Systems, behaviours and coinduction (I)

L.S. Barbosa

Dept. Informática,
Universidade do Minho
Braga, Portugal

DI-CCTC, UM, 2009

- Motivation
- Streams and deterministic automata

Functional architectures

$\boxed{\text{The architecture of functional designs}}$

Interfaces:	$f :: \cdots \longrightarrow \cdots$
Components:	$f = \cdots$
Connectors:	$\cdot$, $\langle\ ,\ \rangle$, $\times$, $+$, ...
Configurations:	functions assembled by composition
Properties:	invariants (pre-, post-conditions)
Behavioural effects:	monads and Kleisli compostion
Underlying maths:	universal algebra and relational calculus

Functional architectures

In particular, we've studied several ways of glueing functions
... each one leading to a different way of aggregating information:

Pipelining: leading to function space B^A (dependency)

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

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

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

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

Functional architectures

Disjunction: leading to coproduct (or disjoint union) $A + B$ (choice)

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

$$\text{where} \qquad [f,g]\,(x) \;= (x = (1,a)) \to f\ a$$
$$(x = (2,b)) \to g\ b$$

Constants & points:

empty	$() : A \longleftarrow \emptyset$
collapse	$! : \mathbf{1} \longleftarrow A$
points	$\underline{a} : A \longleftarrow \mathbf{1}$

Functional architectures

The underlying 'semantic universe' assumes 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

upon which combinators are defined by universal arrows

- associated to the product, sum and exponential constructions
- which behave ... as they should (formally, form a ccc)

but what is a category?

what does universal mean?

A parenthesis to come later ...

$$\Big(\quad \ldots \quad \Big)$$

Functional architectures

The algebra of functions provides

- provides a tool to think with when approaching a design problem
- and the possibility of animating and iterating models

It also paves the way to the ability of calculating within the models and transform them into effective programs. But this often requires both

- a notational shift (eg, getting rid of variables!)
- a wider mathematical framework (namely, relations and the relational calculus)

Functional architectures

$\boxed{\text{Example: modelling vs calculation}}$

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

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

$\boxed{\text{Now show:}}$

that any function which builds a pair is a pairing function, ie,

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

Functional architectures

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

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

Functional architectures

$\boxed{\text{Alternative universal definition}}$

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

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

that is

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

Note that

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

Functional architectures

Proof.

$$
\begin{array}{ll}
 & h = \langle \pi_1 \cdot h, \pi_2 \cdot h \rangle \\
\equiv & \qquad \{ \text{ universal property with } f = \pi_1 \cdot h,\ g = \pi_2 \cdot h \ \} \\
 & \pi_1 \cdot h = \pi_1 \cdot h \ \wedge \ \pi_2 \cdot h = \pi_2 \cdot h
\end{array}
$$

- simpler and smaller proof
- both proof and definition are generic and hold in other modelling universes (eg, relations, partial maps or ordered structures, ...)

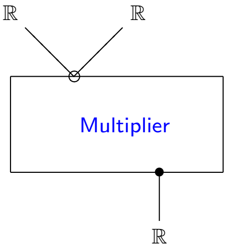
Question

Can such a calculational discipline, well established in functional programming, be extended to reason about the architecture of dynamic, reactive, state-based systems?

- persistence, i.e., internal state and state transitions
- continued interaction along the whole computational process
- potential infinite behaviour
- observability through well-defined interfaces to ensure flow of data

Behaviour

$\boxed{\text{Example: the Multiplier component}}$



Its tranformational behaviour is captured by relation:

$$M : \mathbb{R} \longleftarrow \mathbb{R} \times \mathbb{R} \quad . \quad (a \times b)\ M\ (a, b)$$

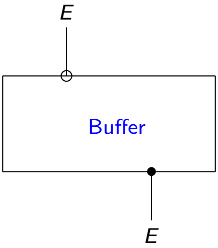
Behaviour

But its successful composition as a part in any larger system requires the knowledge of other properties, eg

- does the Multiplier consume a and b in a specific order?
- does it consume whichever of a and b that arrives first?
- does it consume a and b only when both are available?
- does it consume a and b atomically?
- does it compute and produce the result atomically together with its last input?

Behaviour

Example: the Buffer component



Behavioural constraints:

- the sequence ofdata items that goes in is exactly the same that comes out: nothing is lost, the buffer generates no data of its own, and the order of the data items is preserved.
- every data item can come out only after it goes in.

Antecipating

$\boxed{B^* \text{ – finite sequences}}$

$$[\mathrm{nil}, \mathrm{cons}] : L \longleftarrow \mathbf{1} + B \times L$$

In general:

a tool box:

an assembly process: artifact $\xleftarrow{a}$ artifact

- abstract data structures as (initial) algebras
- emphasis is on construction

Antecipating

$\boxed{B^\omega - \text{streams}}$

$$\langle \mathrm{at}, \mathrm{m} \rangle : B \times U \longleftarrow U$$

In general:

a lens:	$\bigcirc\!\frown\!\bigcirc$
an observation structure:	$\bigcirc\!\frown\!\bigcirc$ universe $\xleftarrow{c}$ universe

- abstract behavioural structures as (final) coalgebras
- emphasis is on observation

Antecipating

- The lens describes the shape (or sginature) of legal observations, whose collection corresponds to the system's generated behaviour.
- The observation structure describes the system's one-step dynamics; It's a sort of behaviour generating machine.

- Motivation
- Streams and deterministic automata

Automata

state space	U
transition function	$\mathsf{m} : U \longleftarrow U$
attribute (or label)	$\mathsf{at} : B \longleftarrow U$

i.e.,

$$p \;=\; \langle \mathsf{at}, \mathsf{m} \rangle : B \times U \longleftarrow U$$

Notation:

$$u \longrightarrow_p u' \quad \equiv \quad \mathsf{m}\, u = u'$$

$$u \downarrow_p b \quad \equiv \quad \mathsf{at}\, u = b$$

Automata

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

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

which means that

$\boxed{\text{Automata behaviours are elements of } B^{\omega} \text{ (\textit{i.e.}, streams)}}$

Streams as functions

$$B^{\omega} \;=\; \{\sigma |\; \sigma : B \longleftarrow \omega\}$$

$\mathsf{hd}\, s \;=\; s\,0$	initial value
$\mathsf{tl}\, s\, n \;=\; s\,(n+1)$	first derivative
$s^0 = s$ and $s^{k+1} = \mathsf{tl}\,(s^k)$	high-order derivatives

Streams as functions

Exercise. Prove that $s\, n \;=\; s^n\, 0$.

$$
\begin{array}{ll}
 & s\, n \\
= & \qquad \{\ \mathsf{tl}\, s \text{ definition } \} \\
 & \mathsf{tl}\, s\, (n-1) \\
= & \qquad \{\ \text{induction } \} \\
 & \mathsf{tl}\, s^{(n-1)}\, 0 \\
= & \qquad \{\ \mathsf{tl}\, s \text{ definition } \} \\
 & s^n\, 0
\end{array}
$$

Automata

$\boxed{\text{Example: A twist automata}}$

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

i.e.,

$$\mathsf{twist} = \langle \pi_1, \mathsf{s} \rangle$$

Exercise. Represent graphically this automata and describe its behaviour.

Automata

$\boxed{\text{Example: A stream automata}}$

state space	$U = B^\omega$
transition function	$\mathsf{m}\, s \;=\; \mathsf{tl}\, s$
attribute	$\mathsf{at}\, s \;=\; \mathsf{hd}\, s$

i.e.,

$$\omega \;=\; \langle \mathsf{hd}, \mathsf{tl} \rangle$$

$\boxed{\text{Automata behaviours form themselves an automata}}$

Automata morphisms

A morphism

$$h : q \longleftarrow p$$

where

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

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

$$\begin{tikzcd} U \arrow[r, "p"] \arrow[d, "h"'] & B \times U \arrow[d, "\mathrm{id} \times h"] \\ V \arrow[r, "q"] & B \times V \end{tikzcd}$$

i.e.,

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

Exercise. Derive the equational characterisation of h above.

A stream automata

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

because

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

and

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

Question

How to reason about automata behaviours?

Sequences & Streams

$\boxed{\text{Reasoning about } B^*}$

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

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

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

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

Sequences & Streams

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

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

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

Consider

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

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

Strategy

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

Coinduction & Bisimulation

Reasoning about B^ω: global view

Stream equality

$$\langle \forall\, n \;:\; n \geq 0 :\; s\, n \;=\; t\, n \rangle$$

can be established by induction over n
However, it

- requires a (workable) formula for arguments $s\, n$, $t\, n$, often not available
- does not scale easily to other behaviour types

Coinduction & Bisimulation

$\boxed{\text{Reasoning about } B^\omega\text{: local view}}$

Two streams s and r are observationally the same if

- they have identical head observations: $\mathsf{hd}\ s = \mathsf{hd}\ r$,
- and their tails — $\mathsf{tl}\ s$ and $\mathsf{tl}\ r$ — support a similar verification.

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

$$\langle x, y\rangle \in R \;\Rightarrow\; \mathsf{hd}\ x = \mathsf{hd}\ y \;\wedge\; \langle \mathsf{tl}\ x, \mathsf{tl}\ y\rangle \in R$$

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

Coinduction & Bisimulation

Th (coinduction): Bisimilarity ($\sim$) coincides with stream equality

Stream equality is, obviously, a bisimulation. Then,

$$
\begin{array}{ll}
 & s \sim r \\
\equiv & \quad \{ \ \sim \text{definition} \ \} \\
 & \langle \exists\, R : B^\omega \longleftarrow B^\omega \ : \ R \ \text{bisimulation} : \ \langle s, r \rangle \in R \rangle \\
\Rightarrow & \quad \{ \ \text{induction on } n \ \} \\
 & \langle \exists\, R : B^\omega \longleftarrow B^\omega \ : \ R \ \text{bisimulation} : \ \langle \forall\, n \ : \ n \geq 0 : \ \langle s^n, r^n \rangle \in R \rangle \rangle \\
\Rightarrow & \quad \{ \ R \ \text{bisimulation} \ \} \\
 & \langle \forall\, n \ : \ n \geq 0 : \ s^n\, 0 = r^n\, 0 \rangle \\
\equiv & \quad \{ \ s\, n = s^n\, 0 \ \} \\
 & \langle \forall\, n \ : \ n \geq 0 : \ s\, n = r\, n \rangle \\
\equiv & \quad \{ \ \text{stream equality} \ \} \\
 & s = t
\end{array}
$$

Coinduction & Bisimulation

Coinduction as a proof principle:

- a systematic way of strengthening the statement to prove: from equality $s = r$ to a larger set R which contains pair $\langle s, r \rangle$
- ensuring that such a set is a bisimulation, *i.e.*, the closure of the original set under taking derivatives

Note that,

- for proving stream equality, coinduction is both sound and complete
- moreover, it generalises from streams to a large class of behaviour types

Coinduction & Bisimulation

Exercise. Check that R below is a bisimulation

$$R \;=\; \{\langle \mathrm{map}\, f\, (\mathrm{gen}\, f\, x)\,,\, \mathrm{gen}\, f\, (f\, x)\rangle \mid x \in \ldots, f \in \ldots\}$$

- $\mathrm{hd}\ (\mathrm{map}\, f\, (\mathrm{gen}\, f\, x)) \;=\; f\, f\, x \;=\; \mathrm{hd}\ (\mathrm{gen}\, f\, (f\, x))$
- $\mathrm{tl}\ (\mathrm{map}\, f\, (\mathrm{gen}\, f\, x)) \;=\; \mathrm{map}\, f\ \ \mathrm{tl}\ \ (\mathrm{gen}\, f\, x)$ and $\mathrm{tl}\ (\mathrm{gen}\, f\, (f\, x)) \;=\; \mathrm{gen}\, f\, (f\, f\, x)$. Thus,

$$\langle \mathrm{tl}\ (\mathrm{map}\, f\, (\mathrm{gen}\, f\, x)), \mathrm{tl}\ (\mathrm{gen}\, f\, (f\, x))\rangle \in R$$

Remark:

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

Remark:

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

Coinduction & Bisimulation

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

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

Coinduction & Bisimulation

Th: Behaviour $[\![(p)]\!]$ is the unique morphism from p to ω

because

$$
\begin{array}{ll}
 & f \text{ and } g \text{ are automata morphisms} \\
\equiv & \qquad \{ \text{ morphism definition } \} \\
 & \mathsf{hd} \cdot f \;=\; \mathsf{at} \;=\; \mathsf{hd} \cdot g \;\text{ and }\; \mathsf{tl} \cdot f \;=\; f \cdot \mathsf{m} \;,\; \mathsf{tl} \cdot g \;=\; g \cdot \mathsf{m} \\
\equiv & \qquad \{ \text{ definition of bisimulation } \} \\
 & \text{relation } R = \{\langle f\,u, g\,u\rangle \mid u \in U\} \text{ is a bisimulation} \\
\equiv & \qquad \{ \text{ coinduction } \} \\
 & \langle \forall\; u \;:\; u \in U :\; f\,u \;=\; g\,u \rangle \\
\equiv & \qquad \{ \text{ function equality } \} \\
 & f \;=\; g
\end{array}
$$

An universal property

Existence and uniqueness of $[\!(p)\!]$ can be captured by the following universal property:

$$k = [\!(p)\!] \iff \omega \cdot k = (\mathrm{id} \times k) \cdot p$$

- Existence $\equiv$ definition principle (co-recursion)
- Uniqueness $\equiv$ proof principle (co-induction)

From which:

$$\begin{array}{rrcl} \text{cancellation} & \omega \cdot [\!(p)\!] & = & (\mathrm{id} \times [\!(p)\!]) \cdot p \\ \text{reflection} & [\!(\omega)\!] & = & \mathrm{id}_\omega \\ \text{fusion} & [\!(p)\!] \cdot h & = & [\!(q)\!] \quad \text{if} \quad p \cdot h = (\mathrm{id} \times h) \cdot q \end{array}$$

An universal property

$\boxed{\text{Example: fusion law}}$

$$
\begin{array}{ll}
 & [\![(p)]\!] \cdot h \;=\; [\![(q)]\!] \\
\equiv & \qquad \{ \text{ universal law } \} \\
 & \omega \cdot [\![(p)]\!] \cdot h \;=\; (\mathrm{id} \times ([\![(p)]\!] \cdot h)) \cdot q \\
\equiv & \qquad \{ \text{ cancellation law and functoriality } \} \\
 & (\mathrm{id} \times [\![(p)]\!]) \cdot p \cdot h \;=\; (\mathrm{id} \times [\![(p)]\!]) \cdot (\mathrm{id} \times h) \cdot q \\
\Leftarrow & \qquad \{ \text{ function equality } \} \\
 & p \cdot h \;=\; (\mathrm{id} \times h) \cdot q
\end{array}
$$

An universal property

... from which the following (main) result is a direct corollary:

Th: morphisms preserve behaviour: $[\![(p)]\!] = [\![(q)]\!] \cdot h$

Definition by coinduction

$\boxed{\text{Example: Stream gen, merge and twist}}$

$$\begin{tikzcd} B^\omega \arrow[r, "\langle \mathsf{hd},\mathsf{tl} \rangle"] & B \times B^\omega \\ B \arrow[u, "\mathsf{gen}"] \arrow[r, "\triangle"] & B \times B \arrow[u, "\mathsf{id}\times\mathsf{gen}"'] \end{tikzcd}$$

$$\mathsf{gen} = [\!(\triangle)\!]$$

$\boxed{\triangle \text{ carries the `genetic inheritance' of the generating process}}$

$\boxed{\text{From a programming viewpoint it is the eureka! step}}$

Definition by coinduction

$\boxed{\text{Coinductive Definition} = \text{behaviour given under all the observers}}$

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

Definition by coinduction

$\boxed{\text{Stream merge}}$

$$
\begin{tikzcd}
B^\omega \arrow[r, "\langle \mathsf{hd},\mathsf{tl} \rangle"] & B \times B^\omega \\
B^\omega \times B^\omega \arrow[u, "\mathsf{merge}"] \arrow[r, "g"] & B \times (B^\omega \times B^\omega) \arrow[u, "\mathsf{id}\times\mathsf{merge}"']
\end{tikzcd}
$$

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

Definition by coinduction

Unfolding the diagram and going pointwise, we get an explicit definition of stream merge:

$$\begin{aligned} \mathrm{hd}\ \mathrm{merge}\,(s,t) &= \mathrm{hd}\ s \\ \mathrm{tl}\ \mathrm{merge}\,(s,t) &= \mathrm{merge}\,(t, \mathrm{tl}\ s) \end{aligned}$$

Exercise. Define operators *odd* and *even* to build the stream of elements in odd (resp., even) positions. Derive the corresponding explicit definitions.

Exercise. Prove, by constructing a suitable bisimulation that $\mathit{even} \cdot \mathrm{merge} = \pi_1$.

Definition by coinduction

$\boxed{\text{Stream twist}}$

$$\begin{tikzcd} B^\omega \arrow[r, "\langle \mathsf{hd},\mathsf{tl} \rangle"] & B \times B^\omega \\ B \times B \arrow[u, "\mathsf{twist} = [\![g]\!]"] \arrow[r, "g"] & B \times (B \times B) \arrow[u, "\mathsf{id} \times \mathsf{twist}"'] \end{tikzcd}$$

$$g \;=\; \langle \pi_1, \mathsf{s} \rangle$$

Exercise. Derive the explicit definition of this operator.

Proof by coinduction

$\boxed{\text{Lemma: } \mathrm{merge} \cdot \langle \mathit{even}, \mathit{odd} \rangle = \mathrm{id}}$

- Start with $R = \{\langle \mathrm{merge}(\mathit{even}\, s, \mathit{odd}\, s), s\rangle \mid s \in B^\omega\}$
- Check the two conditions on bisimulations
  - Clearly

$$\mathrm{hd}\, \mathrm{merge}(\mathit{even}\, s, \mathit{odd}\, s) = \mathrm{hd}\, \mathit{even}\, s = \mathrm{hd}\, s$$

  - The following pair is not in R:

$$\begin{aligned} \langle \mathrm{tl}\, \mathrm{merge}(\mathit{even}\, s, \mathit{odd}\, s), \mathrm{tl}\, s\rangle &= \langle \mathrm{merge}(\mathit{odd}\, s, \mathrm{tl}(\mathit{even}\, s)), \mathrm{tl}\, s\rangle \\ &= \langle \mathrm{merge}(\mathit{odd}\, s, (\mathit{even}\, \mathrm{tl}\, \mathrm{tl}\, s)), \mathrm{tl}\, s\rangle \end{aligned}$$

- Extend R to $R \cup \{\langle \mathrm{merge}(\mathit{odd}\, s, (\mathit{even}\, \mathrm{tl}\, \mathrm{tl}\, s)), \mathrm{tl}\, s\rangle \mid s \in B^\omega\}$ and iterate the construction

Proof by coinduction

- Check the two conditions on bisimulations
  - Clearly

$$\mathsf{hd}\,\mathsf{merge}(\mathit{odd}\,s, \mathit{even}\,\mathsf{tl}\,\mathsf{tl}\,s) \;=\; \mathsf{hd}\,\mathit{odd}\,s \;=\; \mathsf{hd}\,\mathsf{tl}\;\,s$$

  - The following pair is in R:

$$\begin{aligned}
&\langle \mathsf{tl}\,\mathsf{merge}(\mathit{odd}\,s, (\mathit{even}\;\,\mathsf{tl}\,\mathsf{tl} s)), \mathsf{tl}\,\mathsf{tl}\;\,s\rangle \\
&= \langle \mathsf{merge}(\mathit{even}\,\mathsf{tl}\,\mathsf{tl}\,s, \mathsf{tl}\,\mathit{odd}\,s), \mathsf{tl}\,\mathsf{tl}\;\,s\rangle \\
&= \langle \mathsf{merge}(\mathit{even}\,\mathsf{tl}\,\mathsf{tl}\,s, \mathit{odd}\,\mathsf{tl}\,\mathsf{tl}\,s), \mathsf{tl}\,\mathsf{tl}\;\,s\rangle
\end{aligned}$$

Exercise. Repeat this proof avoiding the explicit construction of a bisimulation.

Proof by coinduction

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

i.e.

$$\mathrm{merge} \cdot (\mathrm{gen} \times \mathrm{gen}) \;=\; \mathrm{twist}$$

Proof by coinduction

$$
\begin{array}{ll}
& \mathsf{merge} \cdot (\mathsf{gen} \times \mathsf{gen}) \;=\; \mathsf{twist} \\
= & \qquad \{ \;\text{merge definition}\; \} \\
& [\![\langle \mathsf{hd} \cdot \pi_1, \mathsf{s} \cdot (\mathsf{tl} \times \mathsf{id}) \rangle]\!] \cdot (\mathsf{gen} \times \mathsf{gen}) \;=\; [\![\langle \pi_1, \mathsf{s} \rangle]\!] \\
\Leftarrow & \qquad \{ \;\text{coinduction fusion}\; \} \\
& \langle \mathsf{hd} \cdot \pi_1, \mathsf{s} \cdot (\mathsf{tl} \times \mathsf{id}) \rangle \cdot (\mathsf{gen} \times \mathsf{gen}) \;=\; \mathsf{id} \times (\mathsf{gen} \times \mathsf{gen}) \cdot \langle \pi_1, \mathsf{s} \rangle \\
= & \qquad \{ \;\times\ \text{absorption and reflection}\; \} \\
& \langle \mathsf{hd} \cdot \mathsf{gen} \cdot \pi_1, \mathsf{s} \cdot ((\mathsf{tl} \cdot \mathsf{gen}) \times \mathsf{gen}) \rangle \;=\; \mathsf{id} \times (\mathsf{gen} \times \mathsf{gen}) \cdot \langle \pi_1, \mathsf{s} \rangle \\
= & \qquad \{ \;\mathsf{tl} \cdot \mathsf{gen} = \mathsf{gen}\ \text{and}\ \mathsf{hd} \cdot \mathsf{gen} = \mathsf{id}\; \} \\
& \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle \;=\; \mathsf{id} \times (\mathsf{gen} \times \mathsf{gen}) \cdot \langle \pi_1, \mathsf{s} \rangle
\end{array}
$$

Proof by coinduction

$$
\begin{array}{ll}
& \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle \;=\; \mathsf{id} \times (\mathsf{gen} \times \mathsf{gen}) \cdot \langle \pi_1, \mathsf{s} \rangle \\
= & \qquad \{ \;\; \times \text{ absorption } \;\; \} \\
& \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle \;=\; \langle \pi_1, (\mathsf{gen} \times \mathsf{gen}) \cdot \mathsf{s} \rangle \\
= & \qquad \{ \;\; \mathsf{s} \text{ is natural, } \textit{i.e.}, \; (f \times g) \cdot \mathsf{s} = \mathsf{s} \cdot (g \times f) \;\; \} \\
& \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle \;=\; \langle \pi_1, \mathsf{s} \cdot (\mathsf{gen} \times \mathsf{gen}) \rangle
\end{array}
$$

Exercise. Repeat this proof by explicitly building a suitable bisimulation.