Streams in Monoidal Categories.

Elena Di Lavore, Giovanni de Felice, Mario Román

November 4th, 2021

Nordic Workshop on Programming Theory (NWPT'21)

Supported by the European Union through the ESF Estonian IT Academy Research Measure.

Main Result

"From a theory of processes to a theory of streams."

Monoidal categories encode theories of processes.

Stream-based programming in monoidal categories.

Construction. Let $\mathbb{C}$ be a symmetric monoidal category. $\textsc{Stream}(\mathbb{C})$ is a symmetric monoidal category with feedback and an id-on-objects functor of the form $\mathbb{C}^{\mathbb{N}} \to \textsc{Stream}(\mathbb{C})$.

Part 0:

Monoidal Categories.

Monoidal Categories

Monoidal categories encode theories of processes that compose sequentially and in parallel. Each theory of processes gives rise to a symmetric monoidal category. Monoidal categories come with a practical graphical calculus.

EXAMPLES:

- Sets and functions. (Set, $\times$)
- Relational queries. (Rel, $\times$)
- Stochastic functions (Stoch, $\times$)
- Linear operators. (Vect, $\otimes$)
- Quantum processes. (Hilb, $\otimes$)

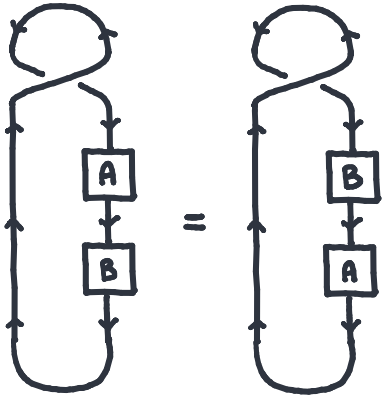


FIG 1. Cyclicity of the trace operator.

$$tr(AB) = tr(BA)$$

Part 1:
Stream-Based Programming

Stream-based Programming

In causal stream-based programming, every declaration is a stream that depends on its previous values.

Lucid

```
n = 0 fby [n+1]
sum x = x + [0 fby [sum x]]
fib = 0 fby [1 fby [ fib + next fib ]]
```

A stream function from $X = X_0, X_1, \ldots$ to $Y = Y_0, Y_1, \ldots$ produces, at each time $t = 0, 1, \ldots$, an output Y_t from all previous inputs $X_0, X_1, \ldots, X_t$.

Stream-based Programming

In causal stream-based programming, every declaration is a stream that depends on its previous values.

```
# Lucid                                  "followed by"
n = 0 fby [n+1]                    ⟶  0,1,2,3,...
sum x = x + [0 fby [sum x]]        ⟶  sum n = 0,1,3,6,10,...
fib = 0 fby [1 fby [fib + next fib]]  ⟶  fib = 0,1,1,2,3,5,8,13...
```

A stream function from $X = X_0, X_1, \ldots$ to $Y = Y_0, Y_1, \ldots$ produces, at each time $t = 0, 1, \ldots$, an output Y_t from all previous inputs $X_0, X_1, \ldots, X_t$.

How to bring streams to any monoidal category?

Stream-based Programming

Plan for this talk.

1. Construct a theory of streams over an arbitrary theory of processes.
2. Describe the theory of streams with a canonical coinductive definition.
3. Example: Recover the classical theory of "cartesian" streams.
4. Example: Obtain a notion of stochastic stream.
5. Feedback syntax for streams.

Part 2:

Stream Transformers

Stream Transformers

Construction (Sprunger, Katsumata, 2019). A stateful morphism sequence is a family of morphisms that, at each instant in time $t = 0, 1, 2, \ldots$, read from a memory M_{t-1}, take an input X_t, produce an output Y_t, and write to a new memory M_t.

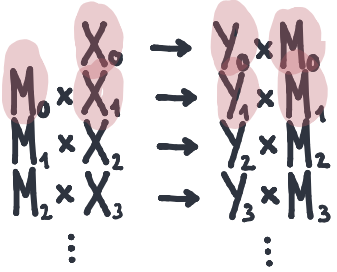


Fig 1. Stateful sequence.

Questions.

- Can we generalize to monoidal categories?
- How to justify this construction?
- Technical point: when are two sequences equal?

Stream Transformers

Construction (Sprunger, Katsumata, 2019). A stateful morphism sequence is a family of morphisms that, at each instant in time $t = 0, 1, 2, \ldots$, read from a memory M_{t-1}, take an input X_t, produce an output Y_t, and write to a new memory M_t.

$$
\begin{array}{ccc}
X_0 & \to & Y_0 \times M_0 \\
M_0 \times X_1 & \to & Y_1 \times M_1 \\
M_1 \times X_2 & \to & Y_2 \times M_2 \\
M_2 \times X_3 & \to & Y_3 \times M_3 \\
\vdots & & \vdots
\end{array}
$$

Fig 1. Stateful sequence.

Questions.

- Can we generalize to monoidal categories?
  Yes (Román, 2020).
- How to justify this construction?
  It's a final coalgebra (DFR, 2021).
- Technical point: when are two sequences equal?
  Dinaturality until stage n (DFR, 2021).

Stream Transformers

Construction (DFR, 21). Let $\mathbb{C}, \otimes$ be a symmetric monoidal category. The symmetric monoidal category $\mathrm{Stream}_{\mathbb{C},\otimes}$ has the same objects as $\mathbb{C}^{\mathbb{N}}$ and stateful morphism sequences as morphisms,

$$\mathrm{Stream}(X;Y) = \lim_n \int^{M_0,\ldots,M_n} \prod_{i=0}^{n} \hom(M_{i-1} \otimes X_i, Y_i \otimes M_i).$$

Stream Transformers

Construction (DFR, 21). Let $\mathbb{C}, \otimes$ be a symmetric monoidal category. The symmetric monoidal category $\mathrm{Stream}_{\mathbb{C},\otimes}$ has the same objects as $\mathbb{C}^{\mathbb{N}}$ and stateful morphism sequences as morphisms,

$$\mathrm{Stream}(X;Y) = \lim_{n} \int^{M_0,\ldots,M_n} \prod_{i=0}^{n} \hom(M_{i-1} \otimes X_i, Y_i \otimes M_i).$$

Limit $n \to \infty$. For each n, describe the sequence until n.

Coend. Dinatural existential quantifier. Choose memory channels.

Input memory. Input. Output. Output memory.

Stateful sequence.

Stream Transformers

Streams have a coinductive definition. $\mathrm{Str}(A_0, A_1, \ldots) = A_0 \times \mathrm{Str}(A_1, A_2, \ldots)$.

> "A stream with types $\mathbf{A} = A_0, A_1, A_2, \ldots$ is an element of A_0 together with a stream with types $A_1, A_2, \ldots$."

Stream transformers can be also given a coinductive definition.

> "A stream transformer from $\mathbf{X} = X_0, X_1, \ldots$ to $\mathbf{Y} = Y_0, Y_1, \ldots$ is a process from X_0 to Y_0 communicating along a memory with a stream transducer from $X_1, X_2, \ldots$ to $Y_1, Y_2, \ldots$."

$$\mathrm{Stream}(\mathbf{X};\mathbf{Y}) = \int^{M} \hom(X_0, Y_0 \otimes M) \times \mathrm{Stream}(M \otimes X_1, X_2, \ldots; Y_1, Y_2, \ldots).$$

Memory channel

Coend: natural existential quantifier

Process from the base category

Another stream

Stream Transformers

$$\mathrm{Stream}(X;Y) = \int^{M} \hom(X_0, Y_0 \otimes M) \times \mathrm{Stream}(M \otimes X_1, X_2, \ldots; Y_1, Y_2, \ldots).$$

Memory channel

Coend: natural existential quantifier

Process from the base category

Another stream

This construction:

1. Generalizes the stateful sequences of Sprunger & Katsumata.
2. Coinductive definition.
3. Provides a natural notion of stream equivalence.
4. Works in arbitrary monoidal categories.

Part 3:

Examples

Cartesian Processes

Definition (Uustalu, Vene, 2008, adapted). A **causal stream function** is a morphism of the coKleisli category of $M : \mathbb{C}^{\mathbb{N}} \to \mathbb{C}^{\mathbb{N}}$.

$$\frac{X \to Y}{\begin{array}{l} X_0 \to Y_0 \\ X_1 \to Y_1 \\ X_2 \to Y_2 \\ X_3 \to Y_3 \\ \vdots \quad \vdots \quad \vdots \end{array}}$$

Memoryless process in $\mathbb{C}^{\mathbb{N}}$.

$$\frac{MX \to Y}{\begin{array}{l} X_0 \to Y_0 \\ X_0 * X_1 \to Y_1 \\ X_0 * X_1 * X_2 \to Y_2 \\ X_0 * X_1 * X_2 * X_3 \to Y_3 \\ \quad \vdots \qquad \vdots \end{array}}$$

Stream transformer in coKleisli(M).

Theorem (DFR, 2021). There is an equivalence of categories

$$\mathbf{Stream}(\mathbf{Set}) \cong \mathbf{coKleisli}(M).$$

Stochastic Processes

Definition (Ross, 1996, adapted). A stochastic process from X to Y is a sequence of functions $f_n : X_0 \times \cdots \times X_n \to D(Y_0 \times \cdots \times Y_n)$ such that f_n is the marginal distribution of f_{n+1} in the first n variables.

$$\begin{array}{ccc} X_0 \times \cdots \times X_{n+1} & \xrightarrow{f_{n+1}} & D(Y_0 \times \cdots \times Y_{n+1}) \\ \pi \downarrow & = & \downarrow D\pi \\ X_0 \times \cdots \times X_n & \xrightarrow{f_n} & D(Y_0 \times \cdots \times Y_n) \end{array}$$

Theorem (DFR, 2021). There is an equivalence of categories

$$\mathsf{Stream}(\mathsf{Stoch}) \cong \mathsf{StochasticProcesses}.$$

Part 4:

Graphical Syntax & Further Work.

Graphical Syntax

Stateful sequences came with a convenient graphical calculus where input/output goes top-bottom and memory goes left to right.

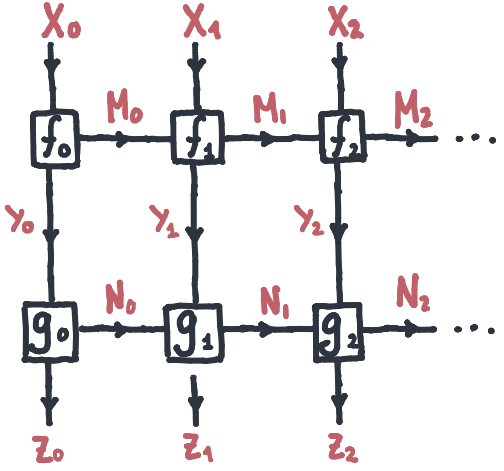


Fig 1. A stateful sequence
$f_n : M_{n-1} \otimes X_n \to Y_n \otimes M_n$
and a stateful sequence
$g_n : N_{n-1} \otimes Y_n \to Z_n \otimes N_n$
compose into a stateful sequence
$(f \fatsemi g)_n : N_{n-1} \otimes M_{n-1} \otimes X_n \to Z_n \otimes M_n \otimes N_n$.

Formalization in terms of "open diagrams": Román 2019, 2020.

Syntax: Categories with Feedback

Can we have a more compact notation? Categories with feedback are a weakening of traces with a graphical calculus in terms of loops.

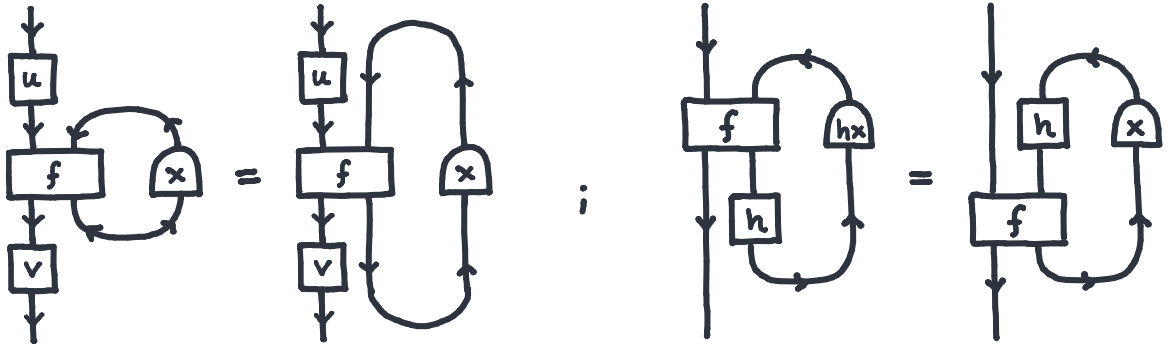
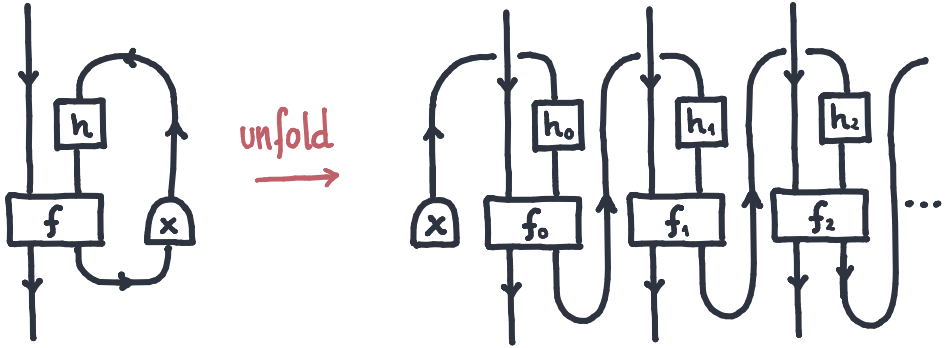


Fig 1. Some axioms for "pointed" or "guarded" feedback.

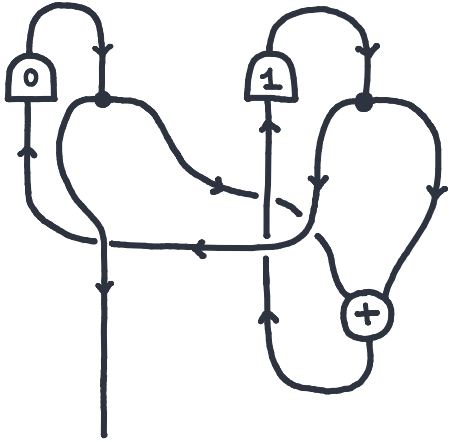
Recipe: use categories with feedback for syntax and stream transformers for semantics.

Syntax: Categories with Feedback

Theorem (DFR, 21). The monoidal category $\mathbf{Stream}_{\mathbb{C},\otimes}$ has feedback. Thus, there exists a feedback-preserving functor from the syntactic category with feedback to $\mathbf{Stream}_{\mathbb{C},\otimes}$.



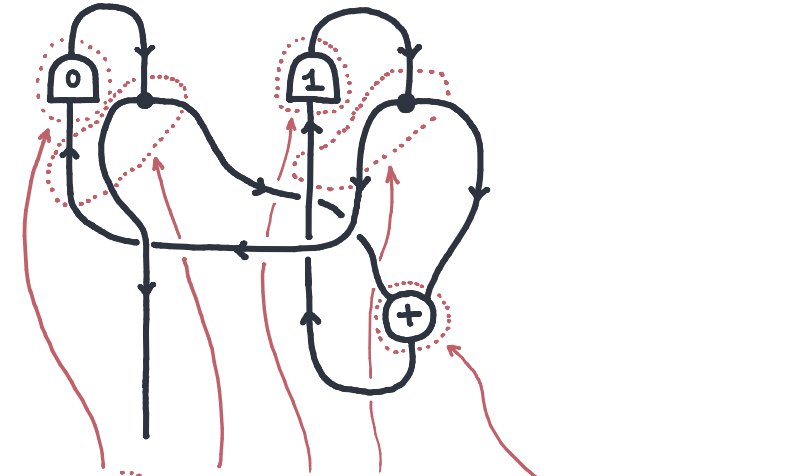
Diagrammatic Stream Programming




```
fib = 0 fby [1 fby [ fib + next fib ]]
```

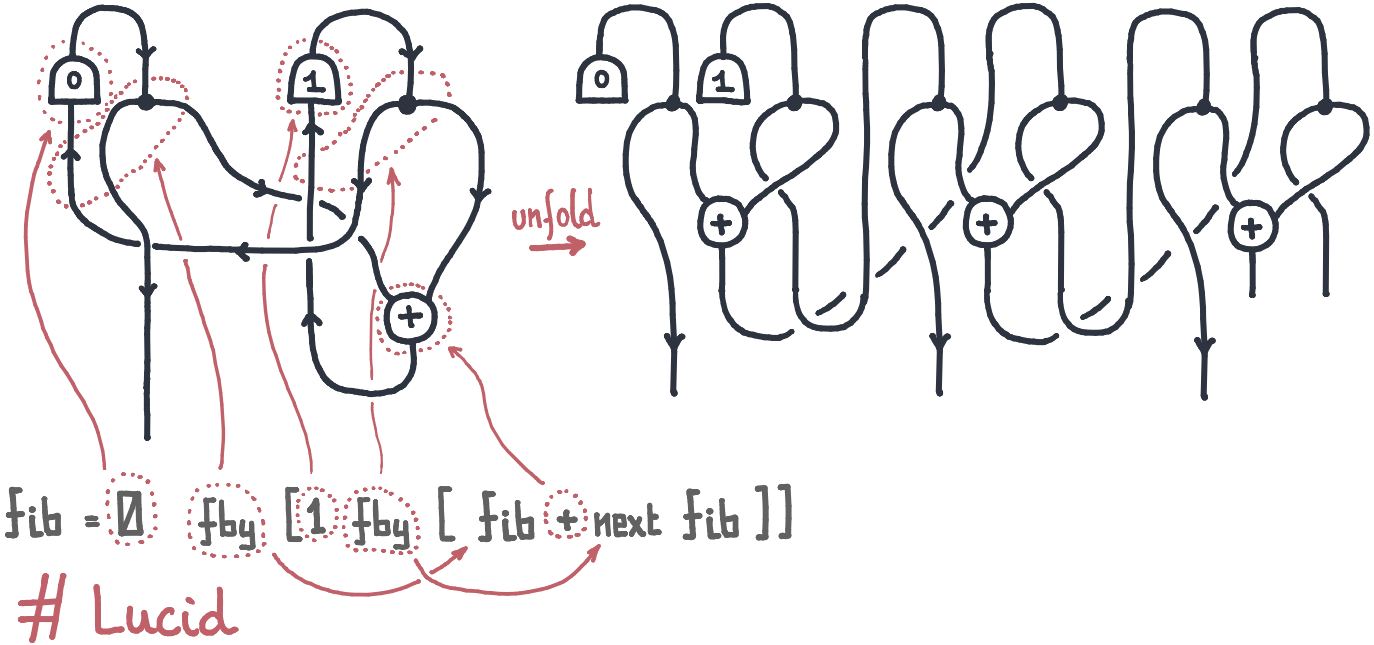
Lucid

Diagrammatic Stream Programming

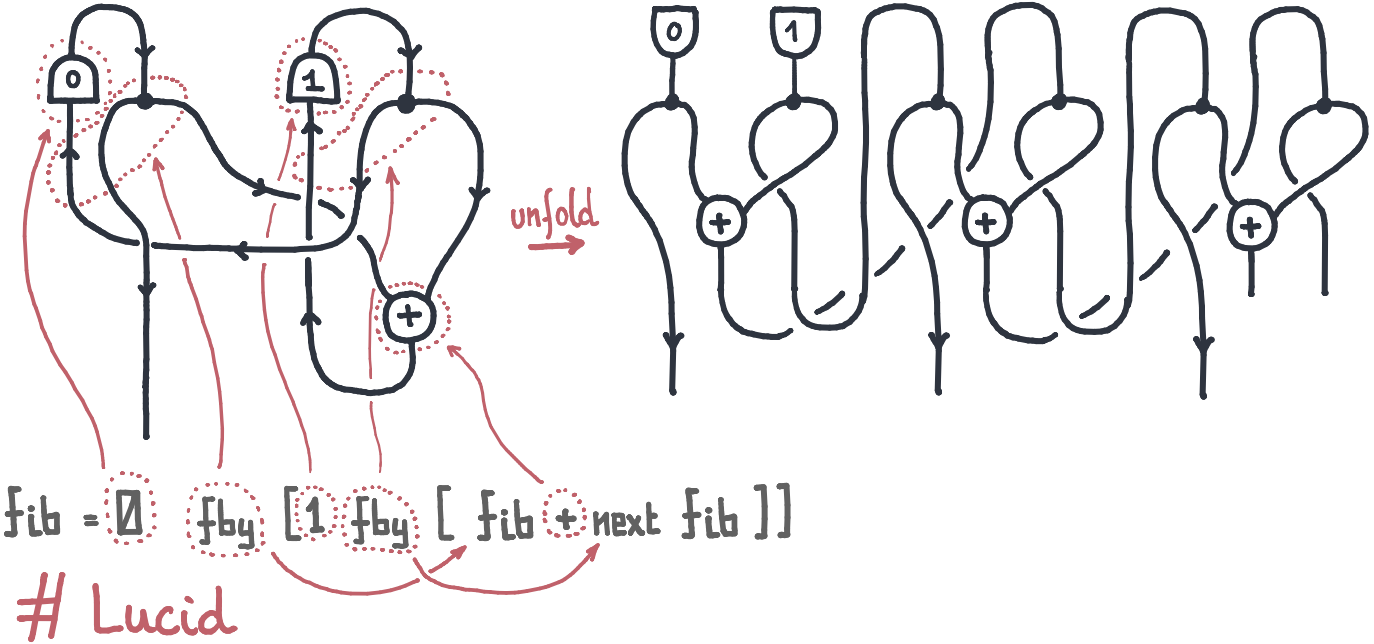


Lucid

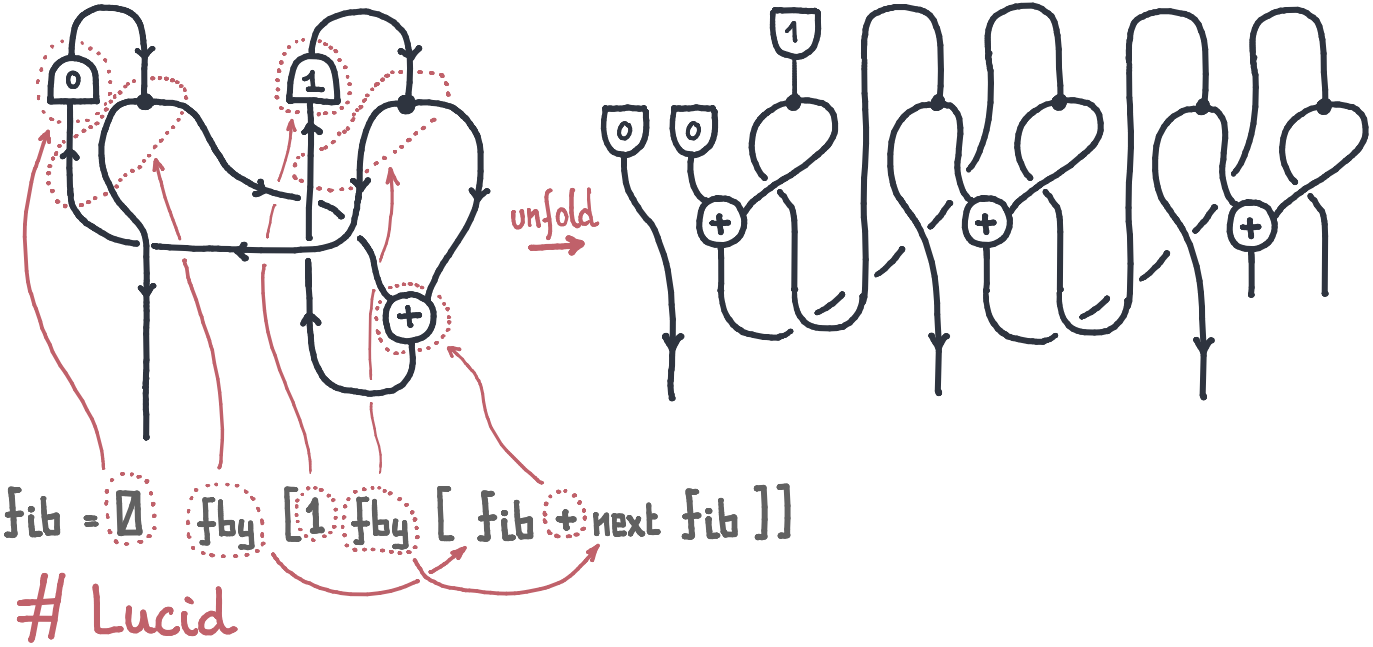
Diagrammatic Stream Programming



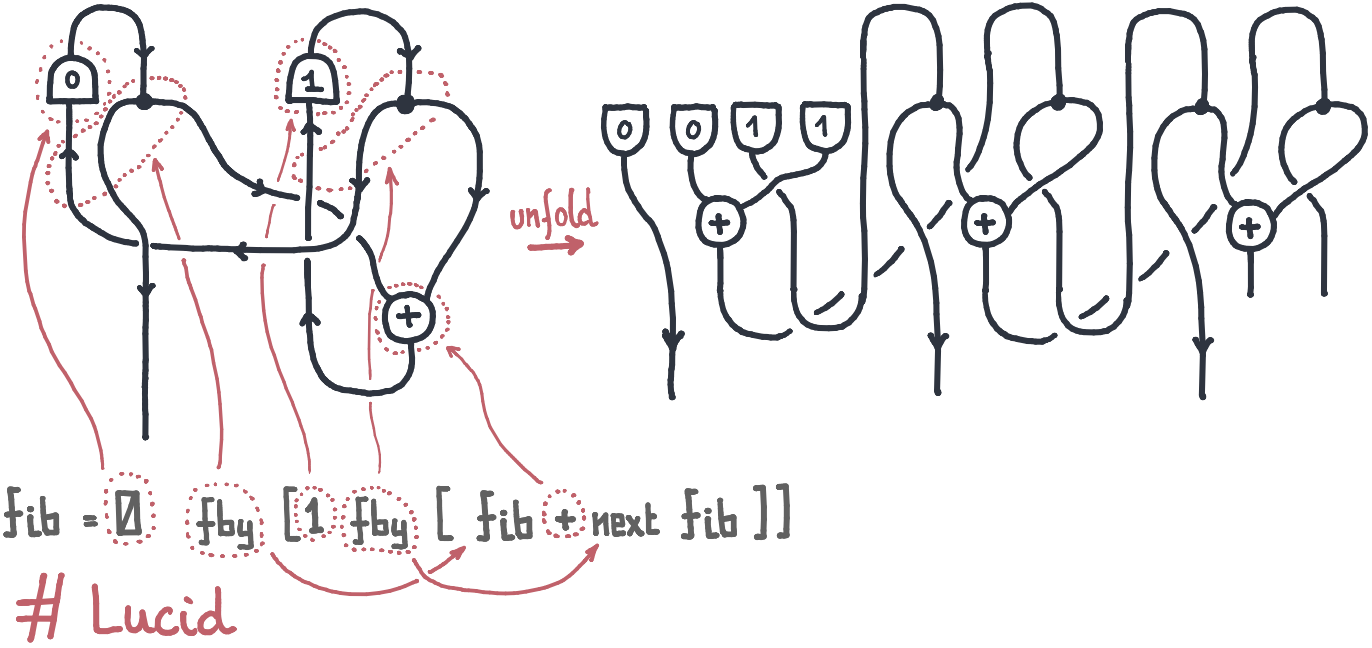
Diagrammatic Stream Programming



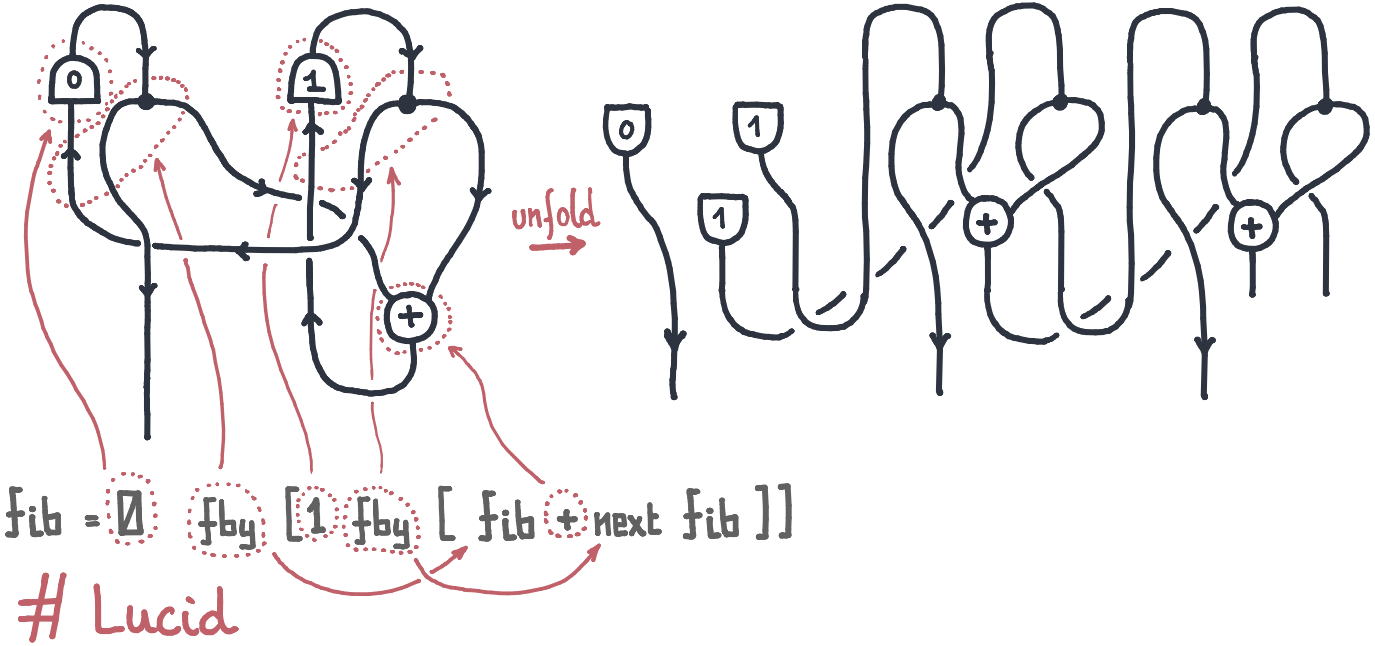
Diagrammatic Stream Programming



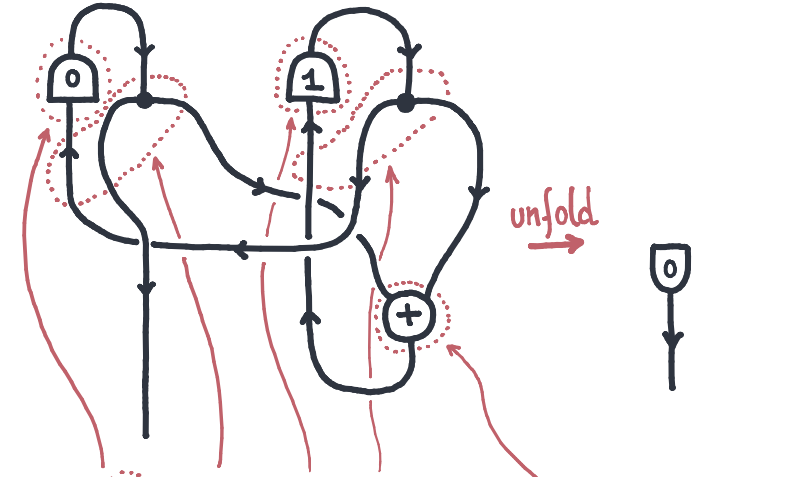
Diagrammatic Stream Programming

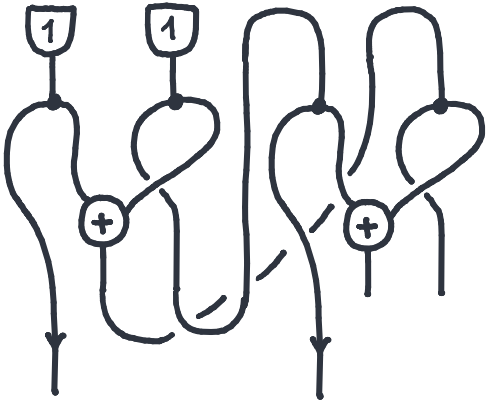


Diagrammatic Stream Programming



Diagrammatic Stream Programming

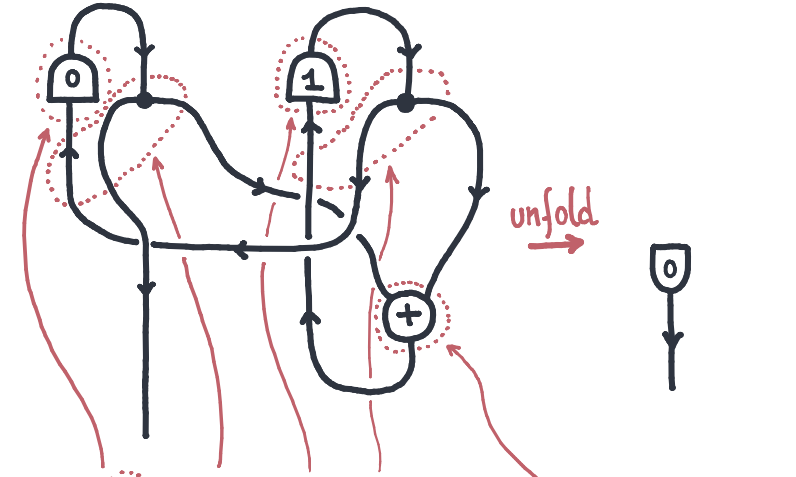


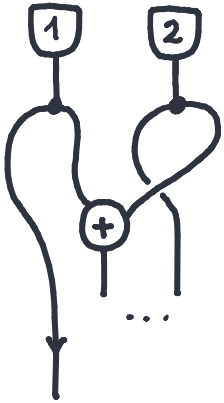


fib = 0 fby [1 fby [fib + next fib]]

Lucid

Diagrammatic Stream Programming





References

Katis, Sabadini, Walters. Feedback, trace, and fixed-point semantics.
RAIRO-Theoretical Informatics and Applications. Tome 36, 2002.

Katsumata, Sprunger. Differentiable Causal Computations via Delayed Trace.
Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), 2019.

Román. Open diagrams via Coend Calculus. (ACT'20).

Uustalu, Vene. The Essence of Dataflow Programming.
Lecture Notes in Computer Science, 2006.

PROFUNCTORS

DEFINITION (Communicating composition). Let $P : \mathbb{A}^{op} \times \mathbb{B} \to \mathbf{Set}$ and $Q : \mathbb{B}^{op} \times \mathbb{C} \to \mathbf{Set}$ be profunctors with $\mathbb{B}, \otimes$ a monoidal category. The communicating composition $(P \odot Q) : \mathbb{A}^{op} \times \mathbb{B}^{op} \times \mathbb{B} \times \mathbb{C} \to \mathbf{Set}$ is defined by

$$(P \odot Q)(A, B, B', C) = \int^{M} P(A, B \otimes M) \times Q(M \otimes B', C).$$

REMARK. $\mathbf{Stream}_{\mathbb{C}}$ can be obtained as the greatest fixpoint of

$$\mathbf{Stream} = \hom \odot \mathbf{Stream}.$$

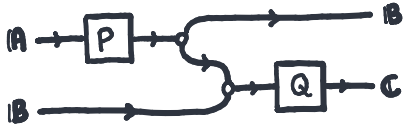


FIG 1. Communicating composition in the graphical calculus of profunctors.

Stream-based Programming

Construction (Uustalu, Vene, 2016). Causal stream functions are morphisms of the cokleisli category of a suitable comonad.

The monoidal category $\mathbb{C}^{\mathbb{N}}$ has families of morphisms of $\mathbb{C}$ indexed by time $t=0,1,\dots$ A "memory" comonad $M : \mathbb{C}^{\mathbb{N}} \to \mathbb{C}^{\mathbb{N}}$ can endow each function with a context representing past inputs, $(MX)_n = X_0 \times \cdots \times X_n$.

$X \to Y$

$$
\begin{aligned}
X_0 &\to Y_0 \\
X_1 &\to Y_1 \\
X_2 &\to Y_2 \\
X_3 &\to Y_3 \\
\vdots \quad &\quad \vdots \quad \quad \vdots
\end{aligned}
$$

Memoryless process in $\mathbb{C}^{\mathbb{N}}$!

$MX \to Y$

$$
\begin{aligned}
X_0 &\to Y_0 \\
X_0 \times X_1 &\to Y_1 \\
X_0 \times X_1 \times X_2 &\to Y_2 \\
X_0 \times X_1 \times X_2 \times X_3 &\to Y_3 \\
\vdots \quad &\quad \vdots
\end{aligned}
$$

Stream transformer in cokleisli(M).

Stream-based Programming

Idea (Naïve Mario). Just substitute $(\times)$ by a tensor product $(\otimes)$.

$$MX \to Y$$

Naïve Mario's monoidal stream transformer

$$\begin{array}{l} X_0 \to Y_0 \\ X_0 \otimes X_1 \to Y_1 \\ X_0 \otimes X_1 \otimes X_2 \to Y_2 \\ X_0 \otimes X_1 \otimes X_2 \otimes X_3 \to Y_3 \\ \quad \vdots \qquad\qquad \vdots \end{array}$$

Theorem (DFR, 2021). The functor $M : \mathbb{C}^{\mathbb{N}} \to \mathbb{C}^{\mathbb{N}}$ defined by $(MX)_n = X_0 \otimes \cdots \otimes X_n$ has a monoidal comonad structure *if and only if* the category $\mathbb{C}, \otimes$ is cartesian.

So, we want a different generalization to arbitrary monoidal categories.

Stream Transformers

Streams have a coinductive definition. $\mathrm{Str}(A_0, A_1, \ldots) = A_0 \times \mathrm{Str}(A_1, A_2, \ldots)$.

> "A stream with types $\mathbf{A} = A_0, A_1, A_2, \ldots$ is an element of A_0 together with a stream with types $A_1, A_2, \ldots$."

Stream transformers can be also given a coinductive definition.

> "A stream transformer from $\mathbf{X} = X_0, X_1, \ldots$ to $\mathbf{Y} = Y_0, Y_1, \ldots$ is a process from X_0 to Y_0 communicating along a memory with a stream transducer from $X_1, X_2, \ldots$ to $Y_1, Y_2, \ldots$."

$$\mathrm{Stream}(\mathbf{X};\mathbf{Y}) = \int^{M} \hom(X_0, Y_0 \otimes M) \times \mathrm{Stream}(M \otimes X_1, X_2, \ldots; Y_1, Y_2, \ldots).$$

Monoidal Categories

Cartesian monoidal categories encode these theories of processes that allow for copying and discarding.

- Plain functions are cartesian.
- Stochastic functions are not cartesian.
- Partial functions are not cartesian.

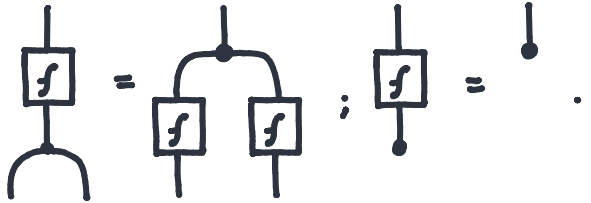


FIG 1. Copy and discard.

FIG. 2. Failure of stochastic copy; failure of partial discard.

Monoidal Categories

Cartesian monoidal categories encode these theories of processes that allow for copying and discarding.

- Plain functions are cartesian.
- Stochastic functions are not cartesian.
- Partial functions are not cartesian.

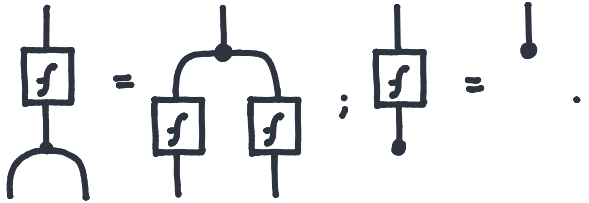


FIG 1. Copy and discard.

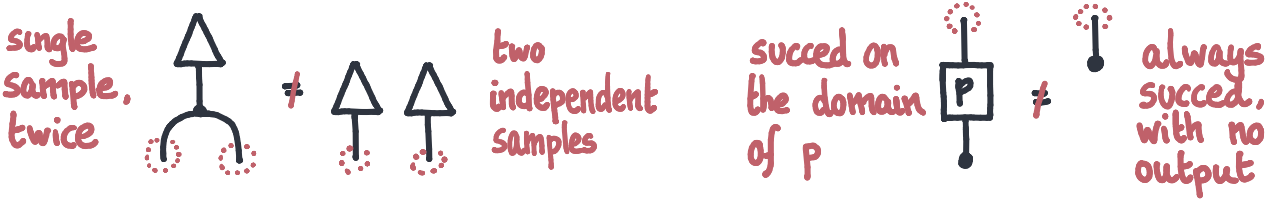


FIG. 2. Failure of stochastic copy; failure of partial discard.