{array}{rcl} [\![x]\!](0) & = & o(x) \\ [\![x]\!]' & = & [\![d(x)]\!] \end{array}$$

where behaviour map is

$$\forall x \in X : \quad [\![x]\!](n) = o(d^n(x)) \quad \forall n \in \mathbb{N}.$$

The Final DAO of Languages

The set $\mathcal{L} = B^{A^*} = \{L \colon A^* \to B\}$ of all B-valued languages over A is a DAO (with output in B, alphabet A) :

$$\begin{array}{rccc} \lambda : & \mathcal{L} & \to & B \times \mathcal{L}^A \\ & L & \mapsto & \langle L(\varepsilon), a \mapsto L_a \rangle \end{array}$$

where $L_a(w) = L(aw)$ for all $w \in A^*$ (left language derivative).

The Final DAO of Languages

The set $\mathcal{L} = B^{A^*} = \{L\colon A^* \to B\}$ of all B-valued languages over A is a DAO (with output in B, alphabet A) :

$$\begin{array}{rccl} \lambda : & \mathcal{L} & \to & B \times \mathcal{L}^A \\ & L & \mapsto & \langle L(\varepsilon), a \mapsto L_a \rangle \end{array}$$

where $L_a(w) = L(aw)$ for all $w \in A^*$ (left language derivative).

$$\begin{tikzcd} X \arrow[r, "\llbracket - \rrbracket"] \arrow[d, "\langle o,d \rangle"'] & \mathcal{L} \arrow[d, "\lambda"] \\ B \times X^A \arrow[r, "id_B \times \llbracket - \rrbracket^A"] & B \times \mathcal{L}^A \end{tikzcd}$$

$$\begin{array}{lcl} \multicolumn{3}{l}{\forall x \in X, \forall a \in A :} \\ \llbracket x \rrbracket(\varepsilon) & = & o(x) \\ \llbracket x \rrbracket_a & = & \llbracket d(x)(a) \rrbracket \end{array}$$

where behaviour map is:

$$\forall x \in X : \quad \llbracket x \rrbracket(w) = o(d(x)(w)) \quad \forall w \in A^*.$$

The DAO of Streams

The set B^ω of streams is a DAO (with output in B, alphabet A).

The DAO of Streams

The set B^ω of streams is a DAO (with output in B, alphabet A).
Case $A = \{1, 2, 3\}$.
For all $\sigma \in B^\omega$ and $j \in \{0, 1, 2\}$, let: $unzip_{j,3}(\sigma)(n) = \sigma(j + 3n)$.
Note that

$$unzip_3 := \langle unzip_{0,3}, unzip_{1,3}, unzip_{2,3} \rangle \colon B^\omega \xrightarrow{\cong} (B^\omega)^{\{1,2,3\}}$$

with inverse $zip_3 \colon (B^\omega)^{\{1,2,3\}} \to B^\omega$.

The DAO of Streams

The set B^ω of streams is a DAO (with output in B, alphabet A).
Case $A = \{1, 2, 3\}$.
For all $\sigma \in B^\omega$ and $j \in \{0, 1, 2\}$, let: $\mathit{unzip}_{j,3}(\sigma)(n) = \sigma(j + 3n)$.
Note that

$$\mathit{unzip}_3 := \langle \mathit{unzip}_{0,3}, \mathit{unzip}_{1,3}, \mathit{unzip}_{2,3} \rangle \colon B^\omega \xrightarrow{\cong} (B^\omega)^{\{1,2,3\}}$$

with inverse $\mathit{zip}_3 \colon (B^\omega)^{\{1,2,3\}} \to B^\omega$.
Take as DAO structure on B^ω,

$$\beta = B^\omega \xrightarrow[\cong]{\langle (-)(0), (-)' \rangle} B \times B^\omega \xrightarrow[\cong]{id_B \times \mathit{unzip}_3} B \times (B^\omega)^{\{1,2,3\}}$$

$$\text{Concretely,} \quad \begin{array}{rrcl} \beta \colon & B^\omega & \to & B \times (B^\omega)^{\{1,2,3\}} \\ & \sigma & \mapsto & \langle \sigma(0), j \mapsto \mathit{unzip}_{j-1,3}(\sigma') \rangle \end{array}$$

The DAO of Streams

By finality of $(\mathcal{L}, \lambda)$, there is unique $h\colon (B^\omega, \beta) \to (\mathcal{L}, \lambda)$.

The DAO of Streams

By finality of $(\mathcal{L}, \lambda)$, there is unique $h\colon (B^\omega, \beta) \to (\mathcal{L}, \lambda)$.
Concretely,

$$h(\sigma)(w) = \sigma(\nu(w)) \quad \forall w \in \{1,2,3\}^*$$

where $\nu\colon \{1,2,3\}^* \xrightarrow{\cong} \mathbb{N}$ is bijective 3-adic numeration (least significant digit first, reverse).

defined by:

$$\nu(\varepsilon) = 0,$$
$$\nu(aw) = a + 3 \cdot \nu(w)$$

$\nu(n)$	n
ε	0
1	$1 + 3 \cdot \nu(\varepsilon) = 1$
2	$2 + 3 \cdot \nu(\varepsilon) = 2$
3	$3 + 3 \cdot \nu(\varepsilon) = 3$
11	$1 + 3 \cdot \nu(1) = 4$
21	$2 + 3 \cdot \nu(1) = 5$
31	$3 + 3 \cdot \nu(1) = 6$
12	$1 + 3 \cdot \nu(2) = 7$
$\vdots$	$\vdots$

The Final DAO of Streams

ν bijective $\Rightarrow$ $h\colon B^\omega \to \mathcal{L}$ bijective $\Rightarrow$ h is DAO-isomorphism $\Rightarrow$ (B^ω, β) is also final.

$$\begin{tikzcd}
X \arrow[r, "{[\![-]\!]}"] \arrow[d, "{\langle o,d \rangle}"'] & B^\omega \arrow[d, "{\langle (-)(0), unzip_3((-)') \rangle}"] \\
B \times X^A \arrow[r, "{id_B \times [\![-]\!]^{\{1,2,3\}}}"] & B \times (B^\omega)^{\{1,2,3\}}
\end{tikzcd}$$

The Final DAO of Streams

ν bijective $\Rightarrow$ $h\colon B^\omega \to \mathcal{L}$ bijective $\Rightarrow$ h is DAO-isomorphism $\Rightarrow$ (B^ω, β) is also final.

$$\begin{tikzcd} X \arrow[rr, "{[\![-]\!]}"] \arrow[d, "{\langle o,d\rangle}"'] & & B^\omega \arrow[d, "{\langle (-)(0), unzip_3((-)') \rangle}"] \\ B \times X^A \arrow[rr, "{id_B \times [\![-]\!]^{\{1,2,3\}}}"] & & B \times (B^\omega)^{\{1,2,3\}} \end{tikzcd}$$

For general k,

- Subcoalgebra $\langle\sigma\rangle$ generated by σ in (B^ω, β) has as its states $D(\sigma) = \{\sigma(j + k^e n)_{n\in\mathbb{N}} \mid \sum_{i=0}^{e-1} k^i \leq j \leq \sum_{i=1}^{e} k^i, e \geq 1\} \cup \{\sigma\}$.
- $D(\sigma)$ finite iff k-kernel(σ) finite.
- σ is k-automatic iff $\langle\sigma\rangle$ is finite iff $h(\sigma)$ is a regular language.
- $\langle\sigma\rangle$ is a minimal DAO that generates σ.

Summary

Summary of today's lecture:

- Coalgebra is a uniform framework of state-based systems.
- Final coalgebras characterise observable behaviour.
- Streams and languages are final coalgebras.
- Coalgebraic perspective on automatic sequences via final DAO of streams.

Next lecture: Coinduction

- proof principle (when are two states equivalent?)
- definition principle: behavioural differential equations

Streams and Coalgebra
Lecture 2

Helle Hvid Hansen and Jan Rutten

Radboud University Nijmegen & CWI Amsterdam

Representing Streams II, Lorentz Center, Leiden, January 2014

Tutorial Overview

Lecture 1 (Hansen): Coalgebra as a Unifying Framework

- Introduction and motivation
- Examples: Streams and languages
- Universal coalgebra: basic concepts

Lecture 2 (Hansen): The Coinduction Principle

- Coinductive definitions: Behavioural differential equations
- Definition format: Syntactic method.
- Coinductive proofs: Bisimulations

Exercises (Hansen/Rutten)

Lecture 3 (Rutten): Newton Series and Shuffle Product

Specifying Streams

Methods:

- pointwise: $\sigma(n) = ...,\quad n \in \mathbb{N}$
- fixed point of substitution $\phi \colon A \to A^*$ (morphic sequence)
- generated by deterministic automaton with output (DAO).

Today: Coinduction (behavioural differential equations).
Idea: Use universal property of final (Z, ζ) to define maps into Z.

A final coalgebra yields a coinduction principle.

- Coinduction wrt final DSO $(B^\omega, \langle(-)(0), (-)'\rangle)$.
- Coinduction wrt final DAO $(B^\omega, \langle(-)(0), unzip_k)\rangle$.

Example: Binary operation on Streams

Define a binary operation on streams $B^\omega \times B^\omega \to B^\omega$:

$$\begin{tikzcd}
B^\omega \times B^\omega \arrow[r, dashed] \arrow[d, "\langle o{,}d \rangle"'] & B^\omega \arrow[d, "\langle (-)(0){,}(-)' \rangle"] \\
B \times (B^\omega \times B^\omega) \arrow[r, dashed] & B \times B^\omega
\end{tikzcd}$$

Example: Binary operation on Streams

Define a binary operation on streams $B^\omega \times B^\omega \to B^\omega$:

$$\begin{tikzcd} B^\omega \times B^\omega \arrow[r, dashed] \arrow[d, "\langle o, d \rangle"'] & B^\omega \arrow[d, "\langle (-)(0), (-)' \rangle"] \\ B \times (B^\omega \times B^\omega) \arrow[r, dashed] & B \times B^\omega \end{tikzcd}$$

by taking as coalgebra structure on $B^\omega \times B^\omega$: $\forall \sigma, \tau \in B^\omega$,

$$o(\sigma, \tau) = \sigma(0), \qquad d(\sigma, \tau) = \langle \tau, \sigma' \rangle$$

Example: Binary operation on Streams

Define a binary operation on streams $B^\omega \times B^\omega \to B^\omega$:

$$\begin{tikzcd}
B^\omega \times B^\omega \arrow[rr, dashed, "zip"] \arrow[d, "\langle o{,}d \rangle"'] & & B^\omega \arrow[d, "\langle (-)(0){,}(-)' \rangle"] \\
B \times (B^\omega \times B^\omega) \arrow[rr, dashed, "id_B \times zip"] & & B \times B^\omega
\end{tikzcd}$$

by taking as coalgebra structure on $B^\omega \times B^\omega$: $\forall \sigma, \tau \in B^\omega$,

$$o(\sigma,\tau) = \sigma(0), \qquad d(\sigma,\tau) = \langle \tau, \sigma' \rangle$$

Operation *zip* is *defined by coinduction*.

Example: Binary operation on Streams

Define a binary operation on streams $B^\omega \times B^\omega \to B^\omega$:

$$
\begin{tikzcd}
B^\omega \times B^\omega \arrow[r, dashed, "zip"] \arrow[d, "\langle o,d \rangle"'] & B^\omega \arrow[d, "\langle (-)(0),(-)' \rangle"] \\
B \times (B^\omega \times B^\omega) \arrow[r, dashed, "id_B \times zip"] & B \times B^\omega
\end{tikzcd}
$$

by taking as coalgebra structure on $B^\omega \times B^\omega$: $\forall \sigma, \tau \in B^\omega$,

$$o(\sigma,\tau) = \sigma(0), \qquad d(\sigma,\tau) = \langle \tau, \sigma' \rangle$$

Operation *zip* is *defined by coinduction*.
Commutativity of the diagram means that $\forall \sigma, \tau \in B^\omega$,

$$zip(\sigma,\tau)(0) = \sigma(0), \quad zip(\sigma,\tau)' = zip(\tau,\sigma') \tag{1}$$

zip is a solution to the *stream differential equation (SDE)* (1).

Behavioural Differential Equations (BDEs)

Define elements of and operations on a final F-coalgebra (Z, ζ) by specifying the behavioural equations it must satisfy.

More examples on $\mathbb{N}^\omega$:

- $\sigma'' = \sigma' + \sigma, \quad \sigma(0) = 0, \quad \sigma'(0) = 1$
  defines Fibonacci $\sigma = (0, 1, 1, 2, 3, 5, 8, \ldots)$.
- $\sigma' = \sigma + (\sigma \times \sigma), \quad \sigma(0) = 1$
  defines $\sigma = (1, 2, 6, 22, 90, 394, 1806, 8558, 41586, \ldots)$ of *(large) Schröder numbers* (sequence A006318 in OEIS), cf. [Winter-Bonsangue-Rutten'12]
- $(n \cdot \sigma)' = n \cdot \sigma' \quad (n \cdot \sigma)(0) = n \cdot \sigma(0) \qquad \forall n \in \mathbb{N}$
  defines scalar product.

Example: Hamming Numbers

Stream $\mathsf{H} \in \mathbb{N}^\omega$ of Hamming numbers

$$\mathsf{H} = (1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, \ldots)$$

consists of natural numbers of the form $2^i 3^j 5^k$ *for* $i, j, k \geq 0$ in increasing order (cf. Dijkstra'81, Yuen'92).

Example: Hamming Numbers

Stream $\mathsf{H} \in \mathbb{N}^\omega$ of Hamming numbers

$$\mathsf{H} = (1, 2, 3, 4, 5, 6, 8, 9, 10, 12, 15, 16, \ldots)$$

consists of natural numbers of the form $2^i 3^j 5^k$ *for* $i, j, k \geq 0$ in increasing order (cf. Dijkstra'81, Yuen'92).
We define H by the following SDEs:

$$\mathsf{H}(0) = 1, \qquad \mathsf{H}' = (2 \cdot \mathsf{H}) \parallel ((3 \cdot \mathsf{H}) \parallel (5 \cdot \mathsf{H}))$$

and
$$(\sigma \| \tau)(0) = \begin{cases} \sigma(0) & \text{if } \sigma(0) < \tau(0) \\ \tau(0) & \text{if } \sigma(0) \geq \tau(0) \end{cases}$$

$$(\sigma \| \tau)' = \begin{cases} \sigma' \| \tau & \text{if } \sigma(0) < \tau(0) \\ \sigma' \| \tau' & \text{if } \sigma(0) = \tau(0) \\ \sigma \| \tau' & \text{if } \sigma(0) > \tau(0) \end{cases}$$

That is, $\|$ merges two streams by taking smallest initial values first, and removing duplicates. Scalar product defined as before.

Ensuring Unique Solutions

SDEs as specification language for streams
$\Rightarrow$ We want unique solutions.

Do all SDEs have unique solutions?

Ensuring Unique Solutions

SDEs as specification language for streams
$\Rightarrow$ We want unique solutions.

Do all SDEs have unique solutions?

- $\sigma(0) = 1,\ \sigma' = \sigma'$ has many solutions.
- $\mathit{even}(\sigma)(0) = \sigma(0),\ \mathit{even}(\sigma)' = \mathit{even}(\sigma'')$ has unique solution,

$$\mathit{even}(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$$

- but $\sigma(0) = 1,\ \sigma' = \mathit{even}(\sigma)$ has no unique solution.

How can we guarantee unique solutions?
When is a stream definition productive?

A Syntactic Format for SDEs

Signature Σ: collection of operation symbols with arities.

E.g. $(B^\omega = \mathbb{R}^\omega)$

operation	$[r]$, $r \in \mathbb{R}$	X	$+$	$-$	$\times$
arity	0	0	2	1	2

$T_\Sigma(V) = \Sigma$*-terms* over (set of generators) V:

$$T_\Sigma(V) \ni t \ ::= \ v \in V \mid [r], r \in \mathbb{R} \mid \mathrm{X} \mid t + t \mid -t \mid t \times t$$

A Syntactic Format for SDEs

Signature Σ: collection of operation symbols with arities.

E.g. $(B^\omega = \mathbb{R}^\omega)$

operation	$[r],\ r \in \mathbb{R}$	X	$+$	$-$	$\times$
arity	0	0	2	1	2

$T_\Sigma(V) = \Sigma$*-terms* over (set of generators) V:

$$T_\Sigma(V) \ni t \ ::= \ v \in V \mid [r], r \in \mathbb{R} \mid \mathrm{X} \mid t + t \mid -t \mid t \times t$$

Def. A *stream definition for* Σ is a set of SDEs, one for each Σ-operation $f \in \Sigma$ of arity k of the form:

$$\begin{aligned} f(\sigma_1, \ldots, \sigma_k)(0) &= o_f(\sigma_1(0), \ldots, \sigma_k(0)) \\ f(\sigma_1, \ldots, \sigma_k)' &= d_f(\sigma_1(0), \ldots, \sigma_k(0)) \end{aligned}$$

where $o_f(\sigma_1(0), \ldots, \sigma_k(0)) \in B$
$d_f(\sigma_1(0), \ldots, \sigma_k(0)) \in T_\Sigma(\sigma_1, \ldots, \sigma_k, \sigma_1', \ldots, \sigma_k')$.

A Syntactic Format for SDEs (II)

Example: Stream definition for stream calculus signature:

$$
\begin{aligned}
[r](0) &= r, & [r]' &= [0] \qquad \forall r \in \mathbb{R} \\
\mathsf{X}(0) &= 0, & \mathsf{X}' &= [1] \\
(\sigma + \tau)(0) &= \sigma(0) + \tau(0), & (\sigma + \tau)' &= \sigma' + \tau' \\
(-\sigma)(0) &= -\sigma(0), & (-\sigma)' &= -\sigma' \\
(\sigma \times \tau)(0) &= \sigma(0) \cdot \tau(0), & (\sigma \times \tau)' &= (\sigma' \times \tau) + ([\sigma(0)] \times \tau')
\end{aligned}
$$

Example: Definition of Hamming numbers.

The syntactic format is also known as the *stream GSOS format* due to analogy with GSOS format from structural operational semantics (cf. [Bartels], [Klin], [Turi-Plotkin]). Generalisations for other coalgebra types immediate from categorical framework.

A Syntactic Format for SDEs (III)

Theorem: [Kupke-Niqui-Rutten'11][Bartels][Klin]
Given a signature Σ, any Σ-definition in the stream GSOS format has a unique solution, i.e., defines a unique interpretation of operation symbols as stream operations.

Proof sketch: A stream GSOS definition induces a coalgebra structure δ on terms:

$$\begin{tikzcd} T_\Sigma(B^\omega) \arrow[r, dashed, "\alpha"] \arrow[d, "\delta"'] & B^\omega \arrow[d, "\cong"] \\ B \times T_\Sigma(B^\omega) \arrow[r, dashed, "id_B \times \alpha"] & B \times B^\omega \end{tikzcd}$$

By coinduction, we obtain a unique term interpretation α which corresponds to unique interpretation of Σ-operations.

A Syntactic Format for SDEs (IV)

Examples not in the stream GSOS format:

$c(0) = 1, \; c' = c',$ since $d_c = c' \notin T_\Sigma(\emptyset)$.

$even(\sigma) = \sigma(0), \; even(\sigma)' = even(\sigma'')$, since $even(\sigma'') \notin T_\Sigma(\sigma, \sigma')$.

(Note: $d(0) = 1, \; d' = even(d)$ is in stream GSOS format.)

A Syntactic Format for SDEs (IV)

Examples not in the stream GSOS format:

$c(0) = 1, \; c' = c'$, since $d_c = c' \notin T_\Sigma(\emptyset)$.

$even(\sigma) = \sigma(0), \; even(\sigma)' = even(\sigma'')$, since $even(\sigma'') \notin T_\Sigma(\sigma, \sigma')$.

(Note: $d(0) = 1, \; d' = even(d)$ is in stream GSOS format.)

Remark on causal operations:

All operations defined in the stream GSOS format are *causal*:
The n'th element of output stream $f(\sigma_1, \ldots, \sigma_k)$ is determined by the first n elements of the input streams $\sigma_1, \ldots, \sigma_k$.
Operation *even* is not causal.

Conversely, all causal stream operations can be defined by a (possibly infinite) stream GSOS definition (cf. [KNR'11]).

Beyond Causality

- Endrullis,Grabmayer,Hendriks,Isihara & Klop. Productivity of stream definitions. Theor.Comp.Sci. 411, 2012.
- Ghani,Hancock & Pattinson. Representations of stream processors using nested fixed points. Logical Methods in Computer Science, 5(3), 2009.
- Abel & Pientka. Well-founded recursion with copatterns. Int. Conf. on Functional Programming (ICFP 2013).

Coinduction wrt final DAO of Streams

Recall the final DAO (output in B, alphabet $A = \{1, 2, 3\}$) of streams:

$$\beta = \langle (-)(0), \mathit{unzip}_3((-)') \rangle \colon B^\omega \to B \times (B^\omega)^{\{1,2,3\}}$$

Coinduction wrt final DAO of Streams

Recall the final DAO (output in B, alphabet $A = \{1, 2, 3\}$) of streams:

$$\beta = \langle (-)(0), \mathit{unzip}_3((-)') \rangle \colon B^\omega \to B \times (B^\omega)^{\{1,2,3\}}$$

Behaviour map $[\![-]\!] \colon (X, \langle o, d \rangle) \to (B^\omega, \alpha)$ is coalgebra morphism:

$$\forall x \in X : \quad \begin{array}{rcl} [\![x]\!](0) & = & o(x) \\ \mathit{unzip}_{0,3}([\![x]\!]') & = & [\![d(x)(1)]\!] \\ \mathit{unzip}_{1,3}([\![x]\!]') & = & [\![d(x)(2)]\!] \\ \mathit{unzip}_{2,3}([\![x]\!]') & = & [\![d(x)(3)]\!] \end{array}$$

using $\mathit{zip}_3 \circ \mathit{unzip}_3 = \mathit{id}_{B^\omega}$, this is equivalent with

$$\forall x \in X : \quad \begin{array}{lcl} [\![x]\!](0) & = & o(x) \\ [\![x]\!]' & = & \mathit{zip}_3([\![d(x)(1)]\!], [\![d(x)(2)]\!], [\![d(x)(3)]\!]) \end{array}$$

Zip-SDEs

Example of a (finite) system of zip_3-SDEs over $X = \{x, y, z\}$ with $B = \{a, b\}$:

$$\begin{array}{rclcrcl} x(0) & = & a, & \quad & x' & = & zip_3(y, x, x), \\ y(0) & = & b, & & y' & = & zip_3(x, y, x), \\ z(0) & = & b, & & z' & = & zip_3(y, z, y), \end{array}$$

Every system of zip_k-SDEs has a unique stream solution $[\![-]\!] : X \to B^\omega$.

TFAE (cf. [Endrullis-Grabmayer-Hendriks-Klop-Moss'11]):

- $\sigma \in B^\omega$ is k-automatic
- σ is in the solution to a <u>finite</u> system of zip_k-SDEs.

Stream Calculus

Recall the SDEs defining stream calculus operations on $\mathbb{R}^\omega$:

$$
\begin{aligned}
[r](0) &= r, & [r]' &= [0] \qquad \forall r \in \mathbb{R} \\
\mathsf{X}(0) &= 0, & \mathsf{X}' &= [1] \\
(\sigma+\tau)(0) &= \sigma(0)+\tau(0), & (\sigma+\tau)' &= \sigma'+\tau' \\
(-\sigma)(0) &= -\sigma(0), & (-\sigma)' &= -\sigma' \\
(\sigma\times\tau)(0) &= \sigma(0)\cdot\tau(0), & (\sigma\times\tau)' &= (\sigma'\times\tau)+([\sigma(0)]\times\tau')
\end{aligned}
$$

- $(\mathbb{R}^\omega, +, -, \times, 0, 1)$ is an integral domain.
- Fundamental theorem: $\sigma = \sigma(0) + \mathsf{X}\times\sigma'$ for all $\sigma\in\mathbb{R}^\omega$.

Convolution inverse:

$$
\sigma^{-1}(0) = \sigma(0)^{-1}, \qquad (\sigma^{-1})' = -(\sigma(0)\times\sigma')\times\sigma^{-1}
$$

under condition $\sigma(0)\neq 0$ (partial operation).

We write: $r\sigma = [r]\times\sigma$, $\frac{1}{\sigma} = 1/\sigma = \sigma^{-1}$ and $\frac{\sigma}{\tau} = \sigma\times\tau^{-1}$.

Polynomial and Rational Streams

- Polynomial stream

$$\sigma = a_0 + a_1 X + a_2 X^2 + a_3 X^3 + \cdots + a_n X^n$$

- Rational stream

$$\sigma = \frac{\rho}{\pi}$$

  where ρ, π polynomial, and $\pi(0) \neq 0$.

Stream calculus operations will come back in several talks.

How to prove stream identities?

$(\mathbb{R}^\omega, +, -, \times, 0, 1)$ is a commutative ring.

Can we prove this using only the SDE definitions?

Equivalence

- When are two systems "the same"?
  When their states encode the same set of behaviours.
- When do two states have the same behaviour?
  When $[\![x]\!] = [\![y]\!]$.

Equivalence

- When are two systems "the same"?
  When their states encode the same set of behaviours.
- When do two states have the same behaviour?
  When $[\![x]\!] = [\![y]\!]$.
- More generally, two states $x \in (X, \gamma)$ and $y \in (Y, \delta)$ are behaviourally equivalent if there exist morphisms

$$(X, \gamma) \xrightarrow{f} (Z, \zeta) \xleftarrow{g} (Y, \delta)$$

  such that $f(x) = g(y)$.
- How do we prove it? We build a bisimulation.

Coalgebraic Bisimulation

Coalgebra morphism = structure-respecting function
Coalgebra bisimulation = structure-respecting relation

Coalgebraic Bisimulation

Coalgebra morphism = structure-respecting function
Coalgebra bisimulation = structure-respecting relation

Def. (Aczel-Mendler) A bisimulation between F-coalgebras (X, γ) and (Y, δ) is a relation $R \subseteq X \times Y$ such that there is a coalgebra structure $\rho\colon R \to F(R)$ such that the projections $\pi_X\colon (R, \rho) \to (X, \gamma)$ and $\pi_Y\colon (R, \rho) \to (Y, \delta)$ are coalgebra morphisms:

$$\begin{tikzcd} X \arrow[d, "\gamma"] & R \arrow[l, "\pi_X"'] \arrow[r, "\pi_Y"] \arrow[d, dashed, "\rho"] & Y \arrow[d, "\delta"] \\ F(X) & F(R) \arrow[l, "F(\pi_X)"'] \arrow[r, "F(\pi_Y)"] & F(Y) \end{tikzcd}$$

Bisimulation for DSOs

Bisimulation diagram:

$$\begin{tikzcd} X \arrow[d, "\langle o_X, d_X \rangle"'] & R \arrow[l, "\pi_X"'] \arrow[r, "\pi_Y"] \arrow[d, dashed, "\rho"] & Y \arrow[d, "\langle o_Y, d_Y \rangle"] \\ B \times X & B \times R \arrow[l, "id_B \times \pi_X"'] \arrow[r, "id_B \times \pi_Y"] & B \times Y \end{tikzcd}$$

- Concretely:

$$\forall \langle x, y \rangle \in R : \quad o_X(x) = o_Y(y) \text{ and } \langle d_X(x), d_Y(y) \rangle \in R.$$

- Stream bisimulation (bisimulation on $(B^\omega, \langle (-)(0), (-)' \rangle)$):

$$\forall \langle \sigma, \tau \rangle \in R : \quad \sigma(0) = \tau(0) \text{ and } \langle \sigma', \tau' \rangle \in R.$$

Basic Facts about Bisimulations

- A map $f\colon X \to Y$ is a coalgebra morphism $f\colon (X,\gamma) \to (Y,\delta)$ iff $Graph(f) = \{\langle x, f(x)\rangle \mid x \in X\}$ is a bisimulation.
- Bisimulations are closed under arbitrary unions. Hence the largest bisimulation (*"bisimilarity"*, $\sim$) between two coalgebras exists: $x \sim y$ iff there is a bisimulation R s.t. $\langle x, y\rangle \in R$.
- Coalgebra morphisms preserve bisimilarity: $x \sim y \;\Rightarrow\; f(x) \sim g(y)$ if f, g are coalgebra morphims.
- Bisimilarity on a single coalgebra is an equivalence relation.
- For all (X,γ) there is a coalgebra map $\gamma_{\sim}\colon X/\!\sim \;\to F(X/\!\sim)$ s.t. the quotient map $q\colon X \to X/\!\sim$ is a coalgebra morphism.
- Generalises existing notions from process theory and modal logic.

Coinductive Proof Principle

Let (Z, ζ) be a final F-coalgebra.

- $id_Z \colon (Z, \zeta) \to (Z, \zeta)$ is the unique F-coalgebra morphism.
- Hence: $\quad (Z, \zeta) \xrightarrow{q} (Z/{\sim}, \zeta_{\sim}) \xrightarrow{!h} (Z, \zeta) \; = \; id_Z$
  hence q is injective, and $\sim = \Delta_Z$ (identity relation).

Coinductive Proof Principle

Let (Z, ζ) be a final F-coalgebra.

- $id_Z : (Z, \zeta) \to (Z, \zeta)$ is the unique F-coalgebra morphism.
- Hence: $\quad (Z, \zeta) \xrightarrow{q} (Z/\!\sim, \zeta_\sim) \xrightarrow{!h} (Z, \zeta) \;=\; id_Z$
  hence q is injective, and $\sim \,=\, \Delta_Z$ (identity relation).

A final coalgebra (Z, ζ) satisfies the Coinductive Proof Principle:

$$\forall x, y \in (Z, \zeta) : \quad x \sim y \;\Rightarrow\; x = y.$$

By preservation of bisimilarity under morphisms, we have:

For all $x \in (X, \gamma)$ and $y \in (Y, \delta)$: $\; x \sim y \;\Rightarrow\; [\![x]\!] = [\![y]\!]$.

A bisimulation is a proof of behavioural equivalence.

Stream Bisimulation Example

Prove that: $\sigma + \tau = \tau + \sigma$ for all $\sigma, \tau \in B^\omega$.

Stream Bisimulation Example

Prove that: $\sigma + \tau = \tau + \sigma$ for all $\sigma, \tau \in B^\omega$.

Take

$$R = \{\langle \sigma + \tau, \tau + \sigma \rangle \mid \sigma, \tau \in B^\omega\}.$$

R is a stream bisimulation:

$$\begin{array}{ccccccc} (\sigma + \tau)(0) & = & \sigma(0) + \tau(0) & = & \tau(0) + \sigma(0) & = & (\tau + \sigma)(0) \\ (\sigma + \tau)' & = & \sigma' + \tau' & R & \tau' + \sigma' & = & (\tau + \sigma)' \end{array}$$

Stream Bisimulation Example

Consider shuffle product defined by SDE

$$(\sigma \otimes \tau)(0) = \sigma(0) \cdot \tau(0), \qquad (\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$$

Prove that : $\sigma \otimes \tau = \tau \otimes \sigma$.

Stream Bisimulation Example

Consider shuffle product defined by SDE

$$(\sigma \otimes \tau)(0) = \sigma(0) \cdot \tau(0), \qquad (\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$$

Prove that : $\sigma \otimes \tau = \tau \otimes \sigma$.
Compute:

$$\begin{array}{lcl}
(\sigma \otimes \tau)' & = & (\sigma' \otimes \tau) + (\sigma \otimes \tau') \\
(\tau \otimes \sigma)' & = & (\tau' \otimes \sigma) + (\tau \otimes \sigma') \\
(\sigma \otimes \tau)'' & = & ((\sigma'' \otimes \tau) + (\sigma \otimes \tau')) + ((\sigma' \otimes \tau') + (\sigma \otimes \tau'')) \\
(\tau \otimes \sigma)'' & = & ((\tau'' \otimes \sigma) + (\tau \otimes \sigma')) + ((\tau' \otimes \sigma') + (\tau \otimes \sigma''))
\end{array}$$

Problem: derivatives keep "growing".

Bisimulation-up-to

Def. Let Σ denote a collection of stream operations. A relation $R \subseteq B^\omega \times B^\omega$ is a stream bisimulation-up-to-Σ if for all $\langle \sigma, \tau \rangle \in R$:

$$\sigma(0) = \tau(0) \qquad \text{and} \qquad \langle \sigma', \tau' \rangle \in \bar{R},$$

where $\bar{R} \subseteq B^\omega \times B^\omega$ is the smallest relation such that

1. $R \subseteq \bar{R}$
2. $\{ \langle \sigma, \sigma \rangle \mid \sigma \in B^\omega \} \subseteq \bar{R}$
3. $\bar{R}$ is closed under the (element-wise application of) operations in Σ. For instance, if Σ contains addition and $\langle \alpha, \beta \rangle, \langle \gamma, \delta \rangle \in \bar{R}$ then $\langle \alpha + \gamma,\ \beta + \delta \rangle \in \bar{R}$.

Lemma: If R is a bisimulation-up-to-Σ then $\bar{R}$ is a bisimulation. (Bisimulation-up-to-Σ is sound)

Bisimulation-up-to Example

Back to commutativity of shuffle product:

Observe:

$$\begin{aligned}(\sigma \otimes \tau)' &= (\sigma' \otimes \tau) + (\sigma \otimes \tau') \\ &= (\sigma \otimes \tau') + (\sigma' \otimes \tau) \\ (\tau \otimes \sigma)' &= (\tau' \otimes \sigma) + (\tau \otimes \sigma')\end{aligned}$$

So

$$R = \{\langle \sigma \otimes \tau, \tau \otimes \sigma \rangle \mid \sigma, \tau \in \mathbb{R}^\omega\}$$

is a bisimulation-up-to addition.

Note: Symbolic reasoning, algorithmic. No knowledge of binomial coefficients needed.

Summary

Coalgebra

- is a uniform framework of state-based systems.
- final coalgebras characterise observable behaviour.
- final coalgebras yield a coinduction principle (proof and definition).

Coinductive definitions

- are behavioural differential equations.
- streams are a final DSO $\rightsquigarrow$ stream differential eqs (SDEs).
- streams are a final DAO $\rightsquigarrow$ zip-SDEs.
- stream GSOS format ensures unique solutions to SDEs.

Coinductive proofs

- are bisimulations (up-to), circular proofs.
- can be constructed as a greatest fixed point.

Selected Coalgebra References

Universal Coalgebra:

J.J.M.M. Rutten. Universal coalgebra: a theory of systems. Theoretical Computer Science, 249(1):380, 2000.

B. Jacobs and J.J.M.M. Rutten. An introduction to (co)algebras and (co)induction. In: Advanced topics in bisimulation and coinduction, Cambridge Tracts in Theor. Comp. Science Volume 52, CUP 2011.

Stream Differential Equations:

J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Theoretical Computer Science Volume 308(1–3), pp. 1–53, 2003.

C. Kupke, M. Niqui, J. Rutten. Stream Differential Equations: concrete formats for coinductive definitions. Technical Report No. RR-11-10, Oxford University, 2011.

J. Winter, M. Bonsangue and J. Rutten. Coalgebraic characterizations of context-free languages. Logical Methods in Computer Science, Vol. 9 (3:14), 2013.

H.H. Hansen, C. Kupke and J. Rutten. Stream differential equations, an overview. In preparation.

Selected Coalgebra References

Automatic Sequences:

C. Grabmayer, J. Endrullis, D. Hendriks, J.W. Klop, and L.S. Moss. Automatic sequences and zip-specifications. Proceedings of LICS, 2012.

C. Kupke and J. Rutten. On the final coalgebra of automatic sequences. Logic and Program Semantics, volume 7230 of LNCS, Springer, 2012.

Automata and Formal Languages:

F. Bonchi, M. Bonsangue, H.H. Hansen, P. Panangaden, J. Rutten, A. Silva. Algebra-coalgebra duality in Brzozowski's minimization algorithm. ACM Transactions on Computational Logic, to appear.

A. Silva, F. Bonchi, M. Bonsangue, and J. Rutten. Generalizing determinization from automata to coalgebras. Logical Methods in Computer Science, 9(1), 2013.

J. Rot, M. Bonsangue, J. Rutten. Coinductive Proof Techniques for Language Equivalence. 7th International Conference on Language and Automata Theory and Applications (LATA 2013). LNCS 7810, Springer.