Probabilistic Programming with Coinductive Data

Alex Simpson

jww. Danel Ahman and Léo Soudant

Faculty of Mathematics and Physics
University of Ljubljana, Slovenia

MFPS, 22nd June 2023

Overview

- Topic: programming with probability distributions on infinite sequences and other coinductive data.
- Examples include: random walks, the Chinese restaurant process and Brownian motion.
- A functional language based on fine-grained call-by-value with primitives for coinductive types and probabilistic sampling.
- The language has a natural denotational semantics, over standard Borel spaces, which exploits limit-preservation properties of the Giry monad to model corecursion for coinductive data.
- A subtle operational semantics is required to correctly implement corecursion, in which effect 'sequencing' is performed out of sequence using a call-by-need strategy,

Some overlap in content with:

> [DKPS 2023] Dash, Kaddar, Paquet & Staton, *Affine Monads and Lazy Structures for Bayesian Programming*, POPL 2023

- Overlap: programming with probability distributions on infinite data, monad-based language and semantics, laziness is operationally necessary and semantically justified because the distribution monad is affine and commutative.
- [DKPS 2023]: a second non-affine monad supporting Bayesian Monte Carlo methods, new lazy inference methods, Haskell implementation.
- This talk: coinductive data types and corecursion justified by limit-preservation properties of the distribution monad, operational semantics, agreement of denotational and operational semantics.

Example 1: iid streams (preliminary)

$$\mathit{iid} \;:\; \text{dist real} \;\to\; \text{stream real}$$

$$\mathit{iid}\ d = \;\text{do}\ x \leftarrow \text{sample}(d)$$
$$\text{in return}\ (\, x : (\mathit{iid}\ d)\,)$$

$$\mathit{bits} \;:\; \text{unit} \;\to\; \text{stream real}$$

$$\mathit{bits}\ () \;=\; \mathit{iid}\ (\text{uniform}\{0, 1\})$$

$$\mathit{gaussians} \;:\; \text{unit} \;\to\; \text{stream real}$$

$$\mathit{gaussians}\ () \;=\; \mathit{iid}\ (\text{normal}(0, 1))$$

$iid\ d = \text{do } x \leftarrow \text{sample}(d) \text{ in return } (x : (iid\ d))$

$bits() = iid\ (\text{uniform}\{0, 1\})$

$xs = bits() : \text{stream real}$

$y = \text{nth } xs\ 1$ — the second element (element with index 1) in xs

$z = \text{nth } xs\ 1$

$y == z \implies ???$

Example 1: iid streams (corrected)

$$
\begin{aligned}
&\mathit{iid} \;:\; \text{dist real} \;\to\; \text{stream real} \\
&\mathit{iid}\ d = \;\; \text{do } x \leftarrow \text{sample}(d) \\
&\qquad\qquad \text{do } \mathit{xs} \leftarrow \mathit{iid}\ d \\
&\qquad\qquad\quad \text{in return } (\, x : \mathit{xs} \,)
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{bits} \;:\; \text{unit} \;\to\; \text{stream real} \\
&\mathit{bits}\ () \;=\; \mathit{iid}\ (\text{uniform}\{0, 1\})
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{gaussians} \;:\; \text{unit} \;\to\; \text{stream real} \\
&\mathit{gaussians}\ () \;=\; \mathit{iid}\ (\text{normal}(0, 1))
\end{aligned}
$$

$iid\ d = \ \text{do}\ x \leftarrow \text{sample}(d)\ \text{in do}\ xs \leftarrow iid\ d\ \text{in return}\ (x : xs)$

$bits() \ = \ iid\ (\text{uniform}\{0, 1\})$

$xs \ = \ bits() \ : \ \text{stream real}$

$y \ = \ \text{nth}\ xs\ 1$ — the second element (element with index 1) in xs

$z \ = \ \text{nth}\ xs\ 1$

$y == z \implies ???$

$iid\ d = \ \text{do}\ x \leftarrow \text{sample}(d)\ \text{in do}\ xs \leftarrow iid\ d\ \text{in return}\ (x : xs)$

$bits() \ = \ iid\ (\text{uniform}\{0, 1\})$

$xs \ = \ bits() \ : \ \text{stream real}$

$y \ = \ \text{nth}\ xs\ 1$ — the second element (element with index 1) in xs

$z \ = \ \text{nth}\ xs\ 1$

$y \texttt{ == } z \implies \text{true}$

$$
\begin{aligned}
&\mathit{iid}\ d = \ \text{do}\ x \leftarrow \text{sample}(d)\ \text{in do}\ \mathit{xs} \leftarrow \mathit{iid}\ d\ \text{in return}\ (x : \mathit{xs}) \\
&\mathit{bits}() \ = \ \mathit{iid}\ (\text{uniform}\{0,1\}) \\
\\
&\mathit{xs} \ = \ \mathit{bits}() \ : \ \text{stream real} \\
&y \ = \ \text{nth}\ \mathit{xs}\ 1 \ \text{— the second element (element with index 1) in}\ \mathit{xs} \\
&z \ = \ \text{nth}\ \mathit{xs}\ 1 \\
&y == z \quad \Longrightarrow
\end{aligned}
$$

xs : stream real behaves as an infinite sequence of bits, randomly generated with the uniform probability distribution on $\{0,1\}^\omega$.

In general, we interpret programs of result type stream real as probabilistically generating infinite sequences of real numbers.

$$\mathit{iid}\ D = \ \text{do}\ x \leftarrow \text{sample}(D)\ \text{in do}\ xs \leftarrow \mathit{iid}\ D\ \text{in return}\ (\, x : xs\,)$$

- The language separates general computations (e.g., sample(D), *iid* D and return $(x : xs)$) from effect-free expressions (e.g., x, xs and $x : xs$). (Fine-grained call-by-value)
- Streams in particular are constructed by $x : xs$ where the head x : real and tail xs : stream real are both expressions.
- If the execution of 'sequencing' do $x \leftarrow M$ in N is performed in-sequence then stream-building programs loop. We avoid this by using a call-by-need strategy to implement out-of-sequence 'sequencing'.

Denotational semantics

Types are measurable spaces:

$$\begin{aligned} [\![\mathsf{real}]\!] &= \mathbb{R} && \text{(Borel } \sigma\text{-algebra)} \\ [\![\mathsf{stream}\ \sigma]\!] &= [\![\sigma]\!]^{\omega} \\ [\![\mathsf{dist}\ \sigma]\!] &= \mathcal{G}[\![\sigma]\!] && (\mathcal{G} \text{ is the Giry monad}) \end{aligned}$$

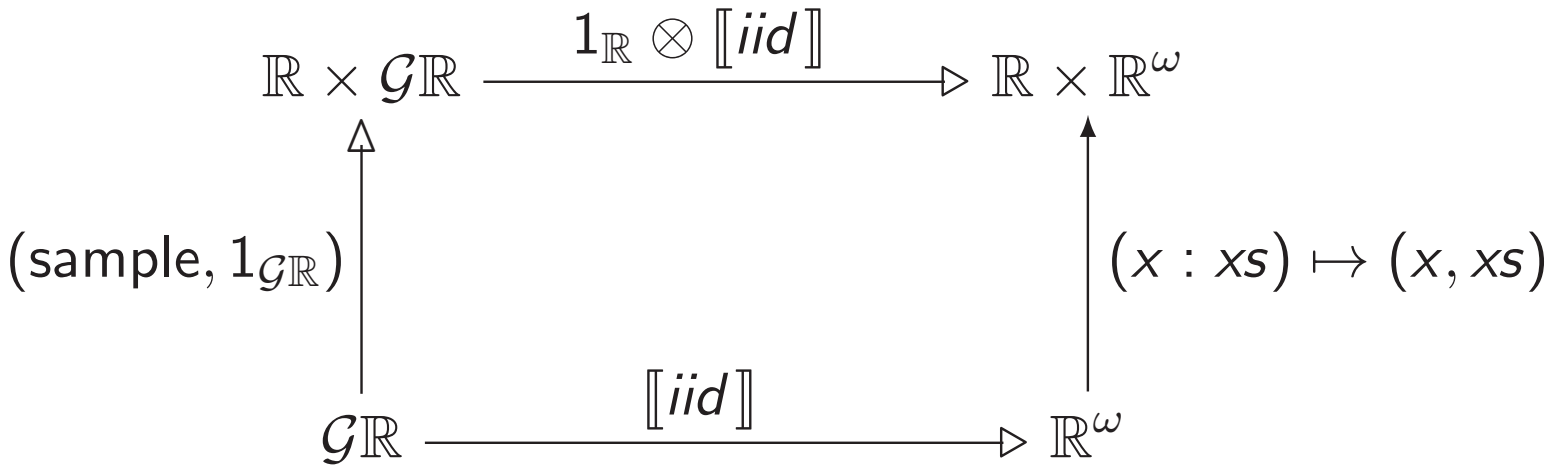
A program $M : \sigma \to \tau$ is interpreted as a Kleisli map for the Giry monad:

$$[\![\sigma]\!] \xrightarrow{[\![M]\!]}\!\!\triangleright\ [\![\tau]\!] \ := \ [\![\sigma]\!] \xrightarrow{[\![M]\!]} \mathcal{G}[\![\tau]\!] \ .$$

equivalently as a Markov kernel.

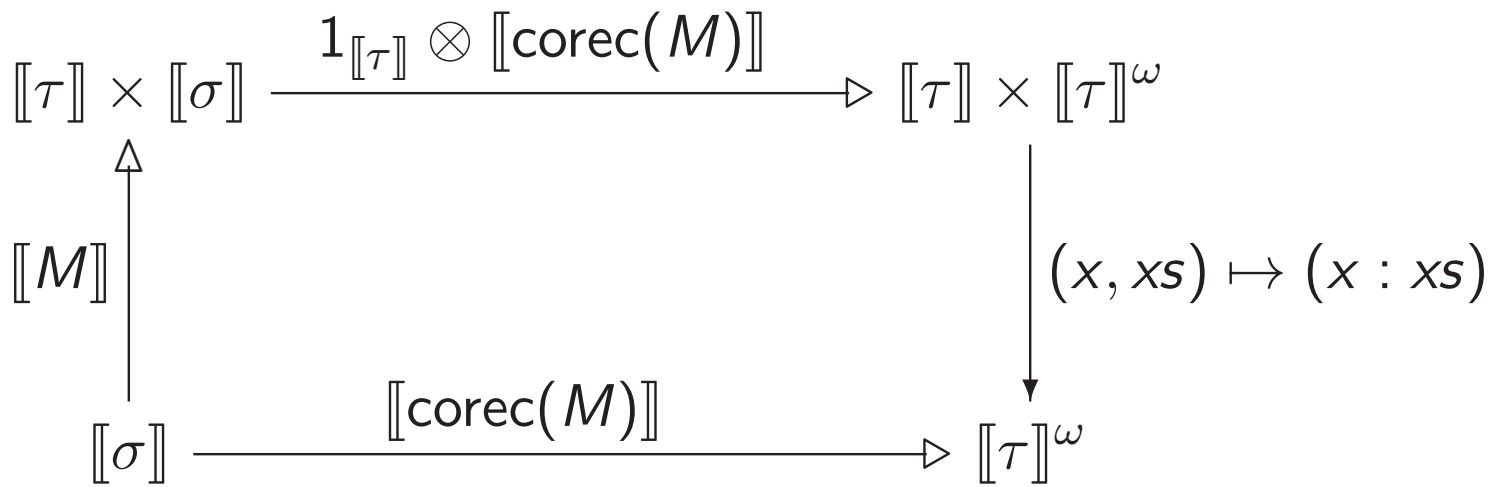
$$
\begin{aligned}
&\mathit{iid} \;:\; \text{dist real} \to \text{stream real} \\
&\mathit{iid}\ d \;=\; \text{do } x \leftarrow \text{sample}(d) \text{ in do } \mathit{xs} \leftarrow \mathit{iid}\ d \text{ in return } (x : \mathit{xs})
\end{aligned}
$$

$[\![\mathit{iid}]\!]$ is the unique Kleisli map such that:



Justified because the right-hand map is a final coalgebra for $1_{\mathbb{R}} \otimes (-)$ in the Kleisli category [Kerstan & König 2013].

For any $M : \sigma \to \tau \times \sigma$, the final coalgebra property determines $\mathsf{corec}(M) : \sigma \to \mathsf{stream}\ \tau$ such that $[\![\mathsf{corec}(M)]\!]$ is the unique Kleisli map such that:



That is, $\mathsf{corec}(M)$ is the unique solution to:

$$\begin{aligned}\mathsf{corec}(M)\,x \;=\; &\mathsf{do}\ p \leftarrow M\,x\ \mathsf{in}\ \mathsf{match}\ p\ \mathsf{as}\ (y, x')\ \mathsf{in}\\ &\mathsf{do}\ ys \leftarrow \mathsf{corec}(M)\,x'\ \mathsf{in}\ \mathsf{return}\ (y : ys)\end{aligned}$$

Example 2: Random walks

Any probability distribution D on $\mathbb{R}$ generates a random walk

$$\left(\sum_{i=1}^{n} X_{i-1}\right)_{n\geq 0}$$

where $(X_n)_{n\geq 0}$ is a sequence of iid random variables, each with distribution D.

This is directly implemented by *walk* : dist real $\rightarrow$ stream real, which makes use of an auxiliary *sums* : stream real $\rightarrow$ stream real implementing the operation

$$(x_n)_{n\geq 0} \mapsto \left(\sum_{i=1}^{n} x_{i-1}\right)_{n\geq 0}$$

$$\mathit{sums\text{-}from} \;:\; \text{stream real} \;\to\; \text{real} \;\to\; \text{stream real}$$

$$\mathit{sums\text{-}from}\ \mathit{xs}\ \mathit{s} = \text{match}\ \mathit{xs}\ \text{as}\ (\mathit{y} : \mathit{ys})\ \text{in}$$

$$\text{do}\ \mathit{zs} \leftarrow \mathit{sums\text{-}from}\ \mathit{ys}\ (\mathit{s}+\mathit{y})\ \text{in return}\ (\mathit{s} : \mathit{zs})$$

$$\mathit{sums} \;:\; \text{stream real} \;\to\; \text{stream real}$$

$$\mathit{sums}\ \mathit{xs} = \mathit{sums\text{-}from}\ \mathit{x}\ 0\mathit{s}$$

$$\mathit{walk} \;:\; \text{dist real} \;\to\; \text{stream real}$$

$$\mathit{walk}\ \mathit{d} \;=\; \text{do}\ \mathit{xs} \leftarrow \mathit{iid}\ \mathit{d}\ \text{in}\ \mathit{sums}\ \mathit{xs}$$

$$\mathit{discrete\text{-}walk} \;:\; \text{unit} \;\to\; \text{stream real}$$

$$\mathit{discrete\text{-}walk}\ ()\ =\ \mathit{walk}\ (\text{uniform}\{-1, 1\})$$

$$\mathit{gaussian\text{-}walk} \;:\; \text{unit} \;\to\; \text{stream real}$$

$$\mathit{gaussian\text{-}walk}\,()\ =\ \mathit{walk}\ (\text{normal}(0, 1))$$

A more direct implementation

$$\mathit{walk\text{-}from} \;:\; \text{dist real} \;\to\; \text{real} \;\to\; \text{stream real}$$

$$\mathit{walk\text{-}from}\ d\ s$$

$$= \text{do } y \leftarrow \text{sample } d \text{ in do } xs \leftarrow \mathit{walk\text{-}from}\ d\ (s+y) \text{ in return } (s : xs)$$

$$\mathit{walk}' \;:\; \text{dist real} \;\to\; \text{stream real}$$

$$\mathit{walk}'\ d \;=\; \mathit{walk\text{-}from}\ d\ 0$$

A more direct implementation

$$\mathit{walk\text{-}from} \;:\; \text{dist real} \;\to\; \text{real} \;\to\; \text{stream real}$$

$$\mathit{walk\text{-}from}\ d\ s$$
$$= \text{do } y \leftarrow \text{sample } d \text{ in do } xs \leftarrow \mathit{walk\text{-}from}\ d\ (s+y) \text{ in return } (s : xs)$$

$$\mathit{walk}' \;:\; \text{dist real} \;\to\; \text{stream real}$$

$$\mathit{walk}'\ d \;=\; \mathit{walk\text{-}from}\ d\ 0$$

By the final coalgebra property *walk-from d* is the unique solution to the highlighted equation above.

Proof that $walk\ d = walk'\ d$

$$walk'\ d = walk\text{-}from\ d\ 0 \qquad walk\ d = \text{do } xs \leftarrow iid\ d \text{ in } sums\ xs$$
$$= \text{do } xs \leftarrow iid\ d \text{ in } sums\text{-}from\ xs\ 0$$

We prove $walk\text{-}from\ d\ s = \text{do } xs \leftarrow iid\ d \text{ in } sums\text{-}from\ xs\ s$.

$$\begin{aligned}
&\text{do } xs \leftarrow iid\ d \text{ in } sums\text{-}from\ xs\ s \\
&= \text{do } xs \leftarrow (\text{do } y \leftarrow \text{sample } d \text{ in do } ys \leftarrow iid\ d \text{ in return } (y : ys)) \text{ in} \\
&\quad \text{match } xs \text{ as } (y : ys) \text{ in do } zs \leftarrow sums\text{-}from\ ys\ (s+y) \text{ in return } (s : zs) \\
&= \text{do } y \leftarrow \text{sample } d \text{ in do } ys \leftarrow iid\ d \text{ in do } xs \leftarrow \text{return } (y : ys) \text{ in} \\
&\quad \text{match } xs \text{ as } (y : ys) \text{ in do } zs \leftarrow sums\text{-}from\ ys\ (s+y) \text{ in return } (s : zs) \\
&= \text{do } y \leftarrow \text{sample } d \text{ in do } ys \leftarrow iid\ d \text{ in} \\
&\quad \text{do } zs \leftarrow sums\text{-}from\ ys\ (s+y) \text{ in return } (s : zs) \\
&= \text{do } y \leftarrow \text{sample } d \text{ in do } zs \leftarrow (\text{do } ys \leftarrow iid\ d \text{ in } sums\text{-}from\ ys\ (s+y)) \text{ in} \\
&\quad \text{return } (s : zs)
\end{aligned}$$

So do $xs \leftarrow iid\ d$ in $sums\text{-}from\ xs$ satisfies the equation characterising $walk\text{-}from\ d$.

□

Example 3: Chinese restaurant process (uses lists)

Diners arrive in sequence at a Chinese restaurant, in which there is an unlimited supply of tables and unlimited space at each table.

When the $(n+1)$-th diner arrives there are n customers seated at m occupied tables ($0 \leq m \leq n$). With probability $\frac{1}{n+1}$ the diner decides to sit at the next empty table, and with probability $\frac{n}{n+1}$ to join an occupied table. In the latter case, the diner chooses one of the existing customers (with the uniform probability distribution) and joins the table at which that customer is sitting.

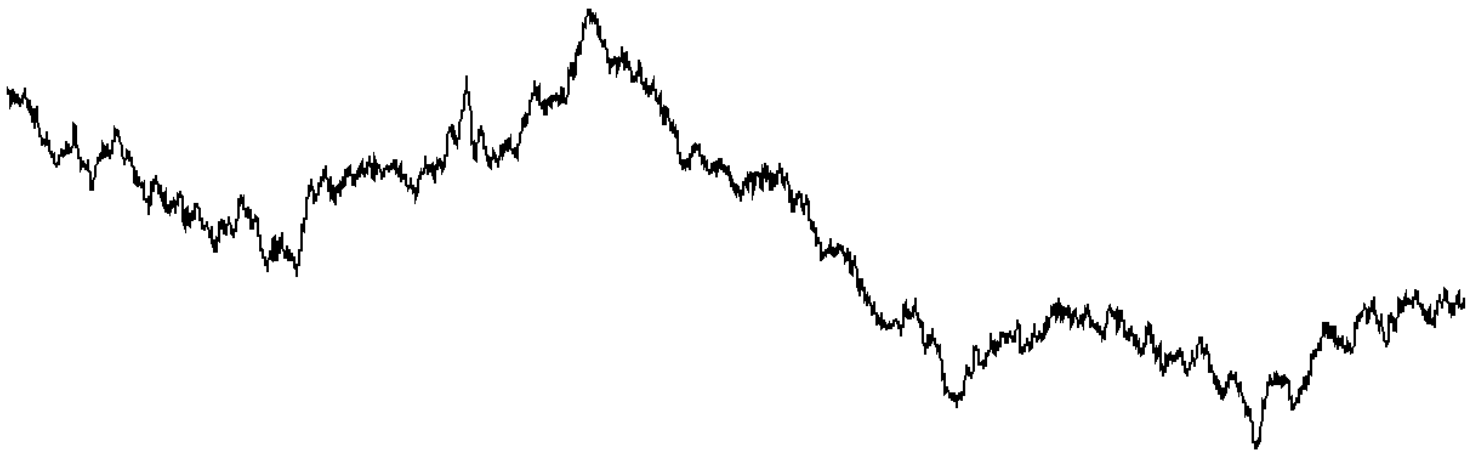
The program *chinese* : unit $\rightarrow$ stream nat returns a sequence $(t_n)_{n \geq 0}$ of natural numbers, where t_n is the table number of the table chosen by diner $n+1$, with empty tables being allocated in numerical order. So necessarily $t_0 = 1$.

$$
\begin{array}{l}
\mathit{arrive} \;:\; \mathsf{nat} \to \mathsf{nat} \to \mathsf{list}\ \mathsf{nat} \to \mathsf{stream}\ \mathsf{nat} \\
\mathit{arrive}\ n\ m\ ts \;=\; \\
\quad \mathsf{do}\ d \leftarrow \mathsf{sample}(\mathsf{uniform}\{0, \ldots, n\})\ \mathsf{in} \\
\quad \mathsf{do}\ t \leftarrow \{\mathsf{if}\ d == 0\ \mathsf{then}\ \mathsf{return}\ (m+1)\ \mathsf{else}\ (\mathsf{nth}\ ts\ (n-d))\}\ \mathsf{in} \\
\quad \mathsf{do}\ cs \leftarrow \mathit{arrive}\ (n+1)\ (d == 0\,?\, m+1 : m)\ (t : ts)\ \mathsf{in} \\
\qquad \mathsf{return}\ (\,t : cs)\,) \\
\\
\mathit{chinese} \;:\; \mathsf{unit} \;\to\; \mathsf{stream}\ \mathsf{nat} \\
\mathit{chinese}() \;=\; \mathit{arrive}\ 0\ 0\ [\,]
\end{array}
$$

arrive n m *ts* is called when n diners are already seated at m tables and *ts* is a list of length n, whose entry at index $n - d$ is the table number (from 1 to m) of diner d (from 1 to n).

Example 4: Brownian motion (uses coinductive trees)

A standard Brownian motion $(B_t)_{t\in[0,1]}$:



We define

$$\mathit{brownian} \ : \ \text{unit} \to \text{B-Tree}$$

where

$$\text{B-Tree} \ := \ \nu X.\ \text{real} \times X \times \text{real} \times X \times \text{real}$$

Every $\text{Node}(x, l, c, r, y)$ at depth n (the root is depth 0) in a tree generated by $\mathit{brownian}\,()$ specifies $B_{c-\frac{1}{2^{n+1}}} = x$ and $B_{c+\frac{1}{2^{n+1}}} = y$, for a Brownian motion $(B_t)_{t\in[0,1]}$, where c ranges over the dyadic rationals in $[0, 1]$ with denominator 2^{n+1} and odd numerator.

(l and r are the child subtrees of $\text{Node}(x, l, c, r, y)$.)

The program uses the Lévy construction of Brownian motion.

$$\text{B-Tree} \;=\; \nu X.\ \text{real} \times X \times \text{real} \times X \times \text{real}$$

$$\mathit{split} \;:\; \text{nat} \;\to\; \text{real} \;\to\; \text{real} \to\; \text{real} \;\to\; \text{B-Tree}$$

$$\mathit{split}\ n\ x\ c\ y \;=$$

$$\text{do } z \leftarrow \text{sample}\left(\text{normal}\left(\frac{x+y}{2}, \frac{1}{2^{n+2}}\right)\right) \text{ in}$$

$$\text{do } l \leftarrow \mathit{split}\ (n+1)\ x\ \left(c - \frac{1}{2^{n+2}}\right)\ z \text{ in}$$

$$\text{do } r \leftarrow \mathit{split}\ (n+1)\ z\ \left(c + \frac{1}{2^{n+2}}\right)\ y \text{ in}$$

$$\text{return Node}(x, l, c, r, y)$$

$$\mathit{brownian} \;:\; \text{unit} \;\to\; \text{B-Tree}$$

$$\mathit{brownian}\,() \;=\; \text{do } y \leftarrow \text{sample}(\text{normal}(0,1)) \text{ in } \mathit{split}\ 0\ 0\ \frac{1}{2}\ y$$

Types

Ground types:

$$\alpha \;::=\; \mathsf{unit} \mid \mathsf{bool} \mid \mathsf{nat} \mid \mathsf{real}$$

Data types:

$$\sigma \;::=\; \alpha \mid \sigma \times \sigma \mid \sigma + \sigma \mid \mathsf{dist}\,\sigma \mid X \mid \mu X.\,\sigma \mid Y \mid \nu Y.\,\sigma$$

N.B., we distinguish between μ-variables X and ν-variables Y.

Restrictions on type formation:

- $\mathsf{dist}\,\sigma$: the type σ has no free μ-variables.
- $\mu X.\,\sigma$: the type σ has no free ν-variables.
- $\nu Y.\,\sigma$: the type σ has no free μ-variables.

Closed types are interpreted as standard Borel spaces.

Standard Borel spaces

SBS is the category whose objects are Polish spaces (topological spaces arising from complete separable metrics), and whose morphisms are Borel-measurable functions.

SBS is the free category with: countable limits, countable extensive coproducts, countably boolean. [Chen 2023]

The first description explicitly exhibits **SBS** as a full subcategory of the category of measurable spaces.

The Giry monad preserves standard Borel spaces.

Every type $\sigma(X_1, \ldots, X_m, Y_1, \ldots, Y_n)$ defines a functor

$$[\![\sigma]\!] : \mathbf{SBS}^m \times \mathbf{SBS}^n \to \mathbf{SBS}$$

with a distributive law

$$\lambda_\sigma : [\![\sigma]\!]\mathcal{G}^{m+n} \Rightarrow \mathcal{G}[\![\sigma]\!]$$

of the Giry monad over $[\![\sigma]\!]$, hence (equivalently) a Kleisli lifting

$$\begin{array}{ccc} \mathbf{SBS}_{\mathcal{G}}^m \times \mathbf{SBS}_{\mathcal{G}}^n & \xrightarrow{[\![\sigma]\!]_{\mathcal{G}}} & \mathbf{SBS}_{\mathcal{G}} \\ \uparrow{\scriptstyle J^{m+n}} & & \uparrow{\scriptstyle J} \\ \mathbf{SBS}^m \times \mathbf{SBS}^n & \xrightarrow{[\![\sigma]\!]} & \mathbf{SBS} \end{array}$$

The functor:

$$[\![\sigma]\!] : \mathbf{SBS}^m \times \mathbf{SBS}^n \to \mathbf{SBS}$$

satisfies:

- for all $\vec{B} \in \mathbf{SBS}^n$, $[\![\sigma]\!](-, \vec{B})$ preserves monos and colimits of ω-chains of monos, and
- for all $\vec{A} \in \mathbf{SBS}^m$, $[\![\sigma]\!](\vec{A}, -)$ preserves limits of ω^{op}-chains.

For example,

$$[\![\mathrm{dist}\ Y]\!] = \mathcal{G} \colon \mathbf{SBS} \to \mathbf{SBS}$$

preserves limits of ω^{op}-chains [folklore] (related to Kolmogorov's extension theorem).

Example $\nu Y.\, \sigma(Y_1, \ldots, Y_n, Y)$

For any $\vec{B} \in \mathbf{SBS}^n$ the final coalgebra for

$$F_{\vec{B}} := [\![\sigma]\!](\vec{B}, -) : \mathbf{SBS} \to \mathbf{SBS}$$

is constructed as the ω^{op}-limit in $\mathbf{SBS}$

$$1 \longleftarrow F_{\vec{B}}1 \longleftarrow F_{\vec{B}}^2 1 \longleftarrow \quad \cdots \quad \cdots \quad Z \longleftarrow F_{\vec{B}}Z$$

Lemma For any category $\mathcal{C}$ with monad T, the functors $J : \mathcal{C} \to \mathcal{C}_T$ and $T : \mathcal{C} \to \mathcal{C}$ preserve exactly the same limits.

Because $\mathcal{G} : \mathbf{SBS} \to \mathbf{SBS}$ preserves terminal object and ω^{op}-limits, it follows that $J : \mathbf{SBS} \to \mathbf{SBS}_{\mathcal{G}}$ does too. So the limit diagram below gives the final coalgebra for $[\![\sigma]\!]_{\mathcal{G}}(\vec{B}, -) : \mathbf{SBS}_{\mathcal{G}} \to \mathbf{SBS}_{\mathcal{G}}$.

$$1 \longleftarrow F_{\vec{B}}1 \longleftarrow F_{\vec{B}}^2 1 \longleftarrow \quad \cdots \quad \cdots \quad Z \longleftarrow F_{\vec{B}}Z$$

Coincidence of final coalgebras

The above shows that the functor $J\colon \mathbf{SBS} \to \mathbf{SBS}_{\mathcal{G}}$ maps final coalgebras for $[\![\sigma]\!](\vec{B}, -) : \mathbf{SBS} \to \mathbf{SBS}$ to final coalgebras for $[\![\sigma]\!]_{\mathcal{G}}(\vec{B}, -) : \mathbf{SBS}_{\mathcal{G}} \to \mathbf{SBS}_{\mathcal{G}}$.

In the case $\sigma = Y_1 \times Y$. we get the result that $J((a : as) \mapsto (a, as))\colon A^\omega \longrightarrow\!\!\triangleright\, A \times A^\omega$ is a final coalgebra for $A \otimes (-)$ on $\mathbf{SBS}_{\mathcal{G}}$, cf. [Kerstan & König 2013].

$[\![\nu Y.\, \sigma(Y_1, \ldots, Y_n, Y)]\!] : \mathbf{SBS}^n \to \mathbf{SBS}$ is defined as the final-coalgebra-finding functor that maps $\vec{B} \in \mathbf{SBS}^n$ to the final coalgebra for $[\![\sigma]\!](\vec{B}, -) : \mathbf{SBS} \to \mathbf{SBS}$.

Similarly, $[\![\nu Y.\, \sigma(Y_1, \ldots, Y_n, Y)]\!]_{\mathcal{G}} : \mathbf{SBS}^n_{\mathcal{G}} \to \mathbf{SBS}_{\mathcal{G}}$ is the final-coalgebra-finding functor that maps $\vec{B} \in \mathbf{SBS}^n$ to the final coalgebra for $[\![\sigma]\!]_{\mathcal{G}}(\vec{B}, -) : \mathbf{SBS}_{\mathcal{G}} \to \mathbf{SBS}_{\mathcal{G}}$.

Program syntax (fine-grained call-by-value [LPT 2003])

We syntactically distinguish between effect-free expressions and (possibly) effectful computations

$$\frac{\Gamma \vdash E : \tau}{\Gamma \vdash \mathsf{return}\ E : \tau}$$

E is an expression, return E is a computation.

$$\frac{\Gamma \vdash M : \tau \qquad \Gamma, x : \tau \vdash N : \tau'}{\Gamma \vdash \mathsf{do}\ x \leftarrow M\ \mathsf{in}\ N : \tau'}$$

M, N and do $x \leftarrow M$ in N are computations.

Inductive types

$$\frac{\Gamma \vdash E : \sigma[X := \mu X.\,\sigma]}{\Gamma \vdash \mathsf{wrap}(E) : \mu X.\,\sigma}$$

$$\frac{\Gamma \vdash E : \mu X.\,\sigma \qquad \Gamma, x : \sigma[X := \mu X.\,\sigma] \vdash M : \tau}{\Gamma \vdash \mathsf{match}\ E\ \mathsf{as}\ \mathsf{wrap}(x)\ \mathsf{in}\ M : \tau}$$

$$\frac{\Gamma \vdash E : \mu X.\,\sigma \qquad \Gamma, x : \sigma[X := \tau] \vdash M : \tau}{\Gamma \vdash \mathsf{rec}\ (x \mapsto M)\ \mathsf{on}\ E : \tau}$$

Return values: $\mathsf{wrap}(V)$, where V is a return value of type $\sigma[X := \mu X.\,\sigma]$.

Coinductive types

$$\frac{\Gamma \vdash E : \sigma[Y := \nu Y.\,\sigma]}{\Gamma \vdash \mathsf{roll}(E) : \nu Y.\,\sigma}$$

$$\frac{\Gamma \vdash E : \nu Y.\,\sigma \qquad \Gamma, x : \sigma[Y := \nu Y.\,\sigma] \vdash M : \tau}{\Gamma \vdash \mathsf{match}\ E\ \mathsf{as}\ \mathsf{roll}(x)\ \mathsf{in}\ M : \tau}$$

$$\frac{\Gamma \vdash E : \tau \qquad \Gamma, x : \tau \vdash M : \sigma[Y := \tau]}{\Gamma \vdash \mathsf{corec}\ (x \mapsto M)\ \mathsf{on}\ E : \nu Y.\,\sigma}$$

Return values: $\mathsf{roll}(E)$, where E is an expression of type $\sigma[Y := \nu Y.\,\sigma]$.

Streams

Custom syntax for the type stream $\tau := \nu Y.\,(\tau \times Y)$:

$$\frac{\Gamma \vdash E : \sigma \quad \Gamma \vdash E' : \text{stream } \sigma}{\Gamma \vdash (E : E') : \text{stream } \sigma}$$

$$\frac{\Gamma \vdash E : \text{stream } \sigma \qquad \Gamma, x : \sigma, y : \text{stream } \sigma \vdash M : \tau}{\Gamma \vdash \text{match } E \text{ as } (x : y) \text{ in } M : \tau}$$

$$\frac{\Gamma \vdash E : \tau \qquad \Gamma, x : \tau \vdash M : \sigma \times \tau}{\Gamma \vdash \text{corec } (x \mapsto M) \text{ on } E : \text{stream } \sigma}$$

Return values: $(V : E)$, where V is a return value of type σ and E is an expression of type stream σ.

Operational semantics

Key feature: call-by-need implementation of 'sequencing':

$$\mathsf{do}\ x \leftarrow M\ \mathsf{in}\ N$$

- N is executed until (if at all) a value for x is needed.
- If and when the first value for x is needed, M is executed, and its result value is substituted for all occurrences of x.
- Programs (which terminate and have only probabilistic effects) are interpreted using the affine and commutative Giry monad.
- Affineness means that the call-by-need strategy is sound when no value for x is required in the execution of N.
- Commutativity means that the call-by-need strategy is sound even though it disregards the usual sequential order of effect invocation specified by $\mathsf{do}\ x \leftarrow M\ \mathsf{in}\ N$.

Configurations

An execution task $\langle x : \tau \leftarrow M \rangle$ says: execute the effectful computation $M : \tau$ and assign the resulting value to x.

The operational semantics defines a transition system between configurations:

$$y_m \dots y_0 \mid \langle x_n : \tau_n \leftarrow M_n \rangle \dots \langle x_0 : \tau_0 \leftarrow M_0 \rangle$$

- Ultimately we are executing the task $\langle x_0 \leftarrow M_0 \rangle$
- $y_m \dots y_0$ is the subsequence of $x_n \dots x_0$ of scheduled tasks. Currently we are executing $\langle x_i \leftarrow M_i \rangle$ for which $x_i = y_m$, and $y_{m-1} \dots y_0$ is the stack of pending tasks. Always $y_0 = x_0$.
- The remaining variables in $x_n \dots x_0$ identify execution tasks that have not yet been scheduled as they may not need to be evaluted (laziness).

Example transition rules

$$x\,\Xi \mid \mathcal{E}\,\langle x \leftarrow M[y]\rangle\,\mathcal{E}' \longrightarrow y\,x\,\Xi \mid \mathcal{E}\,\langle x \leftarrow M[y]\rangle\,\mathcal{E}'$$

where $M[-]$ is an evaluation context. A value for y is needed, so x is suspended and y becomes active.

$$x\,\Xi \mid \mathcal{E}\,\langle x \leftarrow \text{return } V\rangle\,\mathcal{E}' \longrightarrow \Xi \mid \mathcal{E}\,\mathcal{E}'[x := V] \qquad (\Xi \text{ nonempty})$$

The return value V is substituted for all occurrences of x in $\mathcal{E}'$, and x is popped off the stack of scheduled tasks.

$$x\,\Xi \mid \mathcal{E}\,\langle x \leftarrow \text{ do } y \leftarrow M \text{ in } N\rangle\,\mathcal{E}' \longrightarrow x\,\Xi \mid \mathcal{E}\,\langle y \leftarrow M\rangle\,\langle x \leftarrow N\rangle\,\mathcal{E}'$$

x is still scheduled so execution contines with N, as required under call-by-need evaluation.

Corecursion for streams

$$\frac{\Gamma \vdash E : \tau \qquad \Gamma, x : \tau \vdash M : \sigma \times \tau}{\Gamma \vdash \text{corec } (x \mapsto M) \text{ on } E : \text{stream } \sigma}$$

$$x\,\Xi \mid \mathcal{E}\,\langle x \leftarrow \text{corec } (x \mapsto M) \text{ on } V\rangle\,\mathcal{E}' \longrightarrow x\,\Xi \mid \mathcal{E}\,\langle x \leftarrow R\rangle\,\mathcal{E}'$$

where V is a return value and R is the expression below.

$$\begin{array}{l}\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}\\ \quad \text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{array}$$

Signature of basic distributions

A basic distribution is given a signature declaration of the form

$$\textit{dist-op} \;:\; (\alpha_1, \ldots, \alpha_k) \;\to_{\mathsf{dist}}\; \alpha$$

E.g.,

$$\begin{aligned} \mathsf{normal} &: (\mathsf{real}, \mathsf{real}) \to_{\mathsf{dist}} \mathsf{real} \\ \mathsf{uniform} &: (\mathsf{real}, \mathsf{real}) \to_{\mathsf{dist}} \mathsf{real} \\ \mathsf{uniform} &: (\mathsf{nat}) \to_{\mathsf{dist}} \mathsf{nat} \end{aligned}$$

for each *dist-op*: $(\alpha_1, \ldots, \alpha_n) \to_{\mathsf{dist}} \beta$

Semantically, we can incorporate any Markov kernel

Sampling a distribution

$$x\,\Xi \mid \mathcal{E}\,\langle x \leftarrow \text{sample}(\textit{dist-op}(v_1, \ldots, v_k))\rangle\,\mathcal{E}' \longrightarrow x\,\Xi \mid \mathcal{E}\,\langle x \leftarrow \text{return } v\rangle\,\mathcal{E}'$$

where $\textit{dist-op}: (\alpha_1, \ldots, \alpha_n) \to_{\text{dist}} \alpha$ is a basic distribution, each $v_i \in [\![\alpha_i]\!]$ $v \in [\![\alpha]\!]$.

N.b., we have one such transition, for every $v \in [\![\alpha]\!]$. The transition system is thus nondeterministic.

A probabilistic interpretation will be superimposed later.

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$x \mid \langle x \leftarrow \text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\langle bs \leftarrow C[()] \rangle$$
$$x \mid \langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1) \rangle$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\langle bs \leftarrow C[()] \rangle$$
$$bs\, x \mid \langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1) \rangle$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs \leftarrow \text{do } p \leftarrow (\text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,\ ()\ )) \text{ in}\\ &\qquad\qquad \text{match } p \text{ as } (b,u) \text{ in do } bs' \leftarrow C[u] \text{ in return } (b : bs')\rangle\\ &bs\, x \mid \langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}&\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\qquad\qquad \text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle p \leftarrow \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,\ ()\ )\rangle\\ &\langle bs \leftarrow \text{match } p \text{ as } (b,u) \text{ in do } bs' \leftarrow C[u] \text{ in return } (b : bs')\rangle\\ bs\, x \ \mid\ &\langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow\ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle p \leftarrow \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,\ ()\ )\rangle\\ &\langle bs \leftarrow \text{match } p \text{ as } (b,u) \text{ in do } bs' \leftarrow C[u] \text{ in return } (b : bs')\rangle\\ p\ bs\ x \ &| \ \langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}
&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1) \\
&C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b, u)) \text{ on } [-] \ .
\end{aligned}$$

$$\begin{aligned}
&\langle b \leftarrow \text{sample}(\text{uniform}\{0,1\})\rangle \\
&\langle p \leftarrow \text{return } (b,\ ()\ )\rangle \\
&\langle bs \leftarrow \text{match } p \text{ as } (b,u) \text{ in do } bs' \leftarrow C[u] \text{ in return } (b : bs')\rangle \\
p\ bs\ x \ \mid\ &\langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle
\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}
\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in} \\
&\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)
\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-]\ .\end{aligned}$$

$$\begin{aligned}&\langle b \leftarrow \text{sample}(\text{uniform}\{0,1\})\rangle\\ &\langle p \leftarrow \text{return } (b,\ ())\rangle\\ &\langle bs \leftarrow \text{match } p \text{ as } (b,u) \text{ in do } bs' \leftarrow C[u] \text{ in return } (b : bs')\rangle\\ b\ p\ bs\ x \mid\ &\langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow\ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\begin{array}{ll} & \langle b \leftarrow \text{return } 0 \rangle \qquad\qquad\qquad (0 \text{ randomly chosen}) \\ & \langle p \leftarrow \text{return } (b,\ ()\ ) \rangle \\ & \langle bs \leftarrow \text{match } p \text{ as } (b,u) \text{ in do } bs' \leftarrow C[u] \text{ in return } (b : bs') \rangle \\ b\ p\ bs\ x \mid & \langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1) \rangle \end{array}$$

Reduction for stream corecursion:

$$\begin{array}{lll} \text{corec } (x \mapsto M) \text{ on } E & \longrightarrow & \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in} \\ & & \qquad \text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys) \end{array}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle p \leftarrow \text{return } (0,\ ()\ )\rangle\\ &\langle bs \leftarrow \text{match } p \text{ as } (b,u) \text{ in do } bs' \leftarrow C[u] \text{ in return } (b : bs')\rangle\\ p\ bs\ x \ \mid\ &\langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs \leftarrow \text{match } (0,()) \text{ as } (b,u) \text{ in do } bs' \leftarrow C[u] \text{ in return } (b : bs')\rangle\\ &bs\, x \mid \langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\langle bs \leftarrow \text{do } bs' \leftarrow C[()] \text{ in return } (0 : bs') \rangle$$
$$bs\, x \mid \langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1) \rangle$$

Reduction for stream corecursion:

$$\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}$$
$$\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs' \leftarrow C[()]\rangle\\ &\langle bs \leftarrow \text{return } (0 : bs')\rangle\\ bs\, x \mid &\langle x \leftarrow \text{return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs' \leftarrow C[()]\rangle\\ x \mid &\langle x \leftarrow \text{return } (\text{nth } (0 : bs')\ 1) + (\text{nth } (0 : bs')\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs' \leftarrow C[()]\rangle\\ x \mid &\langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs' \leftarrow C[()]\rangle\\ &bs'\ x \mid \langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\langle bs' \leftarrow \text{do } p \leftarrow (\text{do } b' \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b', ())) \text{ in}$$
$$\text{match } p \text{ as } (b', u) \text{ in do } bs'' \leftarrow C[u] \text{ in return } (b' : bs'')\rangle$$
$$bs'\ x \mid \langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle$$

Reduction for stream corecursion:

$$\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}$$
$$\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle p \leftarrow \text{do } b' \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b',())\rangle\\ &\langle bs' \leftarrow \text{match } p \text{ as } (b',u) \text{ in do } bs'' \leftarrow C[u] \text{ in return } (b' : bs'')\rangle\\ bs'\ x \mid &\langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle p \leftarrow \text{do } b' \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,\ ()\ )\rangle\\ &\langle bs' \leftarrow \text{match } p \text{ as } (b',u) \text{ in do } bs'' \leftarrow C[u] \text{ in return } (b' : bs'')\rangle\\ p\ bs'\ x \mid &\langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}
&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1) \\
&C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .
\end{aligned}$$

$$\begin{aligned}
&\langle b' \leftarrow \text{sample}(\text{uniform}\{0,1\})\rangle \\
&\langle p \leftarrow \text{return } (b',\ ()\ )\rangle \\
&\langle bs' \leftarrow \text{match } p \text{ as } (b',u) \text{ in do } bs'' \leftarrow C[u] \text{ in return } (b' : bs'')\rangle \\
p\ bs'\ x \mid &\langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle
\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}
\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in} \\
&\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)
\end{aligned}$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\begin{array}{ll} & \langle b' \leftarrow \text{sample}(\text{uniform}\{0,1\})\rangle \\ & \langle p \leftarrow \text{return } (b',\,())\rangle \\ & \langle bs' \leftarrow \text{match } p \text{ as } (b',u) \text{ in do } bs'' \leftarrow C[u] \text{ in return } (b' : bs'')\rangle \\ b'\ p\ bs\ x \mid & \langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle \end{array}$$

Reduction for stream corecursion:

$$\begin{array}{lll} \text{corec } (x \mapsto M) \text{ on } E & \longrightarrow & \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in} \\ & & \quad \text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys) \end{array}$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\langle b' \leftarrow \text{return } 1 \rangle \qquad \text{(1 randomly chosen)}$$
$$\langle p \leftarrow \text{return } (b', ()) \rangle$$
$$\langle bs' \leftarrow \text{match } p \text{ as } (b', u) \text{ in do } bs'' \leftarrow C[u] \text{ in return } (b' : bs'') \rangle$$
$$b'\ p\ bs\ x \mid \langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1) \rangle$$

Reduction for stream corecursion:

$$\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}$$
$$\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle p \leftarrow \text{return } (1,\ ()\ )\rangle\\ &\langle bs' \leftarrow \text{match } p \text{ as } (b',u) \text{ in do } bs'' \leftarrow C[u] \text{ in return } (b' : bs'')\rangle\\ p\ bs'\ x \mid &\langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\langle bs' \leftarrow \text{match } (1,()) \text{ as } (b',u) \text{ in do } bs'' \leftarrow C[u] \text{ in return } (b' : bs'')\rangle$$
$$bs'\ x \mid \langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1)\rangle$$

Reduction for stream corecursion:

$$\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}$$
$$\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\langle bs' \leftarrow \text{do } bs'' \leftarrow C[()] \text{ in return } (1 : bs'') \rangle$$
$$bs'\ x \mid \langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1) \rangle$$

Reduction for stream corecursion:

$$\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}$$
$$\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\begin{array}{l} \langle bs'' \leftarrow C[()] \rangle \\ \langle bs' \leftarrow \text{return } (1 : bs'') \rangle \\ bs'\ x \mid \langle x \leftarrow \text{return } (\text{nth } bs'\ 0) + (\text{nth } (0 : bs')\ 1) \rangle \end{array}$$

Reduction for stream corecursion:

$$\begin{array}{l} \text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in} \\ \qquad\qquad \text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys) \end{array}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs'' \leftarrow C[()]\rangle\\ x \mid &\langle x \leftarrow \text{return } (\text{nth } (1 : bs'')\ 0) + (\text{nth } (0 : 1 : bs'')\ 1)\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs'' \leftarrow C[()]\rangle\\ &x \mid \langle x \leftarrow \text{return } 1 + \text{nth } (0:1:bs'')\ 1\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y,x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y:ys)\end{aligned}$$

Illustrative execution of:

$$\begin{aligned}&\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)\\ &C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .\end{aligned}$$

$$\begin{aligned}&\langle bs'' \leftarrow C[()]\rangle\\ x \mid &\langle x \leftarrow \text{return } 1 + \text{nth } (1 : bs'')\ 0\rangle\end{aligned}$$

Reduction for stream corecursion:

$$\begin{aligned}\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ &\text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}\\ &\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)\end{aligned}$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\begin{array}{c} \langle bs'' \leftarrow C[()] \rangle \\ x \mid \langle x \leftarrow \text{return } 1+1 \rangle \end{array}$$

Reduction for stream corecursion:

$$\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}$$
$$\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\begin{array}{r} \langle bs'' \leftarrow C[()] \rangle \\ x \mid \langle x \leftarrow \text{return } 2 \rangle \end{array}$$

Reduction for stream corecursion:

$$\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}$$
$$\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)$$

Illustrative execution of:

$$\text{do } bs \leftarrow C[()] \text{ in return } (\text{nth } bs\ 1) + (\text{nth } bs\ 1)$$
$$C[-] \ := \ \text{corec } (u \mapsto \text{do } b \leftarrow \text{sample}(\text{uniform}\{0,1\}) \text{ in return } (b,u)) \text{ on } [-] \ .$$

$$\langle bs'' \leftarrow C[()] \rangle$$
$$x \mid \langle x \leftarrow \text{return } 2 \rangle$$

Reduction for stream corecursion:

$$\text{corec } (x \mapsto M) \text{ on } E \ \longrightarrow \ \text{do } p \leftarrow M[x := E] \text{ in match } p \text{ as } (y, x') \text{ in}$$
$$\text{do } ys \leftarrow \text{corec } (x \mapsto M) \text{ on } x' \text{ in return } (y : ys)$$

Execution returns 0 with probability $\frac{1}{2}$, and 2 with probability $\frac{1}{2}$.

Termination theorem

Every transition sequence from an initial configuration

$$x_0 \mid \langle x_0 : \tau \leftarrow M \rangle$$

terminates in a terminal configuration

$$x_0 \mid \mathcal{E}\, \langle x_0 : \tau \leftarrow \text{return } V \rangle$$

where V is a return value of type τ.

Structure of the transition system

For every non-terminal configuration

$$x\Xi \mid \mathcal{E}\,\langle x \leftarrow M\rangle\,\mathcal{E}'$$

- If M is of the form $\mathsf{sample}(\mathit{dist\text{-}op}\,(v_1, \ldots, v_n))$ then there is one outgoing transition

$$\longrightarrow \quad x\Xi \mid \mathcal{E}\,\langle x \leftarrow \mathsf{return}\ v\rangle\,\mathcal{E}'$$

  for every $v \in [\![\alpha]\!]$, where $\mathit{dist\text{-}op} : (\alpha_1, \ldots, \alpha_n) \to_{\mathsf{dist}} \alpha$

  This is a probabilistic configuration with law $\mathit{d\text{-}op}\,(v_1, \ldots, v_n)$

- Otherwise, there is exactly one outgoing transition.

  This is a deterministic configuration

(Configurations are identified up to α-equivalence.)

Probabilistic interpretation of operational semantics

The execution of an initial configuration

$$x_0 \mid \langle x_0 : \tau_0 \leftarrow M \rangle$$

is probabilistic and terminating. It should thus be described by a probability measure on the set of terminal configurations $x_0 \mid \mathcal{E}\, \langle x_0 : \tau_0 \leftarrow \mathsf{return}\ V \rangle$.

The required probability measure can be defined in ZF+DC by endowing the set of configurations with the structure of a standard Borel space and establishing relevant measurability properties of the operational semantics, cf. [SYHKW 2016].

Such (cumbersome) work can be avoided by working in a set theory in which all sets are measurable.

Solovay's Axiom LM

LM: Every set of reals is Lebesgue measurable.

Theorem (Solovay): If ZFC+I is consistent then so is ZF+DC+LM.

Theorem (ZF+DC+LM): If μ is Borel probability measure on a standard Borel space X then every subset of X is μ-measurable (i.e., measurable w.r.t. the completion μ^* of μ).

It follows that, for any set X of countable or continuum cardinality, there is a one-one correspondence between:

- powerset probability measures $\mathcal{P}(X) \to [0,1]$, and
- Borel probabiity measures $\mathcal{B}(X) \to [0,1]$, where $\mathcal{B}(X)$ is the Borel σ-algebra of a standard Borel structure on X.

Define:

$$\mathcal{M}_1(X) \ := \ \text{set of powerset probability measures } \mathcal{P}(X) \to [0,1]$$

The type of a configuration $\Xi \mid \mathcal{E}\,\langle x_0 : \tau_0 \leftarrow M_0\rangle$ is τ_0. Define:

$$\begin{aligned} \mathsf{Configs}_\tau &:= \text{the set of all configurations of type } \tau \\ \mathsf{TermConfigs}_\tau &:= \text{the set of all terminal configurations of type } \tau \end{aligned}$$

These sets always have continuum cardinality.

We define:

$$\mathsf{Terminals}_\tau : \mathsf{Configs}_\tau \to \mathcal{M}_1(\mathsf{TermConfigs}_\tau)$$

by well-founded recursion on $\longrightarrow$ (justified by termination).

- If $\Xi \mid \mathcal{E}$ is a terminal configuration of type τ then:

$$\mathsf{Terminals}_\tau(\Xi \mid \mathcal{E}) \;=\; \delta_{\Xi \mid \mathcal{E}}$$

(where δ_x is the Dirac measure $X \mapsto \mathbf{1}_{x \in X}$).

- If $\Xi \mid \mathcal{E}$ is deterministic and $\Xi \mid \mathcal{E} \;\longrightarrow\; \Xi' \mid \mathcal{E}'$ then

$$\mathsf{Terminals}_\tau(\Xi \mid \mathcal{E}) \;=\; \mathsf{Terminals}_\tau(\Xi' \mid \mathcal{E}')$$

- If $\Xi \mid \mathcal{E}$ is probabilistic of the form

$$x\Xi \mid \mathcal{E}\,\langle x \leftarrow \mathsf{sample}(\mathit{d\text{-}op}\,(v_1, \ldots, v_n))\rangle\,\mathcal{E}'$$

where $\mathit{d\text{-}op} : (\alpha_1, \ldots, \alpha_n) \to_{\mathsf{dist}} \alpha$, then

$$\mathsf{Terminals}_\tau(\Xi \mid \mathcal{E}) \;=\; \int_{v \in [\![\alpha]\!]} \mathsf{Terminals}_\tau(x\Xi \mid \mathcal{E}\,\langle x \leftarrow \mathsf{return}\ v\rangle\,\mathcal{E}')\,\mathrm{d}[\![\mathit{d\text{-}op}]\!](v_1, \ldots, v_n)^*$$

Agreement of operational and denotational semantics

For a closed term $M : \tau$ where $\tau = \alpha_1 \times \cdots \times \alpha_k$ define $\mathsf{Values}_\tau(M)$ to be the pushforward of $\mathsf{Terminals}(x_0 \mid \langle x_0 \leftarrow M \rangle)$ along

$$x_0 \mid \mathcal{E} \, \langle x_0 \leftarrow \mathsf{return}(v_1, \ldots, v_k) \rangle \quad \mapsto \quad (v_1, \ldots, v_k)$$

So

$$\mathsf{Values}_\tau(M) \in \mathcal{M}_1(\llbracket \alpha_1 \rrbracket \times \cdots \times \llbracket \alpha_k \rrbracket)$$

Theorem (Adequacy) For any closed term $M : \tau$ we have

$$\mathsf{Values}_\tau(M) \;=\; \llbracket M \rrbracket^* \;.$$

Agreement of operational and denotational semantics

For a closed term $M : \tau$ where $\tau = \alpha_1 \times \cdots \times \alpha_k$ define $\mathsf{Values}_\tau(M)$ to be the pushforward of $\mathsf{Terminals}(x_0 \mid \langle x_0 \leftarrow M \rangle)$ along

$$x_0 \mid \mathcal{E} \, \langle x_0 \leftarrow \mathsf{return}(v_1, \ldots, v_k) \rangle \quad \mapsto \quad (v_1, \ldots, v_k)$$

So

$$\mathsf{Values}_\tau(M) \in \mathcal{M}_1(\llbracket \alpha_1 \rrbracket \times \cdots \times \llbracket \alpha_k \rrbracket)$$

Theorem (Adequacy) For any closed term $M : \tau$ we have

$$\mathsf{Values}_\tau(M) \;=\; \llbracket M \rrbracket^* \;.$$

This justifies the operational semantics as a correct implementation of the intended denotational semantics.

References

[DKPS 2023] S. Dash, Y. Kaddar, H. Paquet & S. Staton, *Affine monads and lazy structures for Bayesian programming*, Proc. POPL 2023.

[Kerstan & König 2013] H. Kerstan & B. König, *Coalgebraic trace semantics for continuous probabilistic transition systems*, LMCS:9(4) 2013

[Chen 2023] R. Chen, *A universal characterization of standard Borel spaces*, JSL:88(2) 2023

[LPT 2003] P. B. Levy, J. Power, H. Thielecke, *Modelling environments in call-by-value programming languages*, Inf. & Comp.:185(2) 2003.

[SYHKW 2016] S. Staton, H. Yang, C. Heunen, O. Kammar, *Semantics for probabilistic programming: higher-order functions, continuous cistributions, and soft constraints*, Proc. LICS 2016.