



Coinductive Reasoning by Calculation

Luís S. Barbosa

DI-CCTC, Minho University, Braga, Portugal

ICTAC'05

Hanoi, 2005.10.18

Introduction

- ✦ The fact that society and everyday life is highly **dependent** on the reliable functioning of computer software
- ✦ The increasing demand for **quality certified** software (e.g., in safety-critical applications) which requires development approaches in which a system would be unacceptable unless accompanied by a **guarantee** that it respects a rigorously **specified** behaviour
- ✦ The need to drive Software Engineering into **solid engineering standards**

... places the

interplay between **modelling** and **effective reasoning**

at the heart of software design

Introduction

c.f. the basic **problem-solving strategy** from school physics:

- ✦ **understand** the problem
- ✦ **build** a mathematical **model** of it
- ✦ **reason** within such a model
- ✦ **upgrade** the model whenever necessary
- ✦ **calculate** a **solution** and **implement** it

✦ How are the **right** abstractions for a problem domain **built**?

✦ To what extent do such abstractions enable **effective reasoning** about engineered systems as well as along the process of their construction?

Introduction

Reasoning styles ...



oriented to system's **verification**: build a more concrete (detailed, closer to the programming environment, ...) model and then check it against the abstract model.

guess-and-verify or theorem-followed-by-proof



oriented to system's **construction** and **understanding**: emphasis on the derivation more concrete models from abstract ones and on seeking for generic laws by suitable (essentially synthetic) calculations.

goal-followed-by-construction

Introduction

- (1) if $a > 0$ e $b > c > 0$ then $a + b > a + c > 0$
- (2) if $a > b > 0$ then $\sqrt{a} > \sqrt{b} > 0$
- (3) $224 > 9 > 0$
- (4) $\sqrt{224} > \sqrt{9} > 0$ by (1) e (2)
- (5) $4\sqrt{14} > 3 > 0$ by (4) and arithmetic
- (6) $57 + 4\sqrt{14} > 57 + 3 > 0$ by (1) and (5)
- (7) $\sqrt{57 + 4\sqrt{14}} > \sqrt{57 + 3} > 0$ by (2) and (6)
- (8) $1 + 2\sqrt{14} > 2\sqrt{15} > 0$ by (7) and arithmetic
- (9) $8 + 1 + 2\sqrt{14} > 8 + 2\sqrt{15} > 0$ by (1) and (8)
- (10) $\sqrt{8 + 1 + 2\sqrt{14}} > \sqrt{8 + 2\sqrt{15}} > 0$ by (2) and (9)
- (11) $\sqrt{2} + \sqrt{7} > \sqrt{3} + \sqrt{5} > 0$ by (10) and arithmetic

Introduction



Formal proof easy to check



However, it fails to provide **insight** into the process



... and is quite difficult to **remember** and **reproduce**: the target is to find the bigger term, but on line (2) numbers 224 and 9 come out of the blue ...



Moreover the proof was **not** made by the order in which is presented ...

but contrast with the following proof (due to R. Backhouse):

Introduction

$$\begin{aligned} & \sqrt{2} + \sqrt{7} \quad \square \quad \sqrt{3} + \sqrt{5} \\ \equiv & \quad \{ \text{squaring is monotonic wrt } \square \} \\ & (\sqrt{2} + \sqrt{7})^2 \quad \square \quad (\sqrt{3} + \sqrt{5})^2 \\ \equiv & \quad \{ \text{arithmetic} \} \\ & 9 + 2\sqrt{14} \quad \square \quad 8 + 2\sqrt{15} \\ \equiv & \quad \{ \text{adition is monotonic wrt } \square \} \\ & 1 + 2\sqrt{14} \quad \square \quad 2\sqrt{15} \\ \equiv & \quad \{ \text{squaring is monotonic wrt } \square \} \\ & (1 + 2\sqrt{14})^2 \quad \square \quad (2\sqrt{15})^2 \end{aligned}$$

Introduction

$\equiv \{ \text{arithmetic} \}$

$$57 + 4\sqrt{14} \square 60$$

$\equiv \{ \text{addition is monotonic wrt } \square \}$

$$4\sqrt{14} \square 3$$

$\equiv \{ \text{squaring is monotonic wrt } \square \}$

$$224 \square 9$$

$\equiv \{ \text{arithmetic} \}$

$\square$ is $>$

Introduction

Proof by construction

- ✦ proof evolves through **syntactic** manipulation
- ✦ it is oriented to **fact finding** rather than to **verification of conjectures**
- ✦ starts with the identification of an **unknown** (a relation in this example)
- ✦ aims at identifying and applying properties of the unknown that may help to characterize it
- ✦ ... in this example, properties of the operators wrt the relation to be determined
- ✦ ... builds **insight**

Introduction

Problem-solving in CS deals with both

information and behavioural structures

e.g. A^* vs A^ω , the latter arising as the behaviour of (the pattern of observations upon) a (deterministic) system.



our focus will be on the behavioural side



resorting to the algebra vs coalgebra duality as a mathematical ‘explanation’ of the intuitive symmetry between data and behavioural structures.

Coalgebra as the mathematics of computational dynamics

Introduction

mathematics $\equiv$ seek for formal structures on which rigorous reasoning can be framed

dynamics $\equiv$ relies on a notion of **state** or **context** which can be (partially) observed and modified



... a 'big' topic: from 'non well-founded' sets [Aczel88]



to 'behavioural satisfaction' [Reichel81] and 'final semantics' [Plotkin, Rutten, Turi 93]



to axiomatic 'coalgebraic specification' [Jacobs97], 'coalgebraic logic' [Moss99, Kur01] and 'coinductive reasoning' [Vene, Uustalu 99, Lenisa98, Bartels04]



... partly documented in the CMCS series (from 1998)

Introduction

Focus on the ‘tradition’ of **program calculi** driven by (**initial** algebra or **final** coalgebra) type specifications providing

	Definition	Proof
initial algebras	recursion	induction
final coalgebras	co -recursion	co -induction



The role of such universals has been fundamental to a whole discipline of model transformation (the **Bird-Meertens** style).



Moreover, such properties can be turned into programming **combinators** and used, not only to **calculate** programs, but also to **program** with.

Introduction

Can such a calculational discipline, well established in **functional programming**, be extended to the world of **dynamic, state-based** systems?



internal state space ('memory' and persistence)



possibility of **interaction** with other systems during the overall computation



observable through well-defined **interfaces** to ensure flow of data

found **everywhere**: from sophisticated plant control systems to formal automata or domestic appliances.

Plan

1. Introduction
2. Sequences & Streams
3. Coinductive Modelling
4. Proofs by Calculation
5. Case study: Revisiting Process Algebras
6. Conclusions

Sequences & Streams

1. Introduction
2. Sequences & Streams
3. Coinductive Modelling
4. Proofs by Calculation
5. Case study: Revisiting Process Algebras
6. Conclusions

Sequences & Streams

Reasoning about A^*

$$\text{len}(\text{map } f \, l) = \text{len } l$$

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

$$\text{len } [] = 0$$

$$\text{len}(h : t) = 1 + \text{len } t$$

$$\text{map } f \, [] = []$$

$$\text{map } f(h : t) = f(h) : \text{map } f \, t$$

Sequences & Streams

Proof (by structural induction). Base case is trivial. Then,

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

Sequences & Streams

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?

Sequences & Streams

Reasoning about A^ω



to check the **validity** of circular definitions, as in

$$\text{map } f (h : t) = f(h) : \text{map } f t$$

$$\text{gen } f x = x : \text{gen } f x$$

$$\text{e.g., fives} = 5 : \text{fives} \quad (\text{i.e. fives} = \text{gen id } 5)$$



to establish their **properties**, as in

$$\text{map } f (\text{gen } f x) = \text{gen } f (f x)$$

Sequences & Streams

The denotational recipe: **Fixpoint Induction**



Interpret recursive definition

$$x = \Phi x$$

as a fixpoint of a continuous function Φ over a **cpo**.



Choose the least fixpoint, μ_Φ , as its semantics, computed, by Kleene's theorem as the limit of infinite chain

$$\perp \sqsubseteq \Phi \perp \sqsubseteq \Phi(\Phi \perp) \sqsubseteq \Phi(\Phi(\Phi \perp)) \sqsubseteq \dots$$

E.g.

$$\perp \sqsubseteq 5 : \text{fives} \perp \sqsubseteq 5 : 5 : \text{fives} \perp \sqsubseteq 5 : 5 : 5 : \text{fives} \perp \sqsubseteq \dots$$

Sequences & Streams

Therefore

$$\text{gen} = \mu_{\Phi} \quad \text{with} \quad \Phi G f x = x : (G f (f x))$$



Rephrase the property to be proved as a chain-complete predicate Ψ on continuous functions, as in

$$\Psi g = \forall_{f,x} . \text{map } f(g f x) = g f (f x)$$



Establish Ψ by induction using inference rule

$$\Psi \perp \wedge (\forall G . \Psi G \Rightarrow \Psi (\Phi G)) \vdash \Psi(\mu_{\Phi})$$

Note

$$\Psi \text{ gen} \equiv \text{map } f(\text{gen } f x) = \text{gen } f(f x)$$

Sequences & Streams

Proof.

$$\begin{aligned} & \Psi \perp \\ \equiv & \quad \{ \Psi \text{ definition} \} \\ & \forall_{f,x} . \text{map } f(\perp f x) = \perp f (f x) \\ \equiv & \quad \{ \perp \text{ definition} \} \\ & \forall_{f,x} . \text{map } f \perp = \perp \\ \equiv & \quad \{ \text{map } f \text{ is strict} \} \\ & \text{true} \end{aligned}$$

Sequences & Streams

$$\begin{aligned} & \Psi (\Phi G) \\ \equiv & \quad \{ \Psi \text{ definition} \} \\ & \forall_{f,x} . \text{map } f(\Phi G f x) = \Phi G f (f x) \\ \equiv & \quad \{ \Phi \text{ definition} \} \\ & \forall_{f,x} . \text{map } f(x : (G f (f x))) = (f x) : (G f (f (f x))) \\ \equiv & \quad \{ \text{map definition} \} \\ & \forall_{f,x} . (f x) : \text{map } f(G f (f x)) = (f x) : (G f (f (f x))) \\ \Leftarrow & \quad \{ \text{generalization of } f x \text{ to } z \text{ and extensionality} \} \\ & \forall_{f,x} . \text{map } f(G f z) = G f (f z) \\ \equiv & \quad \{ \Psi \text{ definition} \} \\ & \Psi G \end{aligned}$$

Sequences & Streams



... a proof method from first principles



Sound, although not complete

(e.g., fails for $\Psi l = l$ is an **infinite** sequence — why?)



Relies on **auxiliary semantic structures** (e.g., types as cpos),
being blind wrt the **structure of the program**

Sequences & Streams

The operational alternative: Coinduction

$$\text{map } f (h : t) = f(h) : \text{map } f t$$



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

Moral

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.

Sequences & Streams

Two streams s and r are equal if they can be related by a **bisimulation**:



Identical **head** observations: $\text{hd } s = \text{hd } r$,



and their **tails** — $\text{tl } s$ and $\text{tl } r$ — support a similar verification.

In other words, look for relation $R \subseteq A^\omega \times A^\omega$ such that

$$\langle u, v \rangle \in R \Rightarrow \text{hd } x = \text{hd } y \wedge \langle \text{tl } x, \text{tl } y \rangle \in R$$

I.e., R is **closed** under the **computational dynamics**

and check whether $\langle s, r \rangle \in R$.

Sequences & Streams

It is easy to check that

$$\{\langle \text{map } f (\text{gen } f x), \text{gen } f (f x) \rangle \mid x \in \dots, f \in \dots\}$$

is a **bisimulation**.

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

... leading to a variety of **bisimulation-up-to** ... proof schemes
[Milner89, Sangiorgi98, ...]

Note the proof can be presented in a **equational style** which leaves **implicit** the bisimulation relation:

Sequences & Streams

$$\begin{aligned} & \text{map } f (\text{gen } f x) \\ = & \quad \{ \text{gen definition} \} \\ & \text{map } f (x : \text{gen } f (f x)) \\ = & \quad \{ \text{map definition} \} \\ & (f x) : \text{map } f (\text{gen } f (f x)) \\ = & \quad \{ \text{coinduction hypothesis} \} \\ & (f x) : \text{gen } f (f (f x)) \\ = & \quad \{ \text{gen definition} \} \\ & \text{gen } f (f x) \end{aligned}$$

The underlying bisimulation allows an instance of the theorem to be used in a **guarded context**, i.e., in the tail of the stream.

Sequences & Streams



For proving stream equality, **coinduction** is both **sound** and **complete**



Moreover, it generalises from **streams** to a large class of **behaviour** types

Plan:

use the **universal** characterization of such types to derive a **calculational toolkit** for coinductive reasoning.

Coinductive Modelling

1. Introduction
2. Sequences & Streams
- 3. Coinductive Modelling**
4. Proofs by Calculation
5. Case study: Revisiting Process Algebras
6. Conclusions

Coinductive Modelling

(Initial) **algebras** are abstract description of **data structures**

$$[\text{nil}, \text{cons}] : 1 + O \times L \longrightarrow L$$

In general:

a **tool box**:



an **assembly process**:



artifact $\xrightarrow{d}$ artifact

The emphasis is on **construction**

Coinductive Modelling

(Final) **coalgebras** are abstract descriptions of **systems' behaviours**

$$\langle \text{at}, m \rangle : U \longrightarrow O \times U$$

In general:

a **lens**:



an **observation structure**: universe $\xrightarrow{f}$  universe

The emphasis is on **observation**

Coinductive Modelling

Observation Shapes



'opaque'

$$\bigcirc \smile \bigcirc U = \mathbf{1}$$



black & white

$$\bigcirc \smile \bigcirc U = \mathbf{2}$$



colouring

$$\bigcirc \smile \bigcirc U = O$$

... in each case the **colour set** acts as a **classifier** of the state space

Coinductive Modelling

Observation Shapes



partiality

$$\bigcirc \smallfrown \bigcirc U = U + \mathbf{1}$$



visible attributes

$$\bigcirc \smallfrown \bigcirc U = O \times U$$



external stimulus

$$\bigcirc \smallfrown \bigcirc U = U^I$$



non determinism

$$\bigcirc \smallfrown \bigcirc U = \mathcal{P}U$$

Coinductive Modelling

Objects

$$p = \langle \text{at}, \text{m} \rangle : U \longrightarrow O \times U$$

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

$$[(p)] u = [\text{at } u, \text{at } (\text{m } u), \text{at } (\text{m } (\text{m } u)), \dots]$$

Behaviours are elements of O^ω — *i.e.*, streams of type O

Coinductive Modelling

Mealy Machines

$$p = \overline{\langle \text{at}, m \rangle} : U \longrightarrow (O \times U)^I$$

Behaviours are elements of O^{I^+} — *i.e.*, functions from non empty sequences of I to O

$$[(p)] u = \lambda [s : i] . \text{at } \langle (\text{next } u) s, i \rangle$$

where

$$(\text{next } u) [] = u \quad \text{and} \quad (\text{next } u) [s : i] = m \langle (\text{next } u) s, i \rangle$$

Coinductive Modelling

Fact: Behaviours of $\circ\smile\circ$ -systems form a $\circ\smile\circ$ -system



Simple objects:

$$\omega = \langle \text{hd}, \text{tl} \rangle : O^\omega \longrightarrow O \times O^\omega$$



Mealy machines:

$$\omega = \overline{\langle \text{root}, \text{branches} \rangle} : O^{I^+} \longrightarrow (O \times O^{I^+})^I$$

where,

$$\text{root } \langle \phi, i \rangle = \phi [i] \quad \text{and} \quad \text{branches } \langle \phi, i \rangle = \lambda s . \phi [i : s]$$

Coinductive Modelling

Relating Systems

A **morphism** is a function connecting the state spaces which **preserves** the coalgebra dynamics, *i.e.*,

$$\begin{array}{ccc} U \times I & \xrightarrow{\langle \text{at}, m \rangle} & O \times U \\ h \times \text{id} \downarrow & & \downarrow \text{id} \times h \\ V \times I & \xrightarrow{\langle \text{at}', m' \rangle} & O \times V \end{array}$$

i.e.,

$$\text{at} = \text{at}' \cdot (h \times \text{id})$$

and

$$h \cdot m = m' \cdot (h \times \text{id})$$

Coinductive Modelling

Fact: $\llbracket p \rrbracket$ is a **morphism** from p to ω

Coinductive Modelling

Fact: $\llbracket p \rrbracket$ is a **morphism** from p to ω

Proof (check both conditions)

$$\begin{aligned} & (\text{root} \cdot (\llbracket p \rrbracket \times \text{id})) \langle u, i \rangle \\ = & \{ \text{root definition} \} \\ & (\llbracket p \rrbracket u) [i] \\ = & \{ \llbracket p \rrbracket \text{ definition} \} \\ & \text{at } \langle (\text{next } u) [], i \rangle \\ = & \{ \text{next definition} \} \\ & \text{at } \langle u, i \rangle \end{aligned}$$

Coinductive Modelling

and

$$\begin{aligned}
 & (\text{branches} \cdot ([p] \times \text{id})) \langle u, i \rangle \\
 = & \{ \text{branches definition} \} \\
 & \lambda [s : j] . ([p] u) [i : s : j] \\
 = & \{ [p] \text{ definition} \} \\
 & \lambda [s : j] . \text{at} \langle (\text{next } u) [i : s], j \rangle \\
 = & \{ \text{next definition} \} \\
 & \lambda [s : j] . \text{at} \langle (\text{next } m \langle u, i \rangle) s, j \rangle \\
 = & \{ [p] \text{ definition} \} \\
 & [p] (m \langle u, i \rangle)
 \end{aligned}$$

Coinductive Modelling

Fact: Morphisms preserve behaviour

$$[(p)]\ u = [(p)]\ h\ u$$

Coinductive Modelling

What's special about $\langle O^{I^+}, \omega \rangle$?



There is always a morphism $— \llbracket p \rrbracket —$ to it from any $\langle U, p = \overline{\langle \text{at}, m \rangle} \rangle$



Because morphisms preserve behaviour, such a morphism is **unique**



$\omega = \langle \text{root}, \text{branches} \rangle$ is the **final** Mealy machine



$\omega = \langle \text{hd}, \text{tl} \rangle$ is the **final** object



behaviour types are **final** coalgebras



modelling and reasoning about them may resort explicitly to such a **universal characterisation**

-
-
-

A parenthesis

(

As our underlying ‘semantic universe’ assume an elementary



space of **types** and **typed arrows** ...



with the structure of a (**partial**) **monoid**



... taken in the sequel as **sets** and **set-theoretical functions**

Function combinators are defined by **universal** arrows



associated to the **product**, **sum** and **exponential** constructions



which behave ... as they should do (**ccc** structure)

Pipelining: leading to **Function space** $[A \rightarrow B]$ (dependency)

$$A \xrightarrow{f} B \xrightarrow{g} A$$

Conjunction: leading to **Product** $A \times B$ (spatial aggregation)

$$C \xrightarrow{\langle f, g \rangle} A \times B$$

where $\langle f, g \rangle (c) = (f\ c, g\ c)$

Disjunction: leading to **Disjoint Union** $A + B$ (**choice**)

$$A + B = \{1\} \times A \cup \{2\} \times B \xrightarrow{[f,g]} C$$

where

$$[f, g](x) = (x = (1, a)) \rightarrow f a$$
$$(x = (2, b)) \rightarrow g a$$

Distinguished Functions:

empty $? : \emptyset \longrightarrow A$

bang $! : A \longrightarrow \mathbf{1}$

points $\underline{a} : \mathbf{1} \longrightarrow A$

-
-
-



Expressed in a **pointfree** notation

more suitable to perform calculations

•
•
•

Example:

The **explicit** definition of the **pairing** function looks obvious but is difficult to **handle**:

$$\langle f, g \rangle (c) = (f\ c, g\ c)$$

Now show:

that any function which builds a pair is a **pairing** function, *i.e.*,

$$\langle \pi_1 \cdot h, \pi_2 \cdot h \rangle = h$$

•
•
•

Proof. Suppose $ha = \langle b, c \rangle$. Then,

$$\begin{aligned} & \langle \pi_1 \cdot h, \pi_2 \cdot h \rangle a \\ = & \quad \{ \text{pairing definition, composition} \} \\ & \langle \pi_1(ha), \pi_2(ha) \rangle \\ = & \quad \{ \text{definition of } h \} \\ & \langle \pi_1 \langle b, c \rangle, \pi_2 \langle b, c \rangle \rangle \\ = & \quad \{ \text{definition of projection functions } \pi_1 \text{ and } \pi_2 \} \\ & \langle b, c \rangle \\ = & \quad \{ \text{definition of } h \text{ again} \} \\ & ha \end{aligned}$$

Alternative implicit definition

$\langle f, g \rangle$ is the **unique solution** of equations

$$\pi_1 \cdot x = f \quad \text{and} \quad \pi_2 \cdot x = g$$

that is

$$k = \langle f, g \rangle \Leftrightarrow \pi_1 \cdot k = f \wedge \pi_2 \cdot k = g$$

Notice



$\Rightarrow$ gives **existence** and $\Leftarrow$ gives **uniqueness**



universal characterisation

Proof.

$$\begin{aligned} h &= \langle \pi_1 \cdot h, \pi_2 \cdot h \rangle \\ &\equiv \{ \text{previous law with } f = \pi_1 \cdot h, g = \pi_1 \cdot h \} \\ &\quad \pi_1 \cdot h = \pi_1 \cdot h \wedge \pi_2 \cdot h = \pi_2 \cdot h \end{aligned}$$



simpler and smaller proof



get rid of variables



both proof and definition are **generic** and hold in other **modelling universes** (e.g., **relations**, **partial maps** or **ordered structures**, ...)

Warning

Expressivity in modelling and suitability for calculation may conflict!



Modelling requires descriptive notations, often domain-specific, hopefully intuitive and executable.



Calculation requires generic, concise and precise notations

... *i.e.*, simple and remarkably effective

(E. W. Dijkstra's definition of elegance in Maths)

Example: extensive use of nested quantifiers

Such a trend for **notational economy** is well-known throughout the history of Maths, as a sort of 'natural language implosion':

.60.ṗ.2.ce son yguales a .30.co

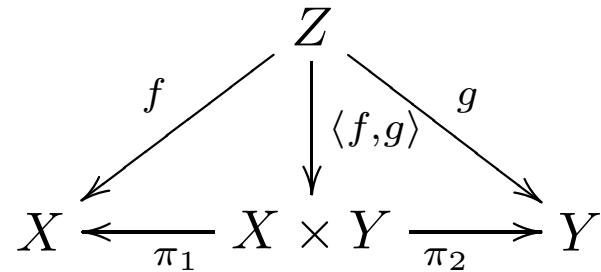
(P. Nunes, Coimbra, 1567) for nowadays $60 + 2x^2 = 30x$

B 3 in A quad - D plano in A + A cubo aequatur Z solido

(F. Viète, Paris, 1591) for nowadays $3BA^3 - DA + A^3 = Z$

But recall also the **Laplace transform**:

maintain different domains for **modelling** and **calculating**!



Universal property:

$$k = \langle f, g \rangle \Leftrightarrow \pi_1 \cdot k = f \wedge \pi_2 \cdot k = g$$

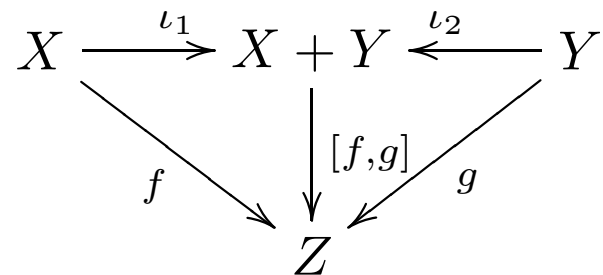
Laws:

$$\pi_1 \cdot \langle f, g \rangle = f, \pi_2 \cdot \langle f, g \rangle = g$$

$$\langle \pi_1, \pi_2 \rangle = \text{id}_{X \times Y}$$

$$\langle g, h \rangle \cdot f = \langle g \cdot f, h \cdot f \rangle$$

$$(i \times j) \cdot \langle g, h \rangle = \langle i \cdot g, j \cdot h \rangle$$



Universal property:

$$k = [f, g] \iff k \cdot \iota_1 = f \wedge k \cdot \iota_2 = g$$

Laws:

$$[f, g] \cdot \iota_1 = f, [f, g] \cdot \iota_2 = g$$

$$[\iota_1, \iota_2] = \text{id}_{X+Y}$$

$$f \cdot [g, h] = [f \cdot g, f \cdot h]$$

$$[g, h] \cdot (i + j) = [g \cdot i, h \cdot j]$$

Function Space

Final & Initial Objects:

$$!_Y \cdot f = !_X$$

$$f \cdot ?_X = ?_Y$$

Function Space

$$\begin{array}{ccc} X & & X \times C \\ \bar{f} \downarrow & & \downarrow \bar{f} \times \text{id}_C \quad \searrow f \\ Y^C & & Y^C \times C \xrightarrow{\text{ev}} Y \end{array}$$

Universal property:

$$k = \bar{f} \Leftrightarrow f = \text{ev} \cdot (k \times \text{id})$$

End of parenthesis

)

Proofs by Calculation

1. Introduction
2. Sequences & Streams
3. Coinductive Modelling
- 4. Proofs by Calculation**
5. Case study: Revisiting Process Algebras
6. Conclusions

Proofs By Calculation

Recall: Mealy machines (on I and O) were introduced as systems observed through

$$\bigcirc \smile \bigcirc = (O \times -)^I$$

$\bigcirc \smile \bigcirc$ as a mapping to decompose the 'observable-universe' U into an 'observation context' $(O \times U)^I$

$$\begin{array}{ccc} U & \xrightarrow{p} & \bigcirc \smile \bigcirc U \\ \downarrow h & & \downarrow \bigcirc \smile \bigcirc h \\ V & \xrightarrow{p'} & \bigcirc \smile \bigcirc U' \end{array}$$

Observation shapes $\bigcirc \smile \bigcirc$ are **functors**
 Observation systems are $\bigcirc \smile \bigcirc$ -**coalgebras**

Proofs By Calculation

Whenever the **space of behaviours** of a class of T-coalgebras can be turned on a T-coalgebra itself

$$\omega_T : \nu_T \longrightarrow T\nu_T$$

this is the **final** coalgebra: from any other T-coalgebra p there is a unique morphism $[(p)]$ making the following diagram to commute:

$$\begin{array}{ccc} \nu_T & \xrightarrow{\omega_T} & T\nu_T \\ \uparrow [(p)] & & \uparrow T[(p)] \\ U & \xrightarrow{p} & TU \end{array}$$

Proofs By Calculation

The universal property is equivalently captured by the following law:

$$k = \llbracket p \rrbracket \Leftrightarrow \omega_{\top} \cdot k = \top k \cdot p$$



Existence $\equiv$ definition principle (co-recursion)



Uniqueness $\equiv$ proof principle (co-induction)

From which:

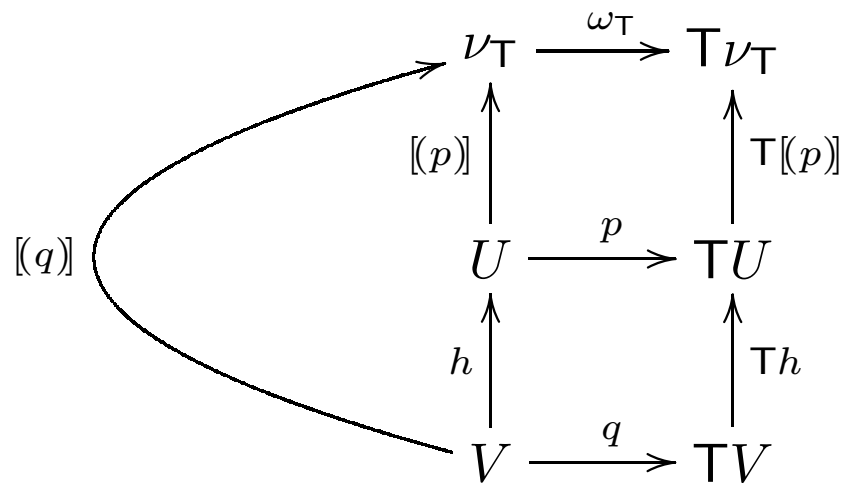
cancellation $\omega_{\top} \cdot \llbracket p \rrbracket = \top \llbracket p \rrbracket \cdot p$

reflection $\llbracket \omega_{\top} \rrbracket = \text{id}_{\nu_{\top}}$

fusion $\llbracket p \rrbracket \cdot h = \llbracket q \rrbracket$ if $p \cdot h = \top h \cdot q$

Proofs By Calculation

Example: fusion law



Proofs By Calculation

Example: fusion law

$$\begin{aligned} & \llbracket p \rrbracket \cdot h = \llbracket q \rrbracket \\ \equiv & \quad \{ \text{universal law} \} \\ & \omega \cdot \llbracket p \rrbracket \cdot h = T(\llbracket p \rrbracket \cdot h) \cdot q \\ \equiv & \quad \{ \text{cancellation law and } T \text{ functor} \} \\ & T\llbracket p \rrbracket \cdot p \cdot h = T\llbracket p \rrbracket \cdot Th \cdot q \\ \Leftarrow & \quad \{ \text{function equality} \} \\ & p \cdot h = \cdot Th \cdot q \end{aligned}$$

Proofs By Calculation

Example: Stream rep, merge and twist

$$\begin{array}{ccc} O^\omega & \xrightarrow{\langle \text{hd}, \text{tl} \rangle} & O \times O^\omega \\ \text{rep} \uparrow & & \uparrow \text{id} \times \text{rep} \\ O & \xrightarrow{\Delta} & O \times O \end{array}$$

$$\text{rep} = [(\Delta)]$$

Δ carries the ‘genetic inheritance’ of the generating process

From a programming viewpoint it is the **eureka!** step

Proofs By Calculation

Coinductive Definition = behaviour specification under all the **observers**

$$\begin{aligned} & (\text{id} \times \text{rep}) \cdot \Delta = \langle \text{hd}, \text{tl} \rangle \cdot \text{rep} \\ = & \quad \{ \Delta \text{ definition} \} \\ & (\text{id} \times \text{rep}) \cdot \langle \text{id}, \text{id} \rangle = \langle \text{hd}, \text{tl} \rangle \cdot \text{rep} \\ = & \quad \{ \times \text{ absorption and fusion} \} \\ & \langle \text{id}, \text{rep} \rangle = \langle \text{hd} \cdot \text{rep}, \text{tl} \cdot \text{rep} \rangle \\ = & \quad \{ \text{structural equality} \} \\ & \text{hd} \cdot \text{rep} = \text{id} \quad \wedge \quad \text{tl} \cdot \text{rep} = \text{rep} \\ = & \quad \{ \text{going pointwise} \} \\ & \text{hd} (\text{rep } a) = a \quad \wedge \quad \text{tl} (\text{rep } a) = \text{rep } a \end{aligned}$$

Proofs By Calculation

Stream merge

$$\begin{array}{ccc} A^\omega & \xrightarrow{\langle \text{hd}, \text{tl} \rangle} & A \times A^\omega \\ \text{merge} \uparrow & & \uparrow \text{id} \times \text{merge} \\ A^\omega \times A^\omega & \xrightarrow{g} & A \times (A^\omega \times A^\omega) \end{array}$$

$$g = \langle \text{hd} \cdot \pi_1, s \cdot (\text{tl} \times \text{id}) \rangle$$

Proofs By Calculation

Stream *twist*

$$\begin{array}{ccc} O^\omega & \xrightarrow{\langle \text{hd}, \text{tl} \rangle} & O \times O^\omega \\ \text{twist} = \llbracket g \rrbracket \uparrow & & \uparrow \text{id} \times \text{twist} \\ O \times O & \xrightarrow{g} & O \times (O \times O) \end{array}$$

$$g = \langle \pi_1, s \rangle$$

Proofs By Calculation

Lemma: $\text{merge } (a^\omega, b^\omega) = (ab)^\omega$

i.e.

$$\text{merge} \cdot (\text{rep} \times \text{rep}) = \text{twist}$$

Proofs By Calculation

$$\begin{aligned} & \text{merge} \cdot (\text{rep} \times \text{rep}) = \text{twist} \\ = & \quad \{ \text{merge definition} \} \\ & [(\langle \text{hd} \cdot \pi_1, s \cdot (\text{tl} \times \text{id}) \rangle)] \cdot (\text{rep} \times \text{rep}) = [(\langle \pi_1, s \rangle)] \\ \Leftarrow & \quad \{ \text{coinduction fusion} \} \\ & \langle \text{hd} \cdot \pi_1, s \cdot (\text{tl} \times \text{id}) \rangle \cdot (\text{rep} \times \text{rep}) = \text{id} \times (\text{rep} \times \text{rep}) \cdot \langle \pi_1, s \rangle \\ = & \quad \{ \times \text{absorption and reflection} \} \\ & \langle \text{hd} \cdot \text{rep} \cdot \pi_1, s \cdot ((\text{tl} \cdot \text{rep}) \times \text{rep}) \rangle = \text{id} \times (\text{rep} \times \text{rep}) \cdot \langle \pi_1, s \rangle \\ = & \quad \{ \text{tl} \cdot \text{rep} = \text{rep} \text{ and } \text{hd} \cdot \text{rep} = \text{id} \} \\ & \langle \pi_1, s \cdot (\text{rep} \times \text{rep}) \rangle = \text{id} \times (\text{rep} \times \text{rep}) \cdot \langle \pi_1, s \rangle \end{aligned}$$

Proofs By Calculation

$$\begin{aligned} & \langle \pi_1, s \cdot (\text{rep} \times \text{rep}) \rangle = \text{id} \times (\text{rep} \times \text{rep}) \cdot \langle \pi_1, s \rangle \\ = & \{ \times \text{absorption} \} \\ & \langle \pi_1, s \cdot (\text{rep} \times \text{rep}) \rangle = \langle \pi_1, (\text{rep} \times \text{rep}) \cdot s \rangle \\ = & \{ s \text{ is natural, i.e., } (f \times g) \cdot s = s \cdot (g \times f) \} \\ & \langle \pi_1, s \cdot (\text{rep} \times \text{rep}) \rangle = \langle \pi_1, s \cdot (\text{rep} \times \text{rep}) \rangle \end{aligned}$$

Proofs By Calculation

Example: Streams of Reals — an exercise from [Rutten, 01]

Sum

$$\begin{array}{ccc} \mathbb{R}^\omega & \xrightarrow{\langle \text{hd}, \text{tl} \rangle} & \mathbb{R} \times \mathbb{R}^\omega \\ \uparrow + & & \uparrow \text{id} \times + \\ \mathbb{R}^\omega \times \mathbb{R}^\omega & \xrightarrow{\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle} & \mathbb{R} \times (\mathbb{R}^\omega \times \mathbb{R}^\omega) \end{array}$$

i.e.,

$$+ = [(\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle)]$$

Proofs By Calculation

Lemma: $\sigma + \tau = \tau + \sigma$

or, in **pointfree** notation

$$+ \cdot s = +$$

where $s = \langle \pi_2, \pi_1 \rangle$ is the swap natural isomorphism

$$\begin{aligned} &+ \cdot s = + \\ \equiv &\{ \text{definition} \} \\ &[[\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle]] \cdot s = [[\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle]] \end{aligned}$$

Proofs By Calculation

$$\begin{aligned} & [[\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle]] \cdot s = [[\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle]] \\ \Leftarrow & \quad \{ \text{coinduction fusion law} \} \\ & \langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle \cdot s = (\text{id} \times s) \cdot \langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle \\ \equiv & \quad \{ \times\text{-fusion and absorption laws} \} \\ & \langle + \cdot (\text{hd} \times \text{hd}) \cdot s, (\text{tl} \times \text{tl}) \cdot s \rangle = \langle + \cdot (\text{hd} \times \text{hd}), s \cdot (\text{tl} \times \text{tl}) \rangle \\ \equiv & \quad \{ s \text{ is natural} \} \\ & \langle + \cdot s \cdot (\text{hd} \times \text{hd}), s \cdot (\text{tl} \times \text{tl}) \rangle = \langle + \cdot (\text{hd} \times \text{hd}), s \cdot (\text{tl} \times \text{tl}) \rangle \\ \equiv & \quad \{ \text{arithmetic addition is commutative (i.e., } + \cdot s = + \text{)} \} \\ & \langle + \cdot (\text{hd} \times \text{hd}), s \cdot (\text{tl} \times \text{tl}) \rangle = \langle + \cdot (\text{hd} \times \text{hd}), s \cdot (\text{tl} \times \text{tl}) \rangle \end{aligned}$$

Proofs By Calculation

Reals as streams

$$\begin{array}{ccc}
 \mathbb{R}^\omega & \xrightarrow{\langle \text{hd}, \text{tl} \rangle} & \mathbb{R} \times \mathbb{R}^\omega \\
 \uparrow [\] & & \uparrow \text{id} \times [\] \\
 \mathbb{R} & \xrightarrow{\langle \text{id}, \underline{0} \cdot ! \rangle} & \mathbb{R} \times \mathbb{R}
 \end{array}$$

i.e.,

$$[\] = [(\langle \text{id}, \underline{0} \cdot ! \rangle)]$$

Then define

$$[r] = [\] r \quad \text{which equivaless to} \quad \underline{[r]} = [\] \cdot \underline{r}$$

Proofs By Calculation

Lemma: $\langle \text{hd}, \text{tl} \rangle \cdot [] = (\text{id} \times []) \cdot \langle \text{id}, \underline{0} \cdot ! \rangle$

$$\begin{aligned} & \langle \text{hd}, \text{tl} \rangle \cdot [] = (\text{id} \times []) \cdot \langle \text{id}, \underline{0} \cdot ! \rangle \\ \equiv & \quad \{ \text{ } \times \text{-fusion and absorption laws} \} \\ & \langle \text{hd} \cdot [], \text{tl} \cdot [] \rangle = \langle \text{id}, [] \cdot \underline{0} \cdot ! \rangle \\ \equiv & \quad \{ \text{pairing equality} \} \\ & \text{hd} \cdot [] = \text{id} \quad \wedge \quad \text{tl} \cdot [] = [] \cdot \underline{0} \cdot ! \\ \Rightarrow & \quad \{ \text{applying to an arbitrary } r \text{ value} \} \\ & \text{hd } [r] = r \quad \wedge \quad \text{tl } [r] = ([] \cdot \underline{0} \cdot !) r = [] 0 = [0] \end{aligned}$$

Proofs By Calculation

Lemma: $\sigma + [0] = \sigma$

or, in **pointfree** notation

$$+ \cdot \langle \text{id}, \underline{[0]} \rangle = \text{id}$$

where $\underline{[0]} = [0] \cdot !$

$$+ \cdot \langle \text{id}, \underline{[0]} \rangle = \text{id}$$

$$\equiv \{ + \text{ definition and coinduction reflexive law } \}$$

$$[(\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle)] \cdot \langle \text{id}, \underline{[0]} \rangle = [(\langle \text{hd}, \text{tl} \rangle)]$$

$$\Leftarrow \{ \text{coinduction fusion law} \}$$

$$\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle \cdot \langle \text{id}, \underline{[0]} \rangle = (\text{id} \times \langle \text{id}, \underline{[0]} \rangle) \cdot \langle \text{hd}, \text{tl} \rangle$$

Proofs By Calculation

$$\begin{aligned}
 & \langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle \cdot \langle \text{id}, \underline{[0]} \rangle = (\text{id} \times \langle \text{id}, \underline{[0]} \rangle) \cdot \langle \text{hd}, \text{tl} \rangle \\
 \equiv & \quad \{ \times\text{-fusion and absorption laws} \} \\
 & \langle + \cdot (\text{hd} \times \text{hd}) \cdot \langle \text{id}, \underline{[0]} \rangle, (\text{tl} \times \text{tl}) \cdot \langle \text{id}, \underline{[0]} \rangle \rangle = \langle \text{hd}, \langle \text{id}, \underline{[0]} \rangle \cdot \text{tl} \rangle \\
 \equiv & \quad \{ \times\text{-fusion and absorption laws (again!)} \} \\
 & \langle + \cdot \langle \text{hd}, \text{hd} \cdot \underline{[0]} \rangle, \langle \text{tl}, \text{tl} \cdot \underline{[0]} \rangle \rangle = \langle \text{hd}, \langle \text{tl}, \underline{[0]} \cdot \text{tl} \rangle \rangle \\
 \equiv & \quad \{ \text{point definition } (f \cdot \underline{p} = \underline{f p}) \text{ and constant} \} \\
 & \langle + \cdot \langle \text{hd}, \underline{\text{hd} [0]} \rangle, \langle \text{tl}, \underline{\text{tl} [0]} \rangle \rangle = \langle \text{hd}, \langle \text{tl}, [0] \cdot ! \cdot \text{tl} \rangle \rangle \\
 \equiv & \quad \{ [0] \text{ laws above and } !\text{-universal } (! = ! \cdot f) \} \\
 & \langle + \cdot \langle \text{hd}, \underline{0} \rangle, \langle \text{tl}, \underline{[0]} \rangle \rangle = \langle \text{hd}, \langle \text{tl}, \underline{[0]} \rangle \rangle \\
 \equiv & \quad \{ \text{arithmetic} \} \\
 & \langle \text{hd}, \langle \text{tl}, \underline{[0]} \rangle \rangle = \langle \text{hd}, \langle \text{tl}, \underline{[0]} \rangle \rangle
 \end{aligned}$$

Proofs By Calculation

Lemma: $[r + p] = [r] + [p]$

corresponding, in **pointfree** notation, to an expression to which **fusion** cannot be directly applied:

$$+ \cdot ([] \times []) = [] \cdot +$$

Proofs By Calculation



Step 1: Show that right hand side $[] \cdot +$ can be re-written as a coinductive extension

$$\begin{array}{ccc}
 \mathbb{R}^\omega & \xrightarrow{\langle \text{hd}, \text{tl} \rangle} & \mathbb{R} \times \mathbb{R}^\omega \\
 \uparrow [] & & \uparrow \text{id} \times [] \\
 \mathbb{R} & \xrightarrow{\langle \text{id}, \underline{0} \cdot ! \rangle} & \mathbb{R} \times \mathbb{R} \\
 \uparrow + & & \uparrow \text{id} \times + \\
 \mathbb{R} \times \mathbb{R} & \xrightarrow{\langle +, \langle \underline{0} \cdot !, \underline{0} \cdot ! \rangle \rangle} & \mathbb{R} \times (\mathbb{R} \times \mathbb{R})
 \end{array}$$



Step 2: Then use **fusion**

Proofs By Calculation

$$\begin{aligned} & [] \cdot + = [(\langle +, \langle \underline{0} \cdot !, \underline{0} \cdot ! \rangle \rangle)] \\ \equiv & \quad \{ \text{[] definition} \} \\ & [(\langle \text{id}, \underline{0} \cdot ! \rangle)] \cdot + = [(\langle +, \langle \underline{0} \cdot !, \underline{0} \cdot ! \rangle \rangle)] \\ \Leftarrow & \quad \{ \text{coinductive fusion law} \} \\ & \langle \text{id}, \underline{0} \cdot ! \rangle \cdot + = (\text{id} \times +) \cdot \langle +, \langle \underline{0} \cdot !, \underline{0} \cdot ! \rangle \rangle \\ \equiv & \quad \{ \times\text{-fusion and absorption laws} \} \\ & \langle +, \underline{0} \cdot ! \cdot + \rangle = \langle +, + \cdot \langle \underline{0}, \underline{0} \rangle \cdot ! \rangle \\ \equiv & \quad \{ \text{arithmetic} \} \\ & \langle +, \underline{0} \cdot ! \cdot + \rangle = \langle +, \underline{0} \cdot ! \rangle \\ \equiv & \quad \{ \text{!-universal law} \} \\ & \langle +, \underline{0} \cdot ! \rangle = \langle +, \underline{0} \cdot ! \rangle \end{aligned}$$

Proofs By Calculation

And now the main result:

$$\begin{aligned} & + \cdot ([] \times []) = [] \cdot + \\ \equiv & \quad \{ + \text{ definition and lemma above } \} \\ & [(\langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle)] \cdot ([] \times []) = [(\langle +, \langle \underline{0} \cdot !, \underline{0} \cdot ! \rangle \rangle)] \\ \Leftarrow & \quad \{ \text{coinductive fusion laws} \} \\ & \langle + \cdot (\text{hd} \times \text{hd}), \text{tl} \times \text{tl} \rangle \cdot ([] \times []) = (\text{id} \times ([] \times [])) \cdot \langle +, \langle \underline{0} \cdot !, \underline{0} \cdot ! \rangle \rangle \\ \equiv & \quad \{ \times\text{-fusion and absorption laws} \} \\ & \langle + \cdot (\text{hd} \times \text{hd}) \cdot ([] \times []), (\text{tl} \times \text{tl}) \cdot ([] \times []) \rangle = \langle +, ([] \times []) \cdot \langle \underline{0} \cdot !, \underline{0} \cdot ! \rangle \rangle \\ \equiv & \quad \{ \text{functoriality and } \times\text{-fusion law} \} \\ & \langle + \cdot (\text{hd} \cdot [] \times \text{hd} \cdot []), (\text{tl} \cdot [] \times \text{tl} \cdot []) \rangle = \langle +, ([] \cdot \underline{0} \cdot ! \times [] \cdot \underline{0} \cdot !) \rangle \end{aligned}$$

Proofs By Calculation

$$\begin{aligned} & \langle + \cdot (\text{hd} \cdot [] \times \text{hd} \cdot []), (\text{tl} \cdot [] \times \text{tl} \cdot []) \rangle = \langle +, ([] \cdot \underline{0}! \times [] \cdot \underline{0}!) \rangle \\ \equiv & \quad \{ \text{previous lemma} \} \\ & \langle + \cdot (\text{id} \times \text{id}), [] \cdot \underline{0}! \times [] \cdot \underline{0}! \rangle = \langle +, ([] \cdot \underline{0}! \times [] \cdot \underline{0}!) \rangle \\ \equiv & \quad \{ \text{identity} \} \\ & \text{true} \end{aligned}$$

Alternative (easier!) proof:

Resorting to previous results, show that **canonical** (final) observations of both streamas are equal

Proofs By Calculation

$$\begin{aligned} & \text{hd } [r + s] \quad \text{and} \quad \text{tl } [r + s] \\ = & \quad \{ \text{hd } [r] = r \quad \text{and} \quad \text{tl } [r] = [0] \} \\ & r + s \quad \text{and} \quad [0] \\ = & \quad \{ \text{previous lemma} \} \\ & r + s \quad \text{and} \quad [0] + [0] \\ = & \quad \{ \text{hd } [r] = r \quad \text{and} \quad \text{tl } [r] = [0] \} \\ & \text{hd } r + \text{hd } s \quad \text{and} \quad \text{tl } [r] + \text{tl } [s] \end{aligned}$$

Proofs By Calculation

Example: map_f and generic laws

$$\text{map}_{f \cdot g} = \text{map}_f \cdot \text{map}_g$$

defining map_f as follows:

$$\begin{array}{ccccc} B^\omega & \xrightarrow{\omega_B} & B \times B^\omega \\ \text{map}_f \uparrow & & \uparrow \text{id} \times \text{map}_f \\ A^\omega & \xrightarrow{\omega_A} A \times A^\omega \xrightarrow{f \times \text{id}} & A \times B^\omega \end{array}$$

Proofs By Calculation

$$\begin{aligned} & \text{map}_{f \cdot g} = \text{map}_f \cdot \text{map}_g \\ \equiv & \quad \{ \text{map definition} \} \\ & [((f \cdot g) \times \text{id}) \cdot \omega] = [(f \times \text{id}) \cdot \omega] \cdot \text{map}_g \\ \Leftarrow & \quad \{ \text{coinduction fusion law} \} \\ & (f \times \text{id}) \cdot \omega \cdot \text{map}_g = (\text{id} \times \text{map}_g) \cdot ((f \cdot g) \times \text{id}) \cdot \omega \\ \equiv & \quad \{ \text{coinduction reflection law} \} \\ & (f \times \text{id}) \cdot (\text{id} \times \text{map}_g) \cdot (g \times \text{id}) \cdot \omega = (\text{id} \times \text{map}_g) \cdot ((f \cdot g) \times \text{id}) \cdot \omega \\ \equiv & \quad \{ \text{functoriality} \} \\ & ((f \cdot g) \times \text{map}_g) \cdot \omega = ((f \cdot g) \times \text{map}_g) \cdot \omega \end{aligned}$$

Proofs By Calculation

but this is just an instance of a **generic** result:

$$\text{map}_T (g \cdot f) = \text{map}_T g \cdot \text{map}_T f$$

$$\begin{array}{ccccc}
 \nu_{T_B} & \xrightarrow{\omega_{T_B}} & T_B \nu_{T_B} \\
 \uparrow \text{map}_T f & & \uparrow T_B \text{map}_T f \\
 \nu_{T_A} & \xrightarrow{\omega_{T_A}} T_A \nu_{T_A} \xrightarrow{T(f, \text{id}_{\text{map}_T A})} T_B \nu_{T_A}
 \end{array}$$

Proofs By Calculation

In general one also gets:

$$\begin{aligned}\text{map}_T \text{id}_A &= \text{id}_{\text{map}_T A} \\ \text{map}_T f \cdot [(p)]_T &= [(T(f, \text{id}) \cdot p)]_T\end{aligned}$$



function `map` extends to a functor mapping a set A into the **behaviour space** of T -coalgebras **parametric** on A



the last equation acts as an **absorption** law for coinductive extension

Proofs By Calculation

Example: Lambek's Lemma

The dynamics of the final coalgebra is an isomorphism

proof idea:



Assume the existence of an inverse α_T to $\omega_T : \nu_T \longrightarrow T\nu_T$.
Then, $\alpha_T \cdot \omega_T = \text{id}_{\nu_T}$ and $\omega_T \cdot \alpha_T = \text{id}_{T\nu_T}$



Take one of this requirements and use it to **conjecture** a definition for α_T (or an **implementation** ...)

Note the use of the **reflection** law to introduce an anamorphism in the calculation, instead of eliminating one



Then check the validity of this conjecture by verifying with it the other requirement

Proofs By Calculation

$$\begin{aligned} & \alpha_T \cdot \omega_T = \text{id}_{\nu_T} \\ \equiv & \quad \{ \text{reflection law} \} \\ & \alpha_T \cdot \omega_T = [(\omega_T)] \\ \equiv & \quad \{ \text{universal law} \} \\ & \omega_T \cdot \alpha_T \cdot \omega_T = T(\alpha_T \cdot \omega_T) \cdot \omega_T \\ \equiv & \quad \{ \text{as a functor } T \text{ preserves composition} \} \\ & \omega_T \cdot \alpha_T \cdot \omega_T = T\alpha_T \cdot T\omega_T \cdot \omega_T \\ \equiv & \quad \{ \text{cancel } \omega_T \text{ from both sides \& universal law} \} \\ & \alpha_T = [(T\omega_T)] \end{aligned}$$

Proofs By Calculation

$$\begin{aligned} & \omega_T \cdot \alpha_T \\ = & \quad \{ \text{replace } \alpha_T \text{ by the derived conjecture} \} \\ & \omega_T \cdot \llbracket T\omega_T \rrbracket \\ = & \quad \{ \llbracket T\omega_T \rrbracket \text{ is a morphism} \} \\ & T\llbracket T\omega_T \rrbracket \cdot T\omega_T \\ = & \quad \{ \text{as a functor } T \text{ preserves composition} \} \\ & T(\llbracket T\omega_T \rrbracket \cdot \omega_T) \\ = & \quad \{ \text{just proved} \} \\ & T \text{id}_{\nu_T} \\ = & \quad \{ \text{as a functor } T \text{ preserves identities} \} \\ & \text{id}_{(T\text{id}_{\nu_T})} \end{aligned}$$

Proofs By Calculation

Warning!

Coinduction is **not enough** either as a **definition** or a **proof** principle

Example: the dual to **primitive recursion**

It allows for the final result to be either generated in successive steps or 'all at once' without recursion. Therefore, the codomain of the source 'coalgebra' becomes the sum of its carrier with the coinductive type itself.

Proofs By Calculation

$$\begin{array}{ccc}
 \nu_T & \xrightarrow{\omega_T} & T \nu_T \\
 \text{apo}_T p \uparrow & & \uparrow T [\text{apo}_T p, \text{id}] \\
 X & \xrightarrow{p} & T (X + \nu_T)
 \end{array}$$

which entails the following universal property

$$h = \text{apo}_T p \iff \omega_T \cdot h = T [h, \text{id}] \cdot p$$

Examples:

$\text{apo} \langle f \cdot \text{hd}, \iota_2 \cdot \text{tl} \rangle$

$\text{apo} \langle f, [s, r] \cdot \phi? \rangle$

Proofs By Calculation

Laws

reflection

$$\text{apo}_T (T \iota_1 \cdot \omega_T) = \text{id}$$

weak fusion

$$\text{apo}_T f \cdot h = \text{apo}_T g \iff f \cdot h = T(h + \text{id}) \cdot g$$

ana as apo

$$[[f]]_T = \text{apo}_T (T \iota_2 \cdot f)$$

any as apo

$$h = \text{apo}_T (T \iota_2 \cdot \omega_T \cdot h)$$

Proofs By Calculation

Different coinduction schemes can be obtained in a systematic way through the notion of

bialgebra for a distributive law λ

[Turi, Plotkin 97, Lenisa 98, Bartel 04]

$$\begin{array}{ccccc} TU & \xrightarrow{\beta} & U & \xrightarrow{\alpha} & BU \\ \tau\alpha \downarrow & & & & \uparrow B\beta \\ TBU & \xrightarrow{\lambda_U} & & & BTU \end{array}$$

Proofs By Calculation

An example of λ -corecursion

$$\begin{array}{ccc}
 \mathbb{R}^\omega & \xrightarrow{\omega} & \mathbb{R} \times \mathbb{R}^\omega \\
 \uparrow [g] & & \uparrow \text{id} \times + \\
 & & \mathbb{R} \times (\mathbb{R}^\omega \times \mathbb{R}^\omega) \\
 & & \uparrow \text{id} \times [g] \times [g] \\
 \mathbf{1} & \xrightarrow{g = \langle \underline{1}, \Delta \rangle} & \mathbb{R} \times (\mathbf{1} \times \mathbf{1})
 \end{array}$$

$$[g] \cdot \langle \text{hd}, \text{tl} \rangle = \langle \underline{1}, + \cdot \langle [g], [g] \rangle \rangle$$

corresponding to stream

$$s = 1 : (s + s) = [1, 2, 4, 8, \dots]$$

Proofs By Calculation

An example of λ -corecursion



$$B = \mathbb{R} \times \text{Id}$$



$$T = \text{Id} \times \text{Id}$$



$\beta = +$ the **unique** T -coalgebra yielding a λ -bialgebra structure



Note the specification is bi-recursive and requires processing **after** recursive invocation: therefore it cannot be a B -coalgebra



Finally note that the example is illustrative but not essential because

$$s = [(\langle \text{id}, \times_2 \rangle)] \cdot \underline{1}$$

Proofs By Calculation

Development of a **calculational toolkit** for λ -coinduction

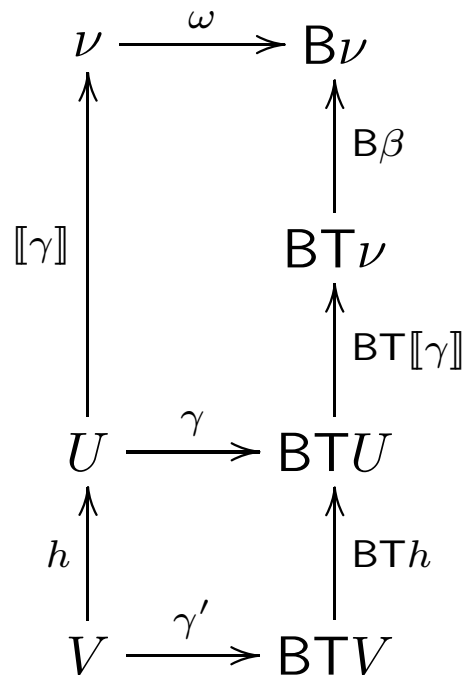
based on the following **universal** law

$$k = \llbracket \gamma \rrbracket \equiv \omega \cdot k = B(\beta \cdot Tk) \cdot \gamma$$

to be compared to [Bartels 04] characterisation of **bisimulation-up-to- λ**

Proofs By Calculation

Example: a **fusion** law



$$\llbracket \gamma \rrbracket \cdot h = \llbracket \gamma' \rrbracket \Leftarrow \gamma \cdot h = BTh \cdot \gamma'$$

CS: Process Algebras

1. Introduction
2. Sequences & Streams
3. Coinductive Modelling
4. Proofs by Calculation
5. Case study: Revisiting Process Algebras
6. Conclusions

CS: Process Algebra

Aims

Apply coinductive reasoning principles and calculational style to the design of process calculi

... relying on the representation of processes as inhabitants of final coalgebras for suitable Set endofunctors



Uniform treatment of processes and data



Direct derivation of functional **interpreters** for process calculi



Genericity: structural aspects of the underlying **behaviour model** clearly separated from the **interaction** structure which defines the synchronisation discipline



Proofs in a calculational (basically **equational** and **pointfree**) style

CS: Process Algebra

The Design Problem

Operational semantics given in terms of an action-indexed transition relation $\xrightarrow{a}$ over processes

... witnessing the collection of actions in which a process gets committed and the resulting ‘continuations’



‘collection of’ $\leadsto$ **behaviour model B**



‘transition relation’ $p \xrightarrow{a} q$ admits different interpretations, e.g.,

- *Active*: ‘ p evolves to q by performing an action a , both q and a being solely determined by p ’
- *Reactive*: ‘ p reacts to an external stimulus a by evolving to q ’

$\leadsto$ different **shapes**



‘actions’ $\leadsto$ a variety of **interaction disciplines**

CS: Process Algebra

The Approach



Represent processes as inhabitants of the carrier of the final coalgebra for suitable Set functors

$$\omega : \nu \longrightarrow T \nu$$



so that **definition** of process' combinators and **proofs** of their properties rely on **finality**



which entails direct derivations of functional **interpreters** for (classical) process calculi

CS: Process Algebra



Let ν_T denote (the) **final** universe of processes of **shape** T



and $\square$ a process **combinator** such that

$$\begin{array}{ccc}
 \nu_T & \xrightarrow{\omega_T} & T \nu_T \\
 \square \uparrow & & \uparrow T \square \\
 U & \xrightarrow{\alpha_{\square}} & T U
 \end{array}$$

commutes for a **gene** coalgebra $\alpha_{\square}$ which specifies its **one-step dynamics**, i.e.

$$\square = [(\alpha_{\square})]_T$$

said its **anamorphism** or **coinductive extension**

CS: Process Algebra

Adopt

$$\mathsf{T} = \mathcal{P}(\mathit{Act} \times \mathsf{Id})$$



corresponding to a **active** interpretation: processes are elements of the final coalgebra $\omega : \nu \longrightarrow \mathcal{P}(\mathit{Act} \times \nu)$ and the transition relation is defined as

$$p \xrightarrow{a} q \iff \langle a, q \rangle \in \omega p$$



but restricting to image-finite processes

CS: Process Algebra

plus



a definition of the **interaction** discipline



and **dynamic** and **static** process combinators



... **parametric** on such a discipline

CS: Process Algebra

Dynamic Combinators



consumed on action occurrence: disappear from the expression representing the process continuation.

inaction

$$\omega \cdot \text{nil} = \underline{\emptyset}$$

prefix

$$\omega \cdot a. = \text{sing} \cdot \text{label}_a$$

choice

$$\omega \cdot + = \cup \cdot (\omega \times \omega)$$

where $\text{sing} = \lambda x. \{x\}$ and $\text{label}_a = \lambda x. \langle a, x \rangle$.

CS: Process Algebra

Laws

$\langle \nu; +, \text{nil} \rangle$ is an Abelian idempotent monoid

$$(p + q) + r = p + (q + r) \quad \equiv \quad + \cdot (+ \times \text{id}) = + \cdot (\text{id} \times +) \cdot a$$

$$p + \text{nil} = p \quad \equiv \quad + \cdot (\text{id} \times \text{nil}) \cdot l^\circ = \text{id}$$

$$p + p = p \quad \equiv \quad + \cdot \Delta = \text{id}$$

$$p + q = q + p \quad \equiv \quad + \cdot s = +$$

CS: Process Algebra

Simple Equational Proof: $+ \cdot s = +$

$$\begin{aligned} & \omega \cdot + \cdot s \\ \equiv & \quad \{ \text{definition} \} \\ & \cup \cdot (\omega \times \omega) \cdot s \\ \equiv & \quad \{ s \text{ natural} \} \\ & \cup \cdot s \cdot (\omega \times \omega) \\ \equiv & \quad \{ \cup \text{ commutative} \} \\ & \cup \cdot (\omega \times \omega) \\ \equiv & \quad \{ \text{definition} \} \\ & \omega \cdot + \end{aligned}$$

CS: Process Algebra

Static Combinators



Persistence through action occurrence enforces (co)recursive definition



Therefore, arise as *anamorphisms* generated by suitable ‘gene’ coalgebras



In general, depend on the calculus *interaction structure*

CS: Process Algebra

Interaction Structure

$\langle Act; \theta, 1 \rangle$ abelian positive monoid with a zero 0



θ determines the interaction discipline



0 represents the absence of interaction: for all $a \in Act$, $a\theta 0 = 0$.



1 technical role to equip the behaviour functor with a monadic structure



being positive means $a\theta a' = 1$ iff $a = a' = 1$

CS: Process Algebra

Interaction Structure: Examples



Co-occurrence: $a\theta b = \langle a, b \rangle$, for all $a, b \in Act$ ($\neq 0, 1$)

CS: Process Algebra

Interaction Structure: Examples



CCS [Milner89]:

- L has an **involution** ($\bar{a}$, for $a \in L$)
- $a\theta\bar{a} = \tau$ $a\theta 1 = a$ $a\theta b = 0$, for $b \neq \bar{a}$



CSP [Hoare85]:

$a\theta a = a$, $a\theta 1 = a$ and $a\theta b = 0$ in all other cases



...

CS: Process Algebra

Renaming

Once fixed an **interaction structure** any homomorphism

$$f : Act \longrightarrow Act$$

lifts to a **renaming** combinator

$$[f] = [(\alpha_{[f]})]$$

defined as the coinductive extension of

$$\alpha_{[f]} = \nu \xrightarrow{\omega} \mathcal{P}(Act \times \nu) \xrightarrow{\mathcal{P}(f \times \text{id})} \mathcal{P}(Act \times \nu)$$

CS: Process Algebra

Renaming

$$\begin{array}{ccccc} \nu & \xrightarrow{\omega} & & \mathcal{P}(\text{Act} \times \nu) & \\ \uparrow [f] & & & \uparrow \mathcal{P}([f] \times \text{id}) & \\ \nu & \xrightarrow{\omega} & \mathcal{P}(\text{Act} \times \nu) & \xrightarrow{\mathcal{P}(f \times \text{id})} & \mathcal{P}(\text{Act} \times \nu) \end{array}$$

CS: Process Algebra

Restriction

$$\backslash_K = [(\alpha \backslash_K)]$$

i.e., the coinductive extension of

$$\alpha \backslash_K = \nu \xrightarrow{\omega} \mathcal{P}(A \times \nu) \xrightarrow{\text{filter}_K} \mathcal{P}(A \times \nu)$$

where $\text{filter}_K = \lambda s . \{t \in s \mid \pi_1 t \notin K\}$

CS: Process Algebra

Interleaving

$$||| = [(\alpha_{|||})]$$

defined as the coinductive extension of

$$\begin{aligned}\alpha_{|||} &= \nu \times \nu \xrightarrow{\Delta} (\nu \times \nu) \times (\nu \times \nu) \\ &\xrightarrow{(\omega \times \text{id}) \times (\text{id} \times \omega)} (\mathcal{P}(\text{Act} \times \nu) \times \nu) \times (\nu \times \mathcal{P}(\text{Act} \times \nu)) \\ &\xrightarrow{\tau_r \times \tau_l} \mathcal{P}(\text{Act} \times (\nu \times \nu)) \times \mathcal{P}(\text{Act} \times (\nu \times \nu)) \xrightarrow{\cup} \mathcal{P}(\text{Act} \times (\nu \times \nu))\end{aligned}$$

CS: Process Algebra

Laws: restriction and renaming

$$\backslash_K \cdot + = + \cdot (\backslash_K \times \backslash_K)$$

$$\backslash_K \cdot ||| = ||| \cdot (\backslash_K \times \backslash_K)$$

$$\backslash_K \cdot a. = (a \in K \rightarrow \text{nil}, a. \cdot \backslash_K)$$

$$\backslash_K \cdot \backslash_K = \backslash_K$$

$$[\text{id}] = \text{id}$$

$$[f] \cdot [f'] = [f \cdot f']$$

$$[f] \cdot a. = (f \ a). \cdot [f]$$

$$[f] \cdot + = + \cdot ([f] \times [f])$$

$$[f] \cdot ||| = ||| \cdot ([f] \times [f])$$

CS: Process Algebra

Proofs by Fusion: $\parallel$ commutativity

By definition

$$\parallel \cdot s = \parallel$$

equivalences

$$[(\alpha_{\parallel})] \cdot s = [(\alpha_{\parallel})]$$

which, by fusion, is implied by

$$\alpha_{\parallel} \cdot s = \mathcal{P}(\text{id} \times s) \cdot \alpha_{\parallel}$$

CS: Process Algebra

$$\begin{aligned} & \alpha_{|||} \cdot s \\ &= \cup \cdot (\tau_r \times \tau_l) \cdot ((\omega \times \text{id}) \times (\text{id} \times \omega)) \cdot \Delta \cdot s \\ &= \cup \cdot (\tau_r \times \tau_l) \cdot ((\omega \times \text{id}) \times (\text{id} \times \omega)) \cdot (s \times s) \cdot \Delta \\ &= \cup \cdot (\tau_r \cdot s \times \tau_l \cdot s) \cdot ((\text{id} \times \omega) \times (\omega \times \text{id})) \cdot \Delta \\ &= \cup \cdot (\mathcal{P}(\text{id} \times s) \cdot \tau_l \times \mathcal{P}(\text{id} \times s) \cdot \tau_r) \cdot ((\text{id} \times \omega) \times (\omega \times \text{id})) \cdot \Delta \\ &= \cup \cdot (\mathcal{P}(\text{id} \times s) \times \mathcal{P}(\text{id} \times s)) \cdot (\tau_l \times \tau_r) \cdot ((\text{id} \times \omega) \times (\omega \times \text{id})) \cdot \Delta \\ &= \mathcal{P}(\text{id} \times s) \cdot \cup \cdot s \cdot (\tau_l \times \tau_r) \cdot ((\text{id} \times \omega) \times (\omega \times \text{id})) \cdot \Delta \\ &= \mathcal{P}(\text{id} \times s) \cdot \cup \cdot (\tau_r \times \tau_l) \cdot ((\omega \times \text{id}) \times (\text{id} \times \omega)) \cdot s \cdot \Delta \\ &= \mathcal{P}(\text{id} \times s) \cdot \cup \cdot (\tau_r \times \tau_l) \cdot ((\omega \times \text{id}) \times (\text{id} \times \omega)) \cdot \Delta \\ &= \mathcal{P}(\text{id} \times s) \cdot \alpha_{|||} \end{aligned}$$

CS: Process Algebra

Synchronous Product



models the simultaneous execution of its two arguments



at each step the resulting action is determined by the interaction structure for the calculus $\parallel = \llbracket \alpha_{\parallel} \rrbracket$

$$\begin{aligned} \alpha_{\otimes} &= \nu \times \nu \xrightarrow{(\omega \times \omega)} \mathcal{P}(\text{Act} \times \nu) \times \mathcal{P}(\text{Act} \times \nu) \\ &\xrightarrow{\delta_r} \mathcal{P}(\text{Act} \times (\nu \times \nu)) \xrightarrow{\text{sel}} \mathcal{P}(\text{Act} \times (\nu \times \nu)) \end{aligned}$$

where $\text{sel} = \text{filter}_{\{0\}}$ filters out all synchronisation failures

CS: Process Algebra



interaction is catered by δ_r — the distributive law for the strong monad $\mathcal{P}(Act \times Id)$

$$\delta_r^{\mathcal{P}(Act \times Id)} \langle c_1, c_2 \rangle = \{ \langle a' \theta a, \langle p, p' \rangle \rangle \mid \langle a, p \rangle \in c_1 \wedge \langle a', p' \rangle \in c_2 \}$$



the underlying Kleisli composition involves the application of the monad multiplication to ‘flatten’ the result which, for a **monoid monad**, requires the suitable application of the underlying monoidal operation: here this fixes the **interaction discipline**

CS: Process Algebra

$$\begin{aligned}
 & \delta_r^{\mathcal{P}(Act \times Id)} \\
 = & \quad \{ \delta_r \text{ definition} \} \\
 & \tau_r^{\mathcal{P}(Act \times Id)} \bullet \tau_l^{\mathcal{P}(Act \times Id)} \\
 = & \quad \{ \bullet \text{ definition} \} \\
 & \mu^{\mathcal{P}(Act \times Id)} \cdot \mathcal{P}(id \times \tau_r^{\mathcal{P}(Act \times Id)}) \cdot \tau_l^{\mathcal{P}(Act \times Id)} \\
 = & \quad \{ \mu \text{ for } \mathcal{P}(Act \times Id) \}
 \end{aligned}$$

$$\eta^{B_1 \ B_2} = \eta^{B_1} \cdot \eta^{B_2}$$

$$\mu^{B_1 \ B_2} = B_1 \ \mu^{B_2} \cdot \mu^{B_1} \cdot B_1 \ \gamma$$

CS: Process Algebra

with $\gamma : B_2 \times B_1 \longrightarrow B_1 \times B_2$

in this case $\gamma = \tau_l^B : Act \times \mathcal{P}Id \longrightarrow \mathcal{P}(Act \times Id)$

Then,

$$\begin{aligned}
 & \mathcal{P}\mu^{(Act \times Id)} \cdot \mu^{\mathcal{P}} \cdot \mathcal{P}\tau_l^{\mathcal{P}} \cdot \mathcal{P}(\text{id} \times \tau_r^{\mathcal{P}(Act \times Id)}) \cdot \tau_l^{\mathcal{P}(Act \times Id)} \\
 = & \quad \{ \mu \text{ for } (Act \times Id) \} \\
 & \mathcal{P}((\theta \times \text{id}) \cdot a^\circ) \cdot \mu^{\mathcal{P}} \cdot \mathcal{P}\tau_l^{\mathcal{P}} \cdot \mathcal{P}(\text{id} \times \tau_r^{\mathcal{P}(Act \times Id)}) \cdot \tau_l^{\mathcal{P}(Act \times Id)} \\
 = & \quad \{ \mu^{\mathcal{P}} \text{ natural and definition} \} \\
 & \bigcup \cdot \mathcal{P}(\mathcal{P}((\theta \times \text{id}) \cdot a^\circ) \cdot \tau_l^{\mathcal{P}}) \cdot \mathcal{P}(\text{id} \times \tau_r^{\mathcal{P}(Act \times Id)}) \cdot \tau_l^{\mathcal{P}(Act \times Id)}
 \end{aligned}$$

CS: Process Algebra

$$\mathcal{P}(Act \times \nu) \times \mathcal{P}(Act \times \nu) \xrightarrow{\tau_l^{\mathcal{P}}} \mathcal{P}(\mathcal{P}(Act \times \nu) \times (Act \times \nu))$$

$$\xrightarrow{\tau_l^{Act \times Id}} \mathcal{P}(Act \times (\mathcal{P}(Act \times \nu) \times \nu))$$

$$\xrightarrow{\mathcal{P}(id \times \tau_r^{\mathcal{P}})} \mathcal{P}(Act \times \mathcal{P}((Act \times \nu) \times \nu))$$

$$\xrightarrow{\mathcal{P}(id \times \mathcal{P}\tau_r^{Act \times Id})} \mathcal{P}(Act \times \mathcal{P}(Act \times (\nu \times \nu)))$$

$$\xrightarrow{\tau_l^{\mathcal{P}}} \mathcal{P}\mathcal{P}(Act \times (Act \times (\nu \times \nu)))$$

CS: Process Algebra

$$\xrightarrow{\mathcal{PP}a^\circ} \mathcal{PP}((Act \times Act) \times (\nu \times \nu))$$

$$\xrightarrow{\mathcal{PP}(\theta \times \text{id})} \mathcal{PP}(Act \times (\nu \times \nu))$$

$$\xrightarrow{\cup} \mathcal{P}(Act \times (\nu \times \nu))$$

CS: Process Algebra

Parallel Composition



combines **interleaving** with **synchronous product**



... to cater for both the **individual** derivation plus the **synchronisations** allowed by θ



... but performed at the **genes** level

$$\begin{aligned}\alpha| &= \nu \times \nu \xrightarrow{\Delta} (\nu \times \nu) \times (\nu \times \nu) \\ &\xrightarrow{(\alpha_{||} \times \alpha_{\otimes})} \mathcal{P}(\text{Act} \times (\nu \times \nu)) \times \mathcal{P}(\text{Act} \times (\nu \times \nu)) \\ &\xrightarrow{\cup} \mathcal{P}(\text{Act} \times (\nu \times \nu))\end{aligned}$$

CS: Process Algebra

Laws ...

But now focus on **proofs!**

CS: Process Algebra



Replace the explicit construction of **bisimulations** by **equational reasoning**



essentially **pointfree** ...



... used in functional programming (cf, **Bird-Meertens** school)

dynamic combinators



simple equational proofs

static combinators



proofs by **fusion**

conditional laws

require

a form of **conditional fusion**

CS: Process Algebra

How can conditional laws be derived?

For example, the following classic law in Ccs:

$$(p \mid q) \setminus_K = p \setminus_K \mid q \setminus_K \quad \Leftarrow \mathcal{L}(p) \cap \overline{\mathcal{L}(p)} \cap (K \cup \overline{K}) = \emptyset$$

or the simple fact that renaming absent actions has no effect

CS: Process Algebra

In several cases, but not in all, renaming with $f = \{b/a\}$ has no effect.

$$\begin{aligned} [f] &= \text{id} \\ \equiv & \quad \{ \text{renaming definition and ana reflection} \} \\ [[\alpha_{[f]}]] &= [[\omega]] \\ \Leftarrow & \quad \{ \text{ana fusion and identity} \} \\ \alpha_{[f]} &= \omega \end{aligned}$$

Intuitively, the last equality holds if a does not show up as an action in the immediate continuations of the process:

CS: Process Algebra

$$\phi = \text{=}\emptyset \cdot \cap \cdot \langle \mathcal{P}\pi_1 \cdot \omega, \text{sing} \cdot \underline{a} \rangle$$

But ϕ is stated as a **local condition**

To be taken as a sufficient condition for $[f] = \text{id}$, it has to be lifted to **β -invariant**.

CS: Process Algebra

The **conditional fusion** theorem [Barbosa01]

Let α and β be T-coalgebras and ϕ a predicate on the carrier of β .
Then

$$(\phi \Rightarrow (\alpha \cdot h = \top h \cdot \beta)) \Rightarrow (\Box_{\beta} \phi \Rightarrow ([\alpha]_{\top} \cdot h = [\beta]_{\top}))$$

where $\Box_{\beta}$ is the *greatest* fix point of $\lambda_x . \phi \cap \circ_{\beta} x$ [Jacobs99]

Informally, $\Box_{\beta} \phi$ reads:

ϕ holds now and in all successor states under β dynamics

CS: Process Algebra

A predicate $\phi : U \longrightarrow \mathbf{2}$ over the carrier of a T-coalgebra $\langle U, p : U \longrightarrow \mathsf{T} U \rangle$ is a p -invariant if it is closed under the coalgebra dynamics:

$$\phi \Rightarrow \circ_p \phi$$

where $\circ_p$:

$$(\circ_p \phi) u \iff \forall u' \in \mathsf{T} p u . \phi u'$$

Informally:

$\circ_p \phi$ holds in all states whose immediate successors under p satisfy ϕ

CS: Process Algebra

Or, in terms of **predicate lifting** (cf, [Jacobs99])

$$\circ_p \phi = \{u \in U \mid p \ u \in (\phi)^T\}$$

$\top$	$(\phi)^T$
Id	ϕ
$\underline{K}$	K
$\top_1 \times \top_2$	$\{\langle x_1, x_2 \rangle \mid x_1 \in (\phi)^{\top_1} \wedge x_2 \in (\phi)^{\top_2}\}$
$\top_1 + \top_2$	$\{\iota_1 \ x_1 \mid x_1 \in (\phi)^{\top_1}\} \cup \{\iota_2 \ x_2 \mid x_2 \in (\phi)^{\top_2}\}$
$\top^K$	$\{f \mid \forall k \in K. \ f \ k \in (\phi)^T\}$
$\mathcal{P}\top$	$\{t \mid \forall x. \ x \in t \Rightarrow x \in (\phi)^T\}$

CS: Process Algebra

Proof

Let X be the carrier of β and i_ϕ the embedding of the subset of X classified by ϕ , i.e., $\phi \cdot i_\phi = \underline{true}!$. Recall that any β -invariant induces a subcoalgebra β' : therefore, $i_{\Box_\beta \phi}$ becomes a comorphism from β' to β . Then

$$\begin{aligned} & \phi \Rightarrow (\alpha \cdot h = \top h \cdot \beta) \\ \equiv & \quad \{ i_\phi \text{ definition} \} \\ & \alpha \cdot h \cdot i_\phi = \top h \cdot \beta \cdot i_\phi \\ \Rightarrow & \quad \{ \Box_\beta \phi \subseteq \phi \} \\ & \alpha \cdot h \cdot i_{\Box_\beta \phi} = \top h \cdot \beta \cdot i_{\Box_\beta \phi} \\ \equiv & \quad \{ i_{\Box_\beta \phi} \text{ is a comorphism from } \beta' \text{ to } \beta \} \end{aligned}$$

CS: Process Algebra

$$\begin{aligned} & \alpha \cdot h \cdot i_{\square_{\beta} \phi} = \top h \cdot \top i_{\square_{\beta} \phi} \cdot \beta' \\ \equiv & \quad \{ \text{functoriality} \} \\ & \alpha \cdot h \cdot i_{\square_{\beta} \phi} = \top (h \cdot i_{\square_{\beta} \phi}) \cdot \beta' \\ \equiv & \quad \{ \text{fusion} \} \\ & [[\alpha]]_{\top} \cdot h \cdot i_{\square_{\beta} \phi} = [[\beta']]_{\top} \\ \equiv & \quad \{ i_{\square_{\beta} \phi} \text{ being a comorphism implies } [[\beta']]_{\top} = [[\beta]]_{\top} \cdot i_{\square_{\beta} \phi} \} \\ & [[\alpha]]_{\top} \cdot h \cdot i_{\square_{\beta} \phi} = [[\beta]]_{\top} \cdot i_{\square_{\beta} \phi} \\ \equiv & \quad \{ i_{\square_{\beta} \phi} \text{ definition} \} \\ & \square_{\beta} \phi \Rightarrow ([[\alpha]]_{\top} \cdot h = [[\beta]]_{\top}) \end{aligned}$$

CS: Process Algebra

Unfolding $\Box_\beta \phi$

$$[\{b/a\}] = \text{id} \Leftarrow \Box_\omega \phi$$

for $\Box_\omega \phi$ the greatest fixpoint of

$$\Phi = \lambda x . \phi \cap \circ_\omega x$$

I.e., by Tarski theorem, the union of all post-fixpoints of Φ :

$$\Box_\omega \phi = \bigcup \{s \in \mathcal{P}\nu \mid s \subseteq \phi \cap \circ_\omega s\}$$

CS: Process Algebra

Then,

$$\begin{aligned} p \in s &\Rightarrow p \in \phi \wedge p \in \circ_{\omega} s \\ &\equiv p \in \phi \wedge p \in \{x \in \nu \mid \omega x \in (s)^{\mathcal{P}(Act \times \nu)}\} \\ &\equiv p \in \phi \wedge p \in \{x \in \nu \mid \omega x \in \{c \in \mathcal{P}(Act \times \nu) \mid \forall t \in Act \times \nu . t \in c \Rightarrow t \in (s)^{Act \times \nu}\}\} \\ &\equiv p \in \phi \wedge p \in \{x \in \nu \mid \omega x \in \{c \in \mathcal{P}(Act \times \nu) \mid \forall t \in Act \times \nu . t \in c \Rightarrow \pi_2 t \in s\}\} \\ &\equiv p \in \phi \wedge p \in \{x \in \nu \mid (\mathcal{P}\pi_2 \cdot \omega) x \in s\} \end{aligned}$$

Seen as a set, predicate ϕ is given by

$$\phi = \{x \in \nu \mid (\mathcal{P}\pi_1 \cdot \omega) x \cap \{a\} = \emptyset\}.$$

CS: Process Algebra

Therefore,

$$\Box_{\omega} \phi = \bigcup \{s \in \mathcal{P}\nu \mid \forall_{x \in \nu} . x \in s \Rightarrow ((\mathcal{P}\pi_1 \cdot \omega) \ x \cap \{a\} = \emptyset) \wedge ((\mathcal{P}\pi_2 \cdot \omega) \ x \in s)\}$$

In words: the set of all processes whose derivations never exhibit action a .

CS: Process Algebra

In Ccs,

$$[\{b/a\}] = \text{id} \Leftarrow \not\in_a \cdot \mathcal{L}$$

where $\not\in_a = \lambda x . a \notin x$

Going pointwise $p [\{b/a\}] = p \Leftarrow a, \bar{a} \notin \mathcal{L}(p)$

where $\mathcal{L}(p)$ is the **sort** of p defined **semantically**:

$$\mathcal{L}(p) = (\text{filter}_{Act-L} \cdot \mathcal{P}\pi_1 \cdot \bigcup \cdot \mathcal{P}\omega) ((\Box_\omega \text{ true}) p)$$

CS: Process Algebra

A Derivation Example

$$\backslash_K \cdot \otimes = \otimes \cdot (\backslash_K \times \backslash_K) \quad \Leftarrow \text{uniform_restriction}$$

The proof relies on an immediate consequence of fusion:

to prove the equality $\phi = \psi$ it is enough to show that both $\omega \cdot \phi = \top \phi \cdot \alpha$

and $\omega \cdot \psi = \top \psi \cdot \alpha$ hold

CS: Process Algebra

Unfolding RHS

$$\begin{aligned} & \omega \cdot \otimes \cdot (\backslash_K \times \backslash_K) \\ = & \quad \{ \text{comorphism, definition} \} \\ & \mathcal{P}(\text{id} \times \otimes) \cdot \text{sel} \cdot \delta_r \cdot (\omega \times \omega) \cdot (\backslash_K \times \backslash_K) \\ = & \quad \{ \text{functoriality} \} \\ & \mathcal{P}(\text{id} \times \otimes) \cdot \text{sel} \cdot \delta_r \cdot (\omega \cdot \backslash_K \times \omega \cdot \backslash_K) \\ = & \quad \{ \text{comorphism, functoriality} \} \\ & \mathcal{P}(\text{id} \times \otimes) \cdot \text{sel} \cdot \delta_r \cdot (\mathcal{P}(\text{id} \times \backslash_K) \times \mathcal{P}(\text{id} \times \backslash_K)) \cdot (\alpha_{\backslash_K} \times \alpha_{\backslash_K}) \\ = & \quad \{ \delta_r \text{ naturality} \} \end{aligned}$$

CS: Process Algebra

Unfolding RHS

$$\begin{aligned} & \mathcal{P}(\text{id} \times \otimes) \cdot \text{sel} \cdot \mathcal{P}(\text{id} \times (\backslash_K \times \backslash_K)) \cdot \delta_r \cdot (\alpha_{\backslash_K} \times \alpha_{\backslash_K}) \\ = & \quad \{ \text{sel definition, functoriality} \} \\ & \mathcal{P}(\text{id} \times \otimes \cdot (\backslash_K \times \backslash_K)) \cdot \text{sel} \cdot \delta_r \cdot (\alpha_{\backslash_K} \times \alpha_{\backslash_K}) \end{aligned}$$

CS: Process Algebra

Unfolding LHS

$$\begin{aligned} & \omega \cdot \backslash_K \cdot \otimes \\ = & \quad \{ \text{comorphism, definition} \} \\ & \mathcal{P}(\text{id} \times \backslash_K) \cdot \text{filter}_K \cdot \omega \cdot \otimes \\ = & \quad \{ \text{comorphism} \} \\ & \mathcal{P}(\text{id} \times \backslash_K) \cdot \text{filter}_K \cdot \mathcal{P}(\text{id} \times \otimes) \cdot \alpha_{\otimes} \\ = & \quad \{ \text{definition} \} \\ & \mathcal{P}(\text{id} \times \backslash_K) \cdot \text{filter}_K \cdot \mathcal{P}(\text{id} \times \otimes) \cdot \text{sel} \cdot \delta_r \cdot (\omega \times \omega) \\ = & \quad \{ \text{filter}_K \text{ naturality} \} \end{aligned}$$

CS: Process Algebra

Unfolding LHS

$$\begin{aligned} & \mathcal{P}(\text{id} \times \backslash_K) \cdot \mathcal{P}(\text{id} \times \otimes) \cdot \text{filter}_K \cdot \text{sel} \cdot \delta_r \cdot (\omega \times \omega) \\ \stackrel{?}{=} & \{ \star \} \\ & \mathcal{P}(\text{id} \times \backslash_K) \cdot \mathcal{P}(\text{id} \times \otimes) \cdot \text{sel} \cdot \delta_r \cdot (\text{filter}_K \times \text{filter}_K) \cdot (\omega \times \omega) \end{aligned}$$

CS: Process Algebra

Last step requires:

$$\text{filter}_K \cdot \text{sel} \cdot \delta_r = \text{sel} \cdot \delta_r \cdot (\text{filter}_K \times \text{filter}_K) \quad (*)$$

Then,

$$\begin{aligned} & \mathcal{P}(\text{id} \times \backslash_K) \cdot \mathcal{P}(\text{id} \times \otimes) \cdot \text{sel} \cdot \delta_r \cdot (\text{filter}_K \times \text{filter}_K) \cdot (\omega \times \omega) \\ = & \quad \{ \text{functoriality} \} \\ & \mathcal{P}(\text{id} \times (\backslash_K \cdot \otimes)) \cdot \text{sel} \cdot \delta_r \cdot (\text{filter}_K \cdot \omega \times \text{filter}_K \cdot \omega) \\ = & \quad \{ \text{definition} \} \\ & \mathcal{P}(\text{id} \times (\backslash_K \cdot \otimes)) \cdot \text{sel} \cdot \delta_r \cdot (\alpha_{\backslash_K} \times \alpha_{\backslash_K}) \end{aligned}$$

and conclude by **conditional fusion**, generating the suitable invariant

CS: Process Algebra

Unfolding (*) to find the invariant

$$\begin{aligned}(\text{filter}_K \cdot \text{sel} \cdot \delta_r) \langle c_1, c_2 \rangle &= \\&= (\text{filter}_K \cdot \text{sel}) \{ \langle a' \theta a, \langle p, p' \rangle \rangle \mid \langle a, p \rangle \in c_1 \wedge \langle a', p' \rangle \in c_2 \} \\&= \text{filter}_K \{ \langle a' \theta a, \langle p, p' \rangle \rangle \mid \langle a, p \rangle \in c_1 \wedge \langle a', p' \rangle \in c_2 \wedge a' \theta a \neq 0 \} \\&= \{ \langle a' \theta a, \langle p, p' \rangle \rangle \mid \langle a, p \rangle \in c_1 \wedge \langle a', p' \rangle \in c_2 \wedge a' \theta a \neq 0 \wedge a' \theta a \notin K \}\end{aligned}$$

and

$$\begin{aligned}(\text{sel} \cdot \delta_r \cdot (\text{filter}_K \times \text{filter}_K)) \langle c_1, c_2 \rangle &= \\&= (\text{sel} \cdot \delta_r) \langle \{ \langle a, p \rangle \in c_1 \mid a \notin K \}, \{ \langle a', p' \rangle \in c_2 \mid a' \notin K \} \rangle \\&= \text{sel} \{ \langle a' \theta a, \langle p, p' \rangle \rangle \mid \langle a, p \rangle \in c_1 \wedge \langle a', p' \rangle \in c_2 \wedge a, a' \notin K \} \\&= \{ \langle a' \theta a, \langle p, p' \rangle \rangle \mid \langle a, p \rangle \in c_1 \wedge \langle a', p' \rangle \in c_2 \wedge a, a' \notin K \wedge a' \theta a \neq 0 \}\end{aligned}$$

CS: Process Algebra

Step $\star$ requires restriction to pairs of processes such that:

$$\phi \langle p, q \rangle =$$

$$\forall a \in (\mathcal{P}\pi_1 \cdot \omega) \ p, a' \in (\mathcal{P}\pi_1 \cdot \omega) \ q \cdot a \theta a' \neq 0 \Rightarrow (a \theta a' \notin K \equiv a, a' \notin K)$$

which is lifted to the **invariant**

$$\text{uniform_restriction} = \Box_{\alpha} \phi$$

where $\alpha = \text{sel} \cdot \delta_r \cdot (\alpha_{\setminus K} \times \alpha_{\setminus K})$

CS: Process Algebra

$$\begin{aligned}
 & \forall_{a \in (\mathcal{P}\pi_1 \cdot \omega) \ p, a' \in (\mathcal{P}\pi_1 \cdot \omega) \ q} . a\theta a' \neq 0 \Rightarrow (a\theta a' \notin K \equiv a, a' \notin K) \\
 \equiv & \quad \{ \text{Ccs interaction structure} \} \\
 & \forall_{a \in \mathcal{P}(\pi_1 \cdot \omega) \ p, a' \in (\mathcal{P}\pi_1 \cdot \omega) \ q} . (a\theta a' = \tau \vee a\theta a' = 1) \Rightarrow a, a' \notin K \\
 \equiv & \quad \{ a\theta a' = 1 \iff a = a' = 1 \} \\
 & \forall_{a \in (\mathcal{P}\pi_1 \cdot \omega) \ p, a' \in (\mathcal{P}\pi_1 \cdot \omega) \ q} . a\theta a' = \tau \Rightarrow a, a' \notin K \\
 \equiv & \quad \{ a\theta a' = \tau \iff a' = \bar{a} \} \\
 & \forall_{a \in (\mathcal{P}\pi_1 \cdot \omega) \ p, a' \in (\mathcal{P}\pi_1 \cdot \omega) \ q} . a' = \bar{a} \Rightarrow a, a' \notin K \\
 \equiv & \quad \{ \text{rearranging} \} \\
 & (\mathcal{P}\pi_1 \cdot \omega) \ p \cap \overline{(\mathcal{P}\pi_1 \cdot \omega) \ q} \cap (K \cup \bar{K}) = \emptyset
 \end{aligned}$$

CS: Process Algebra

The CSP case

In this case $a\theta a' \neq 0 \Rightarrow (a\theta a' \notin K \equiv a, a' \notin K)$ is trivially true and the law holds without any side condition

Remarks



Constraints are **discovered** rather than **postulated**



They depend essentially on the chosen **interaction structure**

CS: Process Algebra

Proof Re-Use: $\backslash_K \cdot | = | \cdot (\backslash_K \times \backslash_K)$

A simple calculation leads to

$$\backslash_K \cdot | = | \cdot (\backslash_K \times \backslash_K) \quad \Leftarrow \text{uniform_restriction}'$$

Note that in $\text{uniform_restriction}'$ ϕ is closed wrt the transitions on α' , which includes both interleavings and synchronisations leading to

$$(p \mid q) \backslash_K = p \backslash_K \mid q \backslash_K \quad \Leftarrow \mathcal{L}(p) \cap \overline{\mathcal{L}(p)} \cap (K \cup \overline{K}) = \emptyset$$

Conclusions

1. Introduction
2. Sequences & Streams
3. Coinductive Modelling
4. Proofs by Calculation
5. Case study: Revisiting Process Algebras
6. Conclusions

Conclusions

In this case-study

- ✦ Coinductive re-construction of (classical) process calculi
- ✦ ... with **genericity** as the **driving motto** at three different levels
 - behaviour model
 - interaction discipline
- ✦ Leads to a simple strategy to **functional prototyping** of process calculi by direct implementation of their semantics in CHARITY [Bar02] and HASKELL
- ✦ Extensions to **weak semantics** in forthcoming P. Ribeiro thesis
- ✦ Extensions to **mobility** still open

Conclusions

In general



Systematic comparison with **bisimulation** proofs



Extension to **generalised coinduction schemes** parametric on a **distributive law** vs [Bartels, 04] **bisimulation-up-to- λ**



Application to **software architecture calculi** for both **forward** and **backward** reasoning [Marco and Rodrigues forthcoming theses]