Recursion in Coalgebras

Mauro Jaskelioff
mjj@cs.nott.ac.uk

School of Computer Science & IT
The University of Nottingham

FoP Away Day 2007

Outline

- Brief overview of coalgebras.
- The problem of divergence when considering unguarded recursion.
- Different approaches to solving the problem.

The coalgebraic Approach
A Quick Overview

- Coalgebras are the dual of algebras

The coalgebraic Approach
A Quick Overview

- Coalgebras are the dual of algebras
- Coalgebras provide elegant models for dynamic systems

The coalgebraic Approach
A Quick Overview

- Coalgebras are the dual of algebras
- Coalgebras provide elegant models for dynamic systems
  - automatas

The coalgebraic Approach
A Quick Overview

- Coalgebras are the dual of algebras
- Coalgebras provide elegant models for dynamic systems
  - automatas
  - transition systems

The coalgebraic Approach
A Quick Overview

- Coalgebras are the dual of algebras
- Coalgebras provide elegant models for dynamic systems
  - automatas
  - transition systems
  - abstract machines

The coalgebraic Approach
A Quick Overview

- Coalgebras are the dual of algebras
- Coalgebras provide elegant models for dynamic systems
  - automatas
  - transition systems
  - abstract machines
  - object oriented systems

The coalgebraic Approach
A Quick Overview

- Coalgebras are the dual of algebras
- Coalgebras provide elegant models for dynamic systems
  - automatas
  - transition systems
  - abstract machines
  - object oriented systems
- Coalgebras are defined over a *behaviour functor B*

The coalgebraic Approach

A Quick Overview

- Coalgebras are the dual of algebras
- Coalgebras provide elegant models for dynamic systems
  - automatas
  - transition systems
  - abstract machines
  - object oriented systems
- Coalgebras are defined over a *behaviour functor B*
- *B* determines what is observable in the system.

The coalgebraic Approach
A Quick Overview

- Coalgebras are the dual of algebras
- Coalgebras provide elegant models for dynamic systems
  - automatas
  - transition systems
  - abstract machines
  - object oriented systems
- Coalgebras are defined over a *behaviour functor* B
- B determines what is observable in the system.
- More concretely: A coalgebra is an arrow

$$X \to BX$$

The carrier X can be thought of as a set of states.

A Simple Coalgebra: LTS

Labelled transition systems are typical examples of coalgebras. The behaviour in this case is the *Set* functor

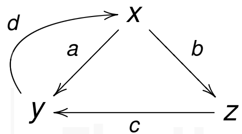
$$BX = \mathcal{P}(A \times X)$$

A Simple Coalgebra: LTS

Labelled transition systems are typical examples of coalgebras. The behaviour in this case is the *Set* functor

$$BX = \mathcal{P}(A \times X)$$

As an example, consider the set of states $X = \{x, y, z\}$, and set of actions $A = \{a, b, c, d\}$
The system



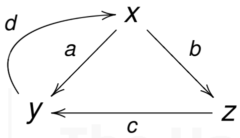
A Simple Coalgebra: LTS

Labelled transition systems are typical examples of coalgebras. The behaviour in this case is the *Set* functor

$$BX = \mathcal{P}(A \times X)$$

As an example, consider the set of states $X = \{x, y, z\}$, and set of actions $A = \{a, b, c, d\}$

The system



is given by the following coalgebra

$$\begin{aligned} \alpha &: X \to \mathcal{P}(A \times X) \\ \alpha(x) &= \{(a, y), (b, z)\} \\ \alpha(y) &= \{(d, x)\} \\ \alpha(z) &= \{(c, y)\} \end{aligned}$$

Complete Behaviour

- A coalgebra $\alpha : X \to BX$ yields one “step” of behaviour.

Complete Behaviour

- A coalgebra $\alpha\colon X \to BX$ yields one “step” of behaviour.
- The complete abstract behaviour of a system is obtained by finality.

$$\begin{array}{ccc}
X & \overset{!_\alpha}{\dashrightarrow} & \nu X.BX \\
{\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \cong} \\
BX & \underset{B!_\alpha}{\longrightarrow} & B(\nu X.BX)
\end{array}$$

Complete Behaviour

- A coalgebra $\alpha\colon X \to BX$ yields one "step" of behaviour.
- The complete abstract behaviour of a system is obtained by finality.

$$
\begin{array}{ccc}
X & \overset{!_\alpha}{\dashrightarrow} & \nu X.BX \\
{\scriptstyle \alpha}\downarrow & & \downarrow{\scriptstyle \cong} \\
BX & \underset{B!_\alpha}{\longrightarrow} & B(\nu X.BX)
\end{array}
$$

- The unique map $!_\alpha$ into the final coalgebra is often called *unfold*

Observational equivalence

The canonical notion of observational equivalence is

Coalgebraic B-bisimulation

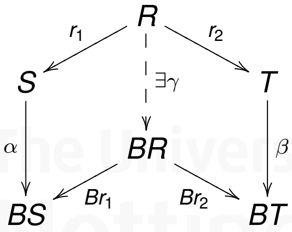
Observational equivalence

The canonical notion of observational equivalence is

Coalgebraic *B*-bisimulation

For $s \in S$, $t \in T$, $R \subseteq S \times T$

$$\langle s, \alpha \rangle \sim_B \langle t, \beta \rangle \quad \Leftrightarrow \quad \exists \gamma$$



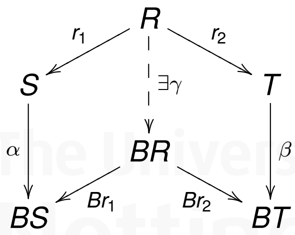
Observational equivalence

The canonical notion of observational equivalence is

Coalgebraic *B*-bisimulation

For $s \in S$, $t \in T$, $R \subseteq S \times T$

$$\langle s, \alpha \rangle \sim_B \langle t, \beta \rangle \quad \Leftrightarrow \quad \exists \gamma$$



Theorem:

$$\langle s, \alpha \rangle \sim_B \langle t, \beta \rangle \qquad \Leftrightarrow \qquad !_\alpha(s) = !_\beta(t)$$

Example: Bisimulation for LTS

For the case of labelled transition systems, the previous diagram means $(s, t) \in R$ iff

$$\forall (a, s') \in \alpha(s). \quad \exists (a, t') \in \beta(t) \ \wedge \ (s', t') \in R$$

$$\forall (a, t') \in \beta(t). \quad \exists (a, s') \in \alpha(s) \ \wedge \ (s', t') \in R$$

$$\alpha(s) = \emptyset \quad \Leftrightarrow \quad \beta(t) = \emptyset$$

which corresponds which the ordinary notion of bisimulation.

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$\begin{aligned} \psi(x) &= a\,;x\,;\psi(b;x) \\ \varphi &= \varphi\,;\psi(a) \end{aligned}$$

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$\begin{aligned} \psi(x) &= a\,;x\,;\psi(b;x) \\ \varphi &= \varphi\,;\psi(a) \end{aligned}$$

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$\begin{aligned} \psi(x) &= a\,;x\,;\psi(b;x) \\ \varphi &= \varphi\,;\psi(a) \end{aligned}$$

When are equations guarded?

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$\begin{aligned} \psi(x) &= a\,;x\,;\psi(b;x) \\ \varphi &= \varphi\,;\psi(a) \end{aligned}$$

When are equations guarded?

- Syntactically guarded

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$
\begin{aligned}
\psi(x) &= a\,;x\,;\psi(b;x) \\
\varphi &= \varphi\,;\psi(a)
\end{aligned}
$$

When are equations guarded?

- Syntactically guarded
  - RHS must begin with a non-recursive operator.

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$\begin{aligned} \psi(x) &= a\,;x\,;\psi(b;x) \\ \varphi &= \varphi\,;\psi(a) \end{aligned}$$

When are equations guarded?

- Syntactically guarded
  - RHS must begin with a non-recursive operator.
  - Avoids silly equations like $x = x$ or cycles $x = y$, $y = x$, etc.

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$\begin{aligned} \psi(x) &= a\,;x\,;\psi(b;x) \\ \varphi &= \varphi\,;\psi(a) \end{aligned}$$

When are equations guarded?

- Syntactically guarded
  - RHS must begin with a non-recursive operator.
  - Avoids silly equations like $x = x$ or cycles $x = y$, $y = x$, etc.
- Behaviourally guarded.

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$\begin{aligned} \psi(x) &= a\,;x\,;\psi(b;x) \\ \varphi &= \varphi\,;\psi(a) \end{aligned}$$

When are equations guarded?

- Syntactically guarded
  - RHS must begin with a non-recursive operator.
  - Avoids silly equations like $x = x$ or cycles $x = y$, $y = x$, etc.
- Behaviourally guarded.
  - It's possible to extract behaviour from the RHS.

A model of Recursion

- Terms of a language as carrier of a coalgebra (which defines the semantics of the language).
- We'll model recursion by systems of equations
- Example

$$\begin{aligned} \psi(x) &= a\,;x\,;\psi(b;x) \\ \varphi &= \varphi\,;\psi(a) \end{aligned}$$

When are equations guarded?

- Syntactically guarded
  - RHS must begin with a non-recursive operator.
  - Avoids silly equations like $x = x$ or cycles $x = y$, $y = x$, etc.
- Behaviourally guarded.
  - It's possible to extract behaviour from the RHS.
  - φ is syntactically but not behaviourally guarded

The Problem with Unguarded Equations

- Behaviourally guarded equations are not problematic: one can always obtain a coalgebra for them.

The Problem with Unguarded Equations

- Behaviourally guarded equations are not problematic: one can always obtain a coalgebra for them.

- $$\psi(x) \mapsto \left\{ (a, \underbrace{x; \psi(b; x)}_{\text{new state}}) \right\}$$

The Problem with Unguarded Equations

- Behaviourally guarded equations are not problematic: one can always obtain a coalgebra for them.

- $$\psi(x) \mapsto \left\{ (a, \underbrace{x; \psi(b; x)}_{\text{new state}}) \right\}$$

- If we cannot obtain behaviour from the RHS of the equation, then the only possible behaviour is divergence.

$$\varphi \quad \mapsto \quad ???$$

The Problem with Unguarded Equations

- Behaviourally guarded equations are not problematic: one can always obtain a coalgebra for them.
- $\psi(x) \mapsto \left\{ (a, \underbrace{x ; \psi(b; x)}_{\text{new state}}) \right\}$
- If we cannot obtain behaviour from the RHS of the equation, then the only possible behaviour is divergence.

$$\varphi \quad \mapsto \quad ???$$

- How to express divergence coalgebraically?

1) Recursion as Syntactic sugar

- The symbols defined by equations are not part of the language. They are syntactic sugar for their infinite expansions.

1) Recursion as Syntactic sugar

- The symbols defined by equations are not part of the language. They are syntactic sugar for their infinite expansions.
- Programs can be infinite.

1) Recursion as Syntactic sugar

- The symbols defined by equations are not part of the language. They are syntactic sugar for their infinite expansions.

- Programs can be infinite.

- This approach needs a category with more structure like CPO.

1) Recursion as Syntactic sugar

- The symbols defined by equations are not part of the language. They are syntactic sugar for their infinite expansions.
- Programs can be infinite.
- This approach needs a category with more structure like CPO.
- Approach followed by Bartek Klin, JLAP 2004.

1) Recursion as Syntactic sugar

- The symbols defined by equations are not part of the language. They are syntactic sugar for their infinite expansions.
- Programs can be infinite.
- This approach needs a category with more structure like CPO.
- Approach followed by Bartek Klin, JLAP 2004.
- It's a domain-theory-oriented solution.

2) Adding divergence to the behaviour

- Consider the behaviour $B + 1$, where we denote the element of 1 by $\perp$.

2) Adding divergence to the behaviour

- Consider the behaviour $B + 1$, where we denote the element of 1 by $\perp$.
- We can then define $\varphi \mapsto \perp$.

2) Adding divergence to the behaviour

- Consider the behaviour $B + 1$, where we denote the element of 1 by $\perp$.
- We can then define $\varphi \mapsto \perp$.
- Drawback: A coalgebra may detect divergence.

2) Adding divergence to the behaviour

- Consider the behaviour $B + 1$, where we denote the element of 1 by $\bot$.
- We can then define $\varphi \mapsto \bot$.
- Drawback: A coalgebra may detect divergence.
- $\mathit{naughty}(t) \mapsto$ if $\alpha(t) = \bot$ then *stop* else $\bot$

2) Adding divergence to the behaviour

- Consider the behaviour $B + 1$, where we denote the element of 1 by $\bot$.
- We can then define $\varphi \mapsto \bot$.
- Drawback: A coalgebra may detect divergence.
- $\mathit{naughty}(t) \mapsto$ if $\alpha(t) = \bot$ then *stop* else $\bot$
- If we work in the category *Set*, this might be acceptable!

3) Ignoring expansions

- Consider a behaviour $B_{\perp}X = X + BX$

3) Ignoring expansions

- Consider a behaviour $B_{\perp}X = X + BX$
- But equation expansions are visible!

3) Ignoring expansions

- Consider a behaviour $B_\perp X = X + BX$
- But equation expansions are visible!
- Given an equation $\chi = a$,

$$\chi \not\sim a$$

3) Ignoring expansions

- Consider a behaviour $B_{\perp}X = X + BX$
- But equation expansions are visible!
- Given an equation $\chi = a$,

$$\chi \not\rightsquigarrow a$$

- We need to consider a notion of observation that ignores equation expansion.

3) Transforming the coalgebra

- We define an endofunctor of $B_\perp$-coalgebras

$$
\begin{aligned}
\Phi_n &: B_\perp\text{-Coalg} \rightarrow B_\perp\text{-Coalg} \\
\Phi_0(k) &= X \xrightarrow{k} X + BX \\
\Phi_{n+1}(k) &= X \xrightarrow{\Phi_n(k)} X + BX \xrightarrow{[k,id]} X + BX
\end{aligned}
$$

3) Transforming the coalgebra

- We define an endofunctor of $B_\perp$-coalgebras

$$\begin{array}{lcl} \Phi_n & : & B_\perp\text{-Coalg} \to B_\perp\text{-Coalg} \\ \Phi_0(k) & = & X \xrightarrow{k} X + BX \\ \Phi_{n+1}(k) & = & X \xrightarrow{\Phi_n(k)} X + BX \xrightarrow{[k,id]} X + BX \end{array}$$

- Given $\alpha, \beta : B_\perp$-Coalg. We define

$$\langle s, \alpha \rangle \approx_B^n \langle t, \beta \rangle$$

to be

$$\langle s, \Phi_n(\alpha) \rangle \sim_B \langle t, \Phi_n(\beta) \rangle$$

3) Transforming the coalgebra

- We define an endofunctor of $B_\perp$-coalgebras

$$\begin{aligned}
\Phi_n &: B_\perp\text{-Coalg} \to B_\perp\text{-Coalg} \\
\Phi_0(k) &= X \xrightarrow{k} X + BX \\
\Phi_{n+1}(k) &= X \xrightarrow{\Phi_n(k)} X + BX \xrightarrow{[k, id]} X + BX
\end{aligned}$$

- Given $\alpha, \beta : B_\perp$-Coalg. We define

$$\langle s, \alpha \rangle \approx_B^n \langle t, \beta \rangle$$

to be

$$\langle s, \Phi_n(\alpha) \rangle \sim_B \langle t, \Phi_n(\beta) \rangle$$

- Claim: if we have n equations, considering Φ_n is enough to eliminate all finite sequences of expansions.

Summary

- Coalgebras provide a nice model of dynamic systems, but

Summary

- Coalgebras provide a nice model of dynamic systems, but
- Divergence can be problematic to model coalgebraically.

Summary

- Coalgebras provide a nice model of dynamic systems, but
- Divergence can be problematic to model coalgebraically.
- We can transform a coalgebra so that it ignores a given number of silent steps.

Summary

- Coalgebras provide a nice model of dynamic systems, but
- Divergence can be problematic to model coalgebraically.
- We can transform a coalgebra so that it ignores a given number of silent steps.

Summary

- Coalgebras provide a nice model of dynamic systems, but
- Divergence can be problematic to model coalgebraically.
- We can transform a coalgebra so that it ignores a given number of silent steps.

Future Work

- Remove dependence from n by some Φ_ω
- Correspondence between $\approx_{B_\perp}$ and what's expected in concrete cases.