Pisa 29 February 2024

EFFECTFUL TRACE SEMANTICS VIA EFFECTFUL STREAMS

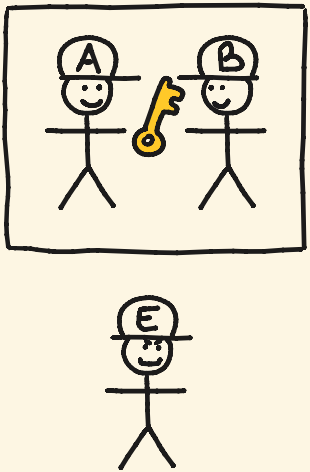
Filippo Bonchi
Università di Pisa

Elena Di Lavore
Università di Pisa

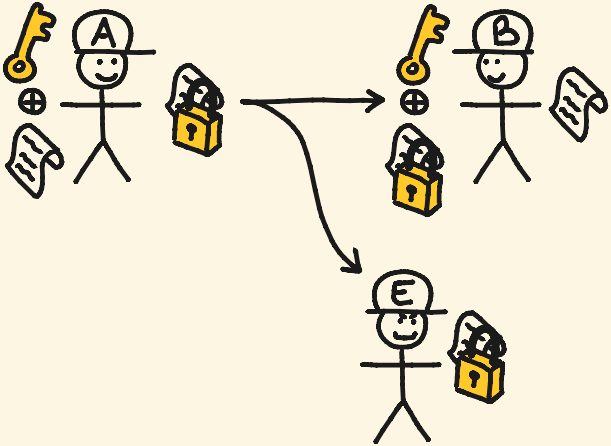
Mario Román
University of Oxford

ONE-TIME PAD PROTOCOL

1. share a key through a secure channel



2. send an encrypted message through a public channel



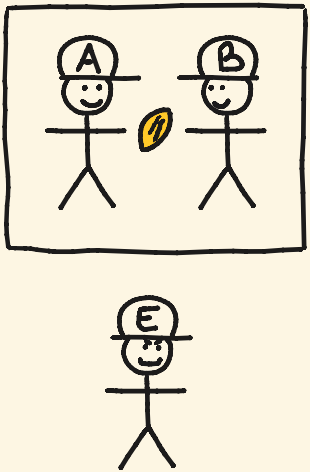
REPEATING THE ONE-TIME PAD

Sending n messages securely requires n private keys
↳ not very useful

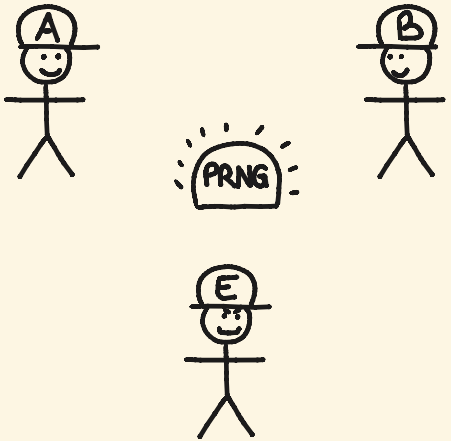
⇒ • privately share a seed
• use identical pseudorandom number generator to obtain a new key for each message

STREAM CIPHER PROTOCOL (1)

1. share a seed through a secure channel

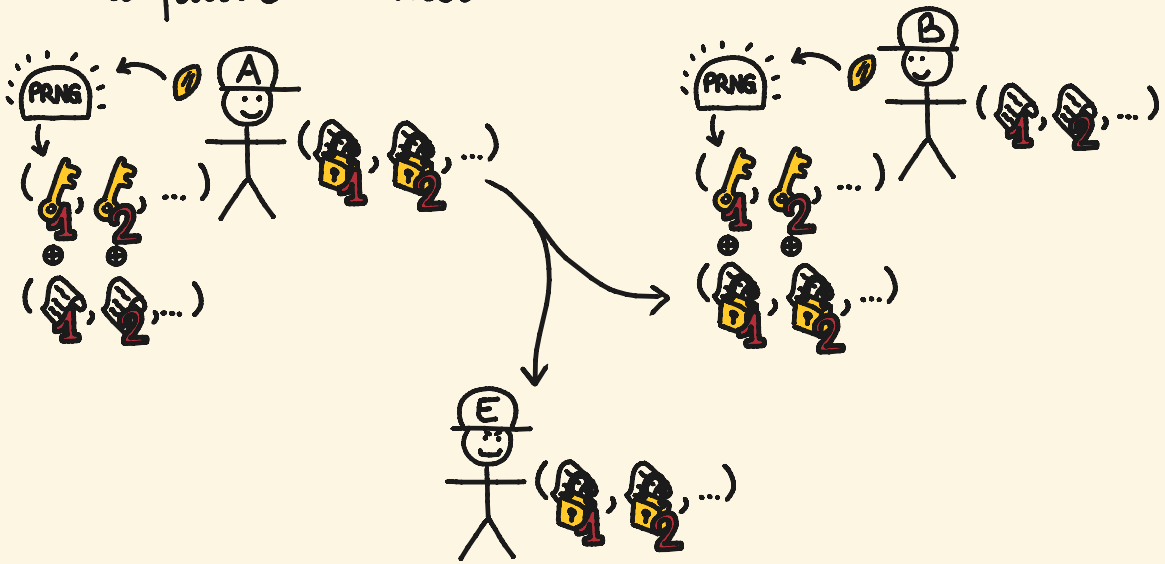


2. share a pseudorandom number generator



STREAM CIPHER PROTOCOL (2)

3. send a stream of encrypted messages through a public channel



(CO) INDUCTION

- inductive data types are initial algebras (of polynomial functors)

data Nat = 0 | succ Nat

$N : X \to 1 + X$

data ListNat = [] | cons Nat ListNat

$L : X \to 1 + Nat \times X$

(CO) INDUCTION

- coinductive data types are final coalgebras (of polynomial functors)

codata CoNat = 0 | succ CoNat

$N : X \to 1 + X$

codata CoListNat = [] | cons Nat CoListNat

$L : X \to 1 + \mathrm{Nat} \times X$

codata StreamNat = cons Nat StreamNat

$S : X \to \mathrm{Nat} \times X$

PRACTICAL COINDUCTION

- $(\!|-|\!) : \mathsf{Nat} \to \mathsf{StreamNat}$
  $(\!|n|\!)^\circ := n$
  $(\!|n|\!)^+ := (\!|n|\!)$
- $- \cdot - : \mathsf{Nat} \times \mathsf{StreamNat} \to \mathsf{StreamNat}$
  $(n \cdot \mathscr{x})^\circ := n \cdot \mathscr{x}^\circ$
  $(n \cdot \mathscr{x})^+ := \mathscr{x}^+$
- $\leq \subseteq \mathsf{StreamNat} \times \mathsf{StreamNat}$ ⇝ lexicographic order
  $\mathscr{x} \leq y$ iff $\mathscr{x}^\circ \leqslant y^\circ$ (1)
  and, if $\mathscr{x}^\circ = y^\circ$, then $\mathscr{x}^+ \leq y^+$ (2)

PRACTICAL COINDUCTION

CLAIM

The lexicographic order $\leq$ on StreamNat is transitive.

RECALL

$x \leq y$ iff

(1) $x^\circ \leq y^\circ$

and (2), if $x^\circ = y^\circ$, then $x^+ \leq y^+$

COINDUCTIVE PROOF

Suppose $x \leq y \leq z$.

By (1), $x^\circ \leq y^\circ \leq z^\circ$.

Then, $x^\circ \leq z^\circ$ by transitivity in Nat.

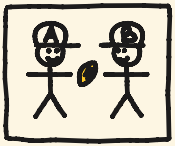
Suppose $x^\circ = z^\circ$. Then $x^\circ = y^\circ$ and $y^\circ = z^\circ$.

By (2), $x^+ \leq y^+$ and $y^+ \leq z^+$.

By coinduction, $x^+ \leq z^+$.

□

STREAM CIPHER PROTOCOL: COMPONENTS

1. 

$rand : I \to S$ ⇝ generate the seed

$setSeed : S \rightsquigarrow I$ ⇝ share the seed

$getSeedA : I \rightsquigarrow S$ ⇝ A and B get the seed

$getSeedB : I \rightsquigarrow S$

2. 

$prng : S \to S \otimes M$ ⇝ pseudorandom number generator

3. 

$-\oplus- : M \otimes M \to M$ ⇝ encrypt / decrypt messages

STREAM CIPHER PROTOCOL COINDUCTIVELY

$$\begin{aligned}
&\mathsf{seedGen}()^{\circ} = \mathbf{do}\\
&\quad \mathsf{rand}() \to s\\
&\quad \mathsf{setSeed}(s) \rightsquigarrow ()\\
&\quad \mathbf{return}\ ()
\end{aligned}$$

$$\begin{aligned}
&\mathsf{seedGen}()^{+\circ} = \mathbf{do}\\
&\quad \mathbf{return}\ ()
\end{aligned}$$

$$\mathsf{seedGen}^{++} = \mathsf{seedGen}^{+}$$

$$\begin{aligned}
&\mathsf{alice}(m)^{\circ} = \mathbf{do}\\
&\quad \mathsf{getSeedA}() \rightsquigarrow (s)\\
&\quad \mathsf{prng}(s) \to (s', k)\\
&\quad \mathbf{return}\ (s', m \oplus k)
\end{aligned}$$

$$\begin{aligned}
&\mathsf{alice}(s, m)^{+\circ} = \mathbf{do}\\
&\quad \mathsf{prng}(s) \to (s', k)\\
&\quad \mathbf{return}\ (s', m \oplus k)
\end{aligned}$$

$$\mathsf{alice}(s, m)^{++} = \mathsf{alice}(s, m)^{+}$$

STREAM CIPHER PROTOCOL COINDUCTIVELY

$$\begin{aligned}
&bob(m)^{\circ} = \mathbf{do}\\
&\quad getSeedB() \rightsquigarrow (s)\\
&\quad prng(s) \to (s', k)\\
&\quad \mathbf{return}\ (s', m \oplus k)
\end{aligned}$$

$$\begin{aligned}
&bob(s, m)^{+\circ} = \mathbf{do}\\
&\quad prng(s) \to (s', k)\\
&\quad \mathbf{return}\ (s', m \oplus k)
\end{aligned}$$

$$bob(s, m)^{++} = bob(s, m)^{+}$$

$$\begin{aligned}
&eve(m)^{\circ} = \mathbf{do}\\
&\quad \mathbf{return}\ (m)
\end{aligned}$$

$$eve(m)^{+} = eve(m)$$

OUTLINE

- effectful categories
- effectful streams
- effectful trace semantics
- causal processes

(PRO) MONADS FOR NOTIONS OF COMPUTATIONS

- Monads add effects to computations
  - ↳ failure and exceptions
    probability
    reading/writing to memory
    input/output

[Moggi 1991]

EXAMPLES

- objects are sets $A, B, \ldots$
- parallel composition is cartesian product $A \times B$
- unit object is $1 := \{*\}$

	MORPHISMS	MONAD	$A \to B$
Set	functions	identity	$A \to B$
Rel	relations	powerset	$A \to \mathcal{P}(B)$
pStoch	stochastic partial functions	subdistributions	$A \to \mathcal{D}_{\leq 1}(B)$
State	functions with global state	state	$S \times A \to S \times B$

Set : FUNCTIONS

Semantics for deterministic and total computations.

- morphisms of type $A \to B$ are functions $f : A \to B$
- composition is function composition

$$f;g : A \to B \to C$$
$$a \mapsto f(a) \mapsto g(f(a))$$

Rel : RELATIONS

Semantics for non-deterministic computations.

The powerset monad adds non-determinism.

$$\begin{aligned} \mathcal{P} : \mathbf{Set} &\to \mathbf{Set} \\ A &\mapsto \{X \subseteq A\} \end{aligned}$$

- morphisms of type $A \to B$ are relations $f \subseteq A \times B$, i.e. functions $f : A \to \mathcal{P}(B)$.
- composition is relational composition

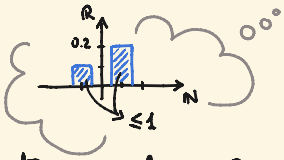
$$f;g\,(a) := \{c \in C \mid \exists\, b \in f(a) \;\wedge\; c \in g(b)\}$$

pStoch : PARTIAL STOCHASTIC FUNCTIONS

Semantics for partial stochastic computations.

The subdistribution monad adds failure and probability.

$$\mathcal{D}_{\leq 1} : \mathbf{Set} \to \mathbf{Set}$$
$$A \mapsto \{\sigma : A \to [0,1] \mid \mathrm{supp}\,\sigma \text{ is finite} \wedge \sum_{a \in A} \sigma(a) \leq 1\}$$



- morphisms of type $A \to B$ are functions $f : A \to \mathcal{D}_{\leq 1}(B)$.

$$\begin{cases} f(b|a) & \rightsquigarrow \text{probability of } b \text{ given } a \\ f(\perp|a) := 1 - \sum_{b \in B} f(b|a) & \rightsquigarrow \text{probability of failure} \end{cases}$$

- composition is $f ; g(c|a) := \sum_{b \in B} f(b|a) \cdot g(c|b)$.

State : STATEFUL FUNCTIONS

Semantics for computations with a global state or memory.
The state promonad adds a global state S.

$$St : Set^{op} \times Set \longrightarrow Set$$
$$(A, B) \longmapsto \{ f : S \times A \to S \times B \}$$

- morphisms of type $A \to B$ are functions $f : S \times A \to S \times B$.

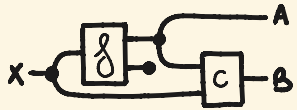
- composition is function composition

$$(s, a) = g(f(s, a))$$

STRING DIAGRAMS & DO-NOTATION

- Symmetric monoidal categories are theories of processes
- String diagrams and do-notation are convenient syntax

$$\nu_X ; ((f ; (\nu_A \otimes \varepsilon_B)) \otimes 1_X) ; (1_A \otimes c)$$

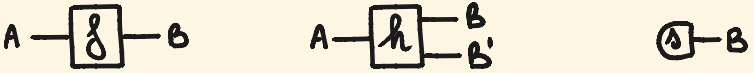



```
cond (x) = do
  f(x) → (a, b)
  c(a, x) → b'
  return (a, b')
```

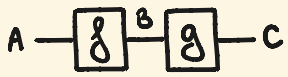
[Gadducci, Montanari 1998; Bruni, Lanese, Montanari 2005; Corradini, Gadducci 1997; Bonchi, Sobociński, Zanasi 2015]

STRING DIAGRAMS & DO-NOTATION

- resources $A, B, \ldots$ and processes $f : A \to B$, with possibly multiple inputs/outputs $h : A \to B \otimes B'$, $s : I \to B$



- sequential composition $f ; g : A \to C$ and identities $1_A : A \to A$

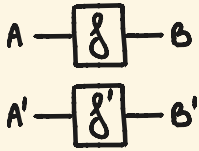



```
comp f g (a) = do
  | f(a) → b
  | g(b) → c
  | return (c)
```


```
id (a) = do
  | return (a)
```

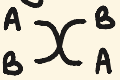
STRING DIAGRAMS & DO-NOTATION

- parallel composition $f \otimes f' : A \otimes A' \to B \otimes B'$




```
tensor ff'(a, a') = do
  f(a) → b
  f'(a') → b'
  return (b, b')
```

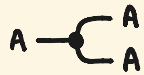
- permuting resources $\sigma_{A,B} : A \otimes B \to B \otimes A$

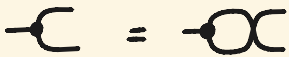



```
swap (a, b) = do
  return (b, a)
```


```
swapNat (a, b) = do
  f(a) → a'
  return (b, a')
```

COPY AND DISCARD

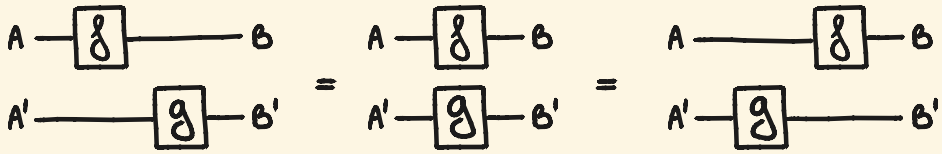




```
copy(a) = do
  return(a, a)

discard(a) = do
  return ()

copyAssoc(a) = do
  return(a, a, a)

copyUnit(a) = do
  return (a)

copyCommut(a) = do
  return(a, a)
```

[Corradini & Gadducci 1999]

THE INTERCHANGE LAW




```
par f g (a, a') = do
  f(a) → b
  g(a') → b'
  return (b, b')
```

=

```
par f g (a, a') = do
  g(a') → b'
  f(a) → b
  return (b, b')
```

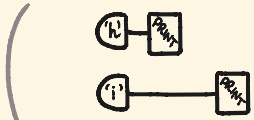
PREMONOIDAL CATEGORIES

Some processes do not interchange.

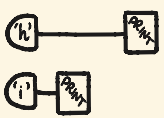
```
printHI() = do
  print('h') ⇝ ()
  print('i') ⇝ ()
  return ()
```

≠

```
printIH() = do
  print('i') ⇝ ()
  print('h') ⇝ ()
  return ()
```



≠



prints "hi"

prints "ih"

⇝ state promonads

⇝ IO monad

STREAM CIPHER PROTOCOL (AGAIN)

$\mathsf{seedGen}()^{\circ} =$ do
- $\mathsf{rand}() \to s$
- $\mathsf{setSeed}(s) \rightsquigarrow ()$
- return $()$

$\mathsf{seedGen}()^{+\circ} =$ do
- return $()$

$\mathsf{seedGen}^{++} = \mathsf{seedGen}^{+}$

$\mathsf{alice}(m)^{\circ} =$ do
- $\mathsf{getSeedA}() \rightsquigarrow (s)$
- $\mathsf{prng}(s) \to (s', k)$
- return $(s', m \oplus k)$

$\mathsf{alice}(s, m)^{+\circ} =$ do
- $\mathsf{prng}(s) \to (s', k)$
- return $(s', m \oplus k)$

$\mathsf{alice}(s, m)^{++} = \mathsf{alice}(s, m)^{+}$

$\mathsf{eve}(m)^{\circ} =$ do
- return (m)

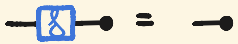
$\mathsf{eve}(m)^{+} = \mathsf{eve}(m)$

PURE COMPUTATIONS

Pure computations are both :

- deterministic

- and total.



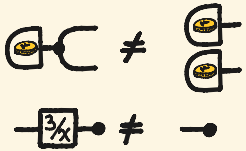
```
copyF(x) = do
  return(f(x), f(x))
```

```
discardF(x) = do
  return()
```

ex $(3 \cdot -) : \mathbb{R} \to \mathbb{R}$

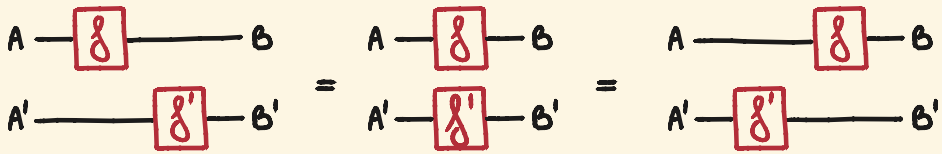
NON-ex $\mathrm{Flip} : 1 \to \mathcal{D}(\{H,T\})$

$(3/-) : \mathbb{R} \to \mathbb{R}$



LOCAL COMPUTATIONS

Local computations interchange,



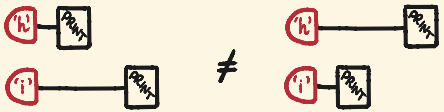

```
localF(a,a') = do
  f(a) → b
  f'(a') → b'
  return (b,b')
```

=

```
localF(a,a') = do
  f'(a') → b'
  f(a) → b
  return (b,b')
```

EFFECTFUL COMPUTATIONS

Effectful computations may have global effects.




```
printHI() = do
  'h'() → c1
  'i'() → c2
  print(c1) ⇝ ()
  print(c2) ⇝ ()
  return ()
```

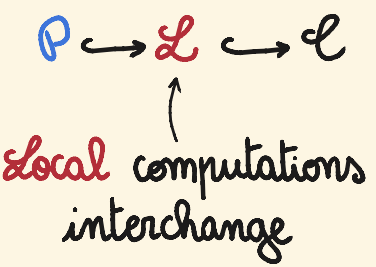
≠

```
printIH() = do
  'h'() → c1
  'i'() → c2
  print(c2) ⇝ ()
  print(c1) ⇝ ()
  return ()
```

EFFECTFUL COPY-DISCARD CATEGORIES

Pure computations can be copied and discarded

Effectful computations may have global effects



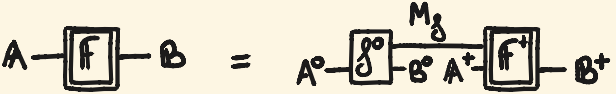
OUTLINE

- effectful categories
- effectful streams
- effectful trace semantics
- causal processes

EFFECTFUL STREAMS

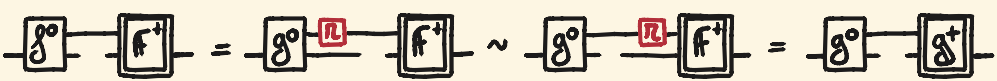
An effectful stream $F : A \to B$ is

- a memory $M_g \in \mathcal{L}$
- a first action $g^\circ : A^\circ \to M_g \otimes B^\circ$ in $\mathcal{C}$
- the rest of the action $F^+ : M_g \cdot A^+ \to B^+$



quotiented by the equivalence relation generated by

$$\begin{cases} g^\circ ; (\pi \otimes 1) = g^\circ \\ F^+ = \pi \cdot g^+ \end{cases} \qquad \text{for } \pi : M_g \to M_g \text{ in } \mathcal{L}$$



COMPOSITIONAL STRUCTURE OF STREAMS

THEOREM

Effectful streams form an effectful category Stream.

- composition and monoidal actions are defined coinductively: for $f : N_g \cdot A \to B$ and $g : N_g \cdot B \to C$,

$$\begin{cases} (f \fatsemi_N g)^\circ := \\ (f \fatsemi_N g)^+ := f^+ \fatsemi_M g^+ \end{cases}$$

$$\begin{cases} (X \otimes_N f)^\circ := \\ (X \otimes_N f)^+ := X^+ \otimes_M f^+ \end{cases}$$

SEMANTICS FOR THE STREAM CIPHER PROTOCOL

Fix two finite sets Char ⇝ messages
Seed ⇝ seeds

Pure computations are functions
Local computations are stochastic functions
Effectful computations read from and write to a global state

(Set, Stoch, SeedStoch)

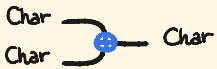
SeedStoch is the Kleisli category of a promonad
that adds a global state Seed ⊗ Seed :

SeedStoch(A, B) := Stoch(Seed ⊗ Seed ⊗ A, Seed ⊗ Seed ⊗ B)

SEMANTICS FOR THE STREAM CIPHER PROTOCOL

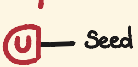
Semantics for pure and local computations.

$[\![- \oplus -]\!] := \text{XOR} : \text{Char} \otimes \text{Char} \to \text{Char}$ in Set



⇝ bitwise XOR

$[\![\text{rand}]\!] := \text{unif} : 1 \to \mathcal{D}(\text{Seed})$ in Stoch



⇝ uniform distribution

$[\![\text{prng}]\!] : \text{Seed} \to \text{Seed} \times \text{Char}$ in Stoch

⇝ use the seed to generate a key

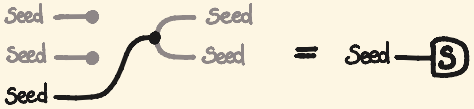
SEMANTICS FOR THE STREAM CIPHER PROTOCOL

Semantics for effectful computations.

$[\![\text{setSeed}]\!] : \text{Seed}^3 \to \text{Seed}^2$

$\text{Seed} \rightsquigarrow 1$

$\rightsquigarrow$ copy the seed to the global state

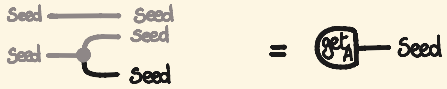
$[\![\text{setSeed}]\!] :=$ 

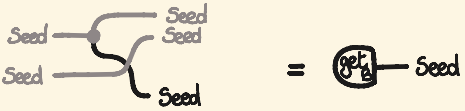
in SeedStoch

$[\![\text{getSeedA}]\!], [\![\text{getSeedB}]\!] : \text{Seed}^2 \to \text{Seed}^3$

$1 \rightsquigarrow \text{Seed}$

$\rightsquigarrow$ alice and bob get their seeds

$[\![\text{getSeedA}]\!] :=$ 

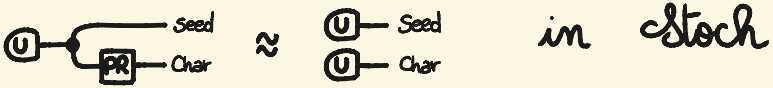
$[\![\text{getSeedB}]\!] :=$ 

ON PSEUDORANDOM NUMBERS

There are no perfect pseudorandom number generators in Stoch.

ASSUMPTION

There is an approximate pseudorandom number generator,
$\mathrm{PR} : \mathrm{Seed} \to \mathcal{D}(\mathrm{Char})$ such that :



in Stoch

[Broadbent & Karvonen 2023]

SEMANTICS FOR THE STREAM CIPHER PROTOCOL

- seedGen = (SEED) : $\mathbb{I} \to \mathbb{I}$ in Stream

$seedGen^{\circ}$:=

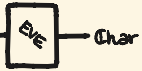
=

$seedGen^{+\circ}$:=

=

$seedGen^{++} = seedGen^{+}$

- eve =



: Char → Char in Stream

eve° :=

=

$eve^{+} = eve$

SEMANTICS FOR THE STREAM CIPHER PROTOCOL

- alice = 

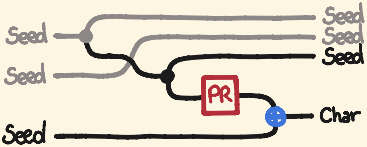
: Char → Char in Stream

- bob = 

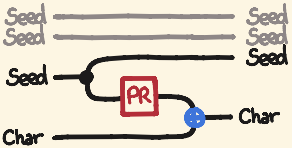
: Char → Char in Stream

alice° := 

= 

bob° := 

= 

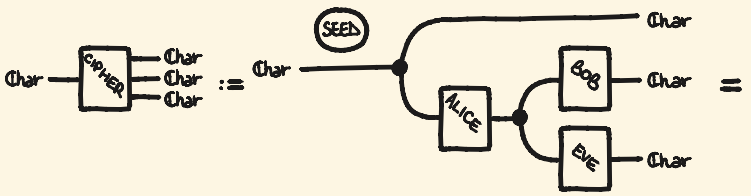
alice⁺° = bob⁺° := 

= 

alice⁺⁺ = alice⁺

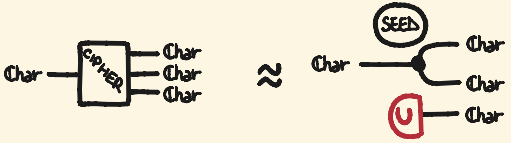
bob⁺⁺ = bob⁺

SEMANTICS FOR THE STREAM CIPHER PROTOCOL




```
cipher(m) = do
  seedGen() ⇝ ()
  alice(m) ⇝ n
  bob(n) ⇝ m'
  eve(n) → n'
  return (m, m', n')
```

SECURITY OF THE PROTOCOL



Alice and Bob see the same message

Eve sees noise

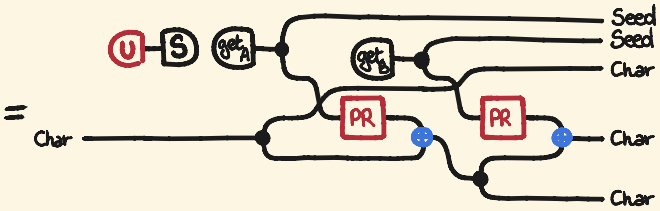
STREAM CIPHER IS SECURE

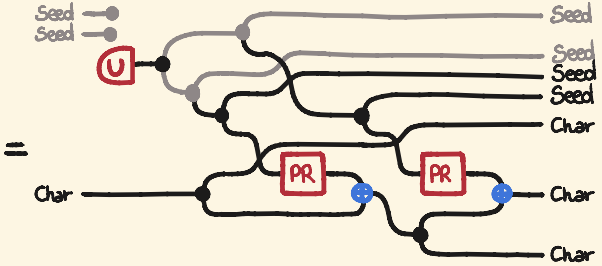
Proceed by coinduction.

cipher°

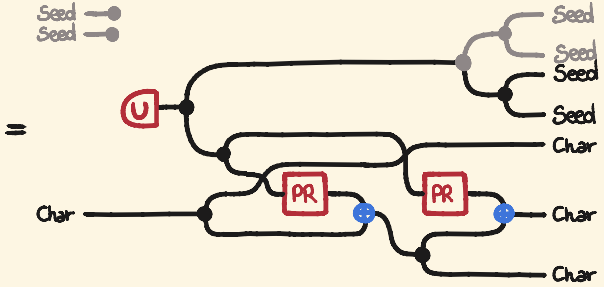
$$
\begin{aligned}
&\mathsf{cipher}(m)^{\circ} = \mathbf{do} \\
= \quad &\mathsf{rand}() \to s \\
&\mathsf{setSeed}(s) \rightsquigarrow () \\
&\mathsf{getSeedA}() \rightsquigarrow s_A \\
&\mathsf{prng}(s_A) \to (s'_A, k_A) \\
&\mathsf{getSeedB}() \rightsquigarrow s_B \\
&\mathsf{prng}(s_B) \to (s'_B, k_B) \\
&\mathsf{return}\ (m, s'_A, m \oplus k_A \oplus k_B, s'_B, m \oplus k_A)
\end{aligned}
$$

STREAM CIPHER IS SECURE



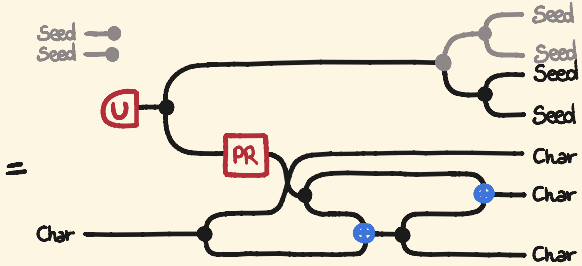


STREAM CIPHER IS SECURE



=

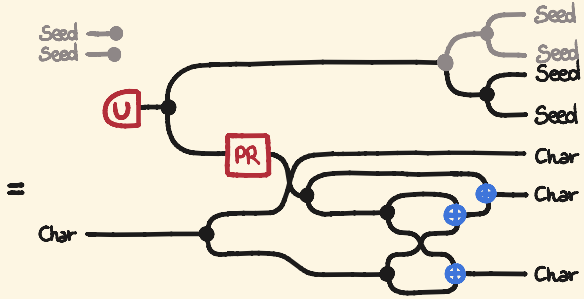
by associativity of copy



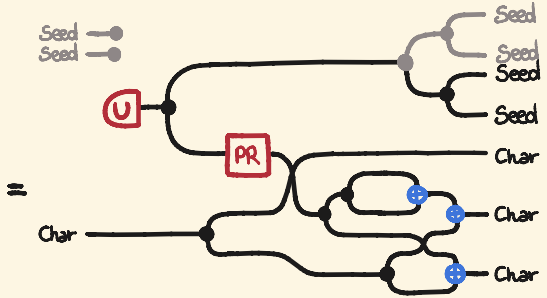
=

pseudorandom is deterministic

STREAM CIPHER IS SECURE

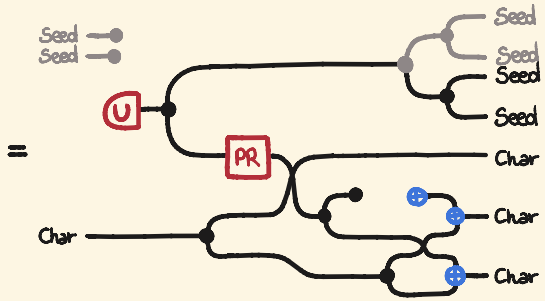


XOR is deterministic

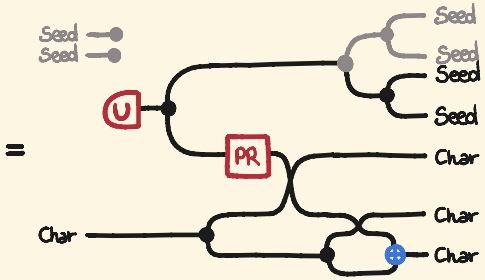


by associativity of copy and XOR

STREAM CIPHER IS SECURE

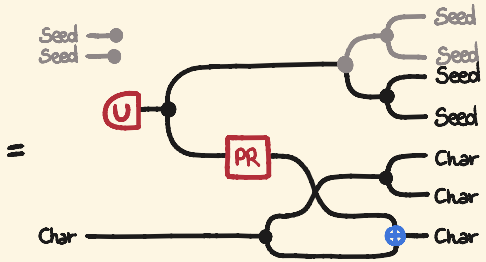


xor is nihilpotent

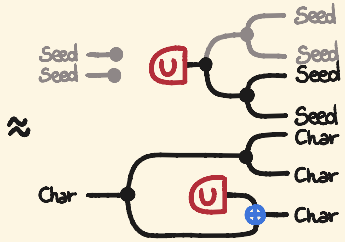


by unitality of copy and xor

STREAM CIPHER IS SECURE

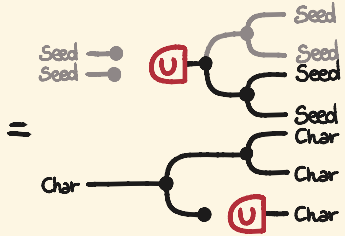


by associativity of copy

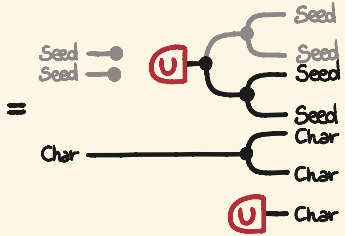


by assumption

STREAM CIPHER IS SECURE



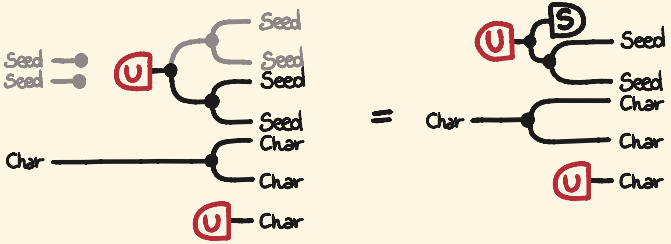
the uniform distribution is a Sweedler integral for xor



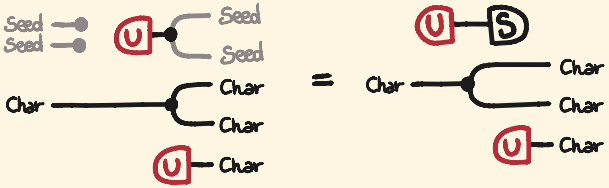
by unitality of copy

STREAM CIPHER IS SECURE

cipher° ≈



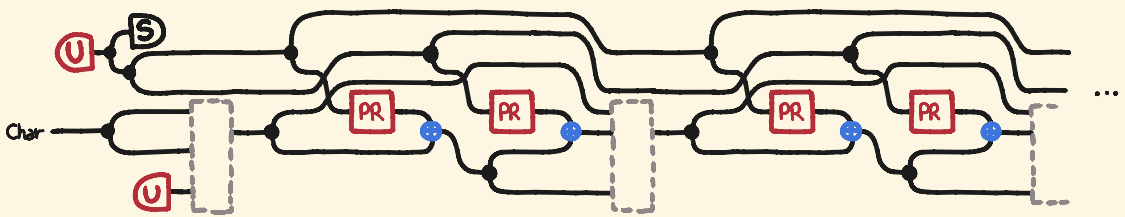
secure° :=



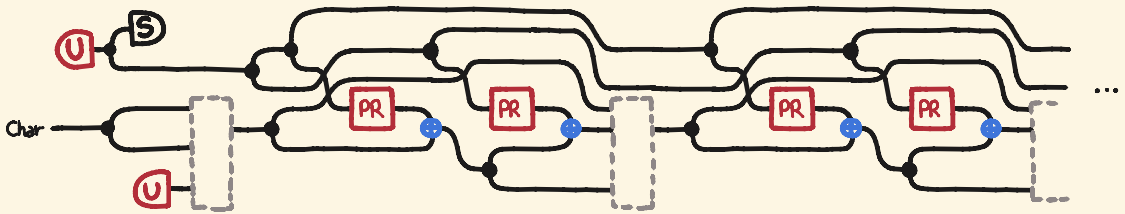
STREAM CIPHER IS SECURE

cipher

=

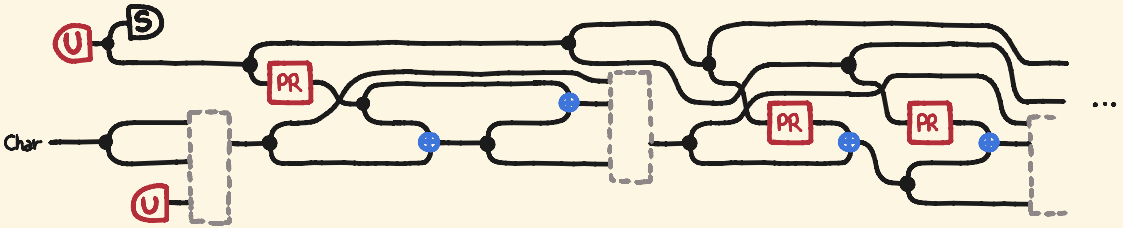


= (by sliding)

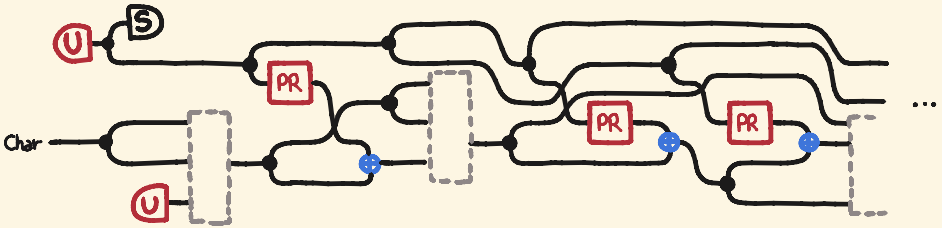


STREAM CIPHER IS SECURE

= (pseudorandom is deterministic)

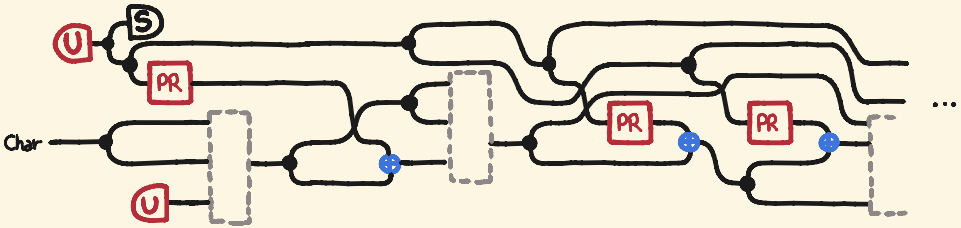


= (XOR is deterministic and nihilpotent)

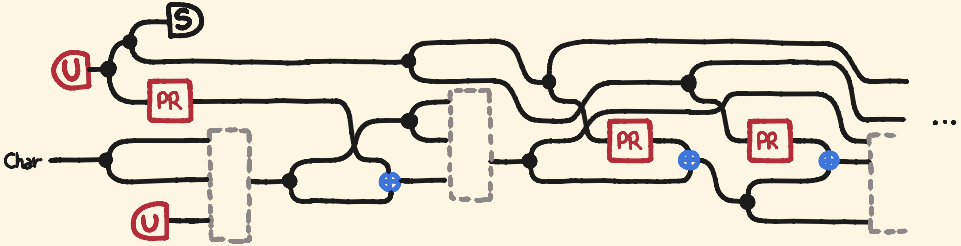


STREAM CIPHER IS SECURE

= (by sliding)

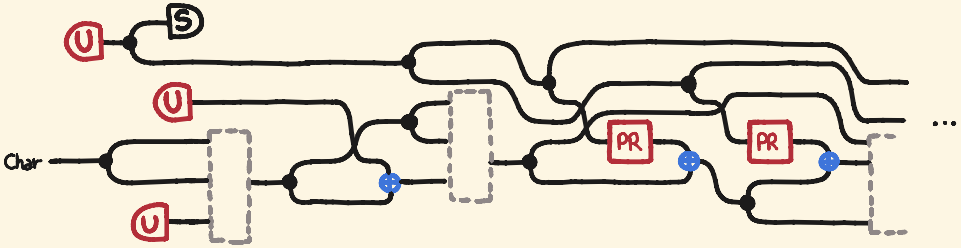


= (by associativity)

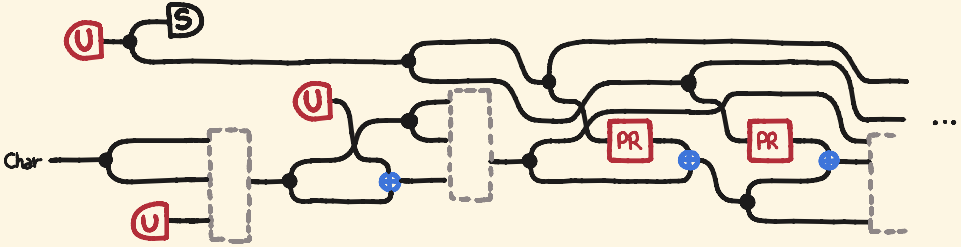


STREAM CIPHER IS SECURE

≈ (by assumption on pseudorandom)

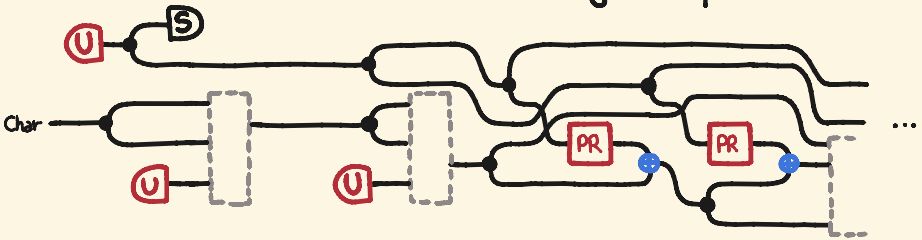


= (by sliding)

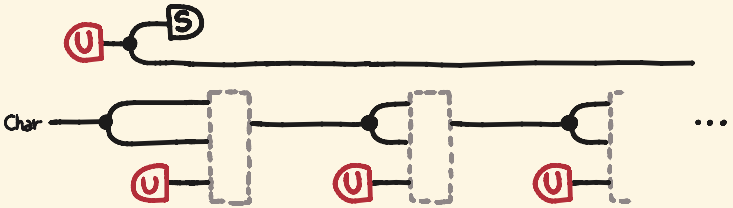


STREAM CIPHER IS SECURE

= (unif is a Sweedler integral for xor)

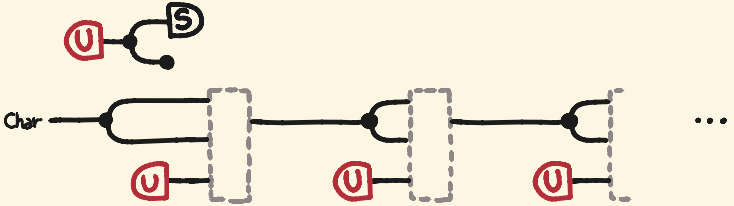


≈ (by coinduction)

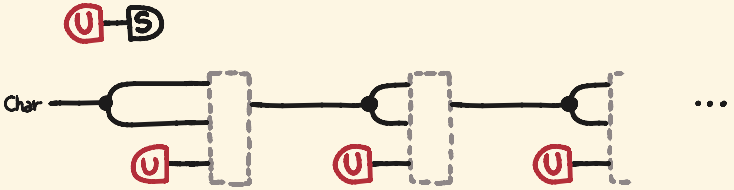


STREAM CIPHER IS SECURE

= (by coinduction)



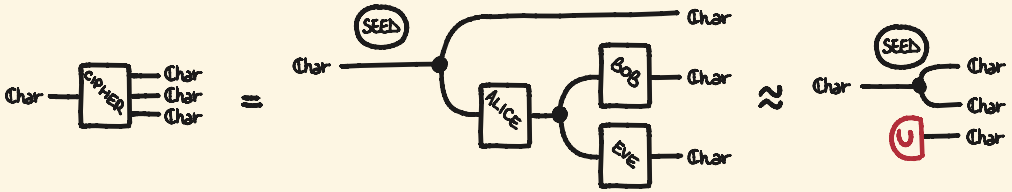
= (by unitality)



= secure

STREAM CIPHER IS SECURE

We have shown



using sliding and coinduction.

OUTLINE

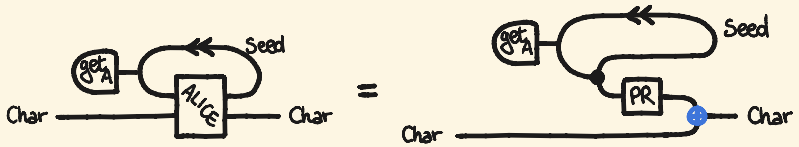
- effectful categories
- effectful streams
- effectful trace semantics
- causal processes

SYSTEMS THAT KEEP OPERATING

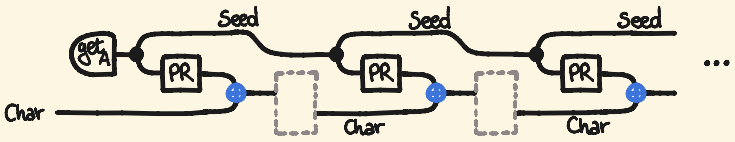
```
alice(m)° = do
  getSeedA() ⇝ (s)
  prng(s) → (s', k)
  return (s', m ⊕ k)
```


```
alice(s, m)+° = do
  prng(s) → (s', k)
  return (s', m ⊕ k)
```

$$\text{alice}(s,m)^{++} = \text{alice}(s,m)^{+}$$



unrolling ⇝



EFFECTFUL MEALY MACHINES

A Mealy machine $(f, S, s_0) : A \to B$ in $\mathcal{C}$ with state space S, inputs A and outputs B is a morphism

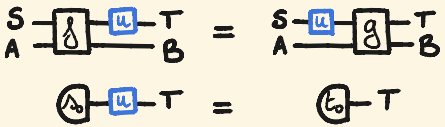
$$f : S \otimes A \to S \otimes B$$

with an initial state

$$s_0 : I \to S$$

A morphism of Mealy machines $u : (f, S, s_0) \to (g, T, t_0)$ is a pure morphism $u : S \to T$ in $\mathcal{P}$

such that

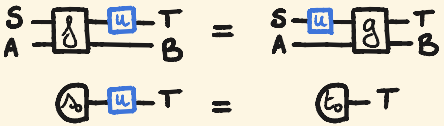


[cf. Katis, Sabadini, Walters 1997]

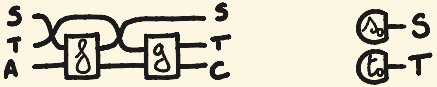
EFFECTFUL CATEGORY OF MEALY MACHINES

Mealy is an effectful category where

- objects are the objects of $\mathcal{C}$
- morphisms $(f, S, s_0) : A \to B$ are Mealy machines quotiented by pure isomorphisms $u : S \xrightarrow{\cong} T$



- composition tensors the state spaces



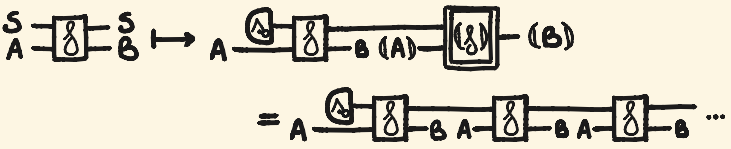
COMPOSITIONAL TRACE SEMANTICS

THEOREM

There is an effectful functor

$$\mathrm{Tr} : \mathbf{Mealy} \to \mathbf{Stream}$$

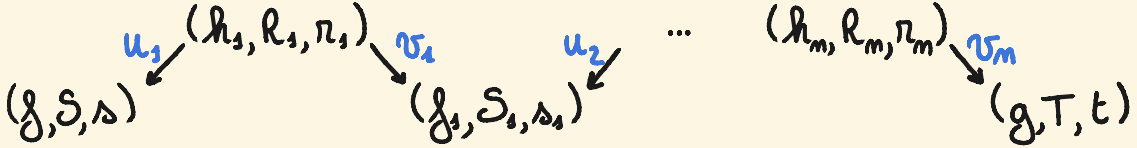
$$A \mapsto (\!|A|\!) = (A, A, \dots)$$



$\rightsquigarrow$ in **Rel** these traces coincide with the classical traces

BISIMULATION

Two effectful Mealy machines $(f, S, s), (g, T, t) : A \to B$ are bisimilar if they belong to the same connected component in $\mathrm{Mealy}(A, B)$:



COALGEBRAIC BISIMULATION

PROPOSITION

When $\mathcal{C} = Kl(M)$, for a commutative monad preserving weak pullbacks, then (f,S,s) and (g,T,t) are bisimilar iff they have the same bisimulation quotient, i.e. there is (h,Q,q) with morphisms

$$(f,S,s) \xrightarrow{u} (h,Q,q) \xleftarrow{v} (g,T,t) \ .$$

EXAMPLES

- Set
- Rel
- pStoch

BISIMULATION ⇒ TRACE EQUIVALENCE

THEOREM

If two effectful Mealy machines are bisimilar, then they are trace equivalent.

PROOF SKETCH

$u : (f, S, s_0) \to (g, T, t_0)$

$$\begin{aligned} \Rightarrow (\mathrm{tr}(g,T,t_0))^\circ &= (t_0 \otimes \mathbb{1}) ; g \\ &= ((s_0 ; u) \otimes \mathbb{1}) ; g \\ &= (s_0 \otimes \mathbb{1}) ; f ; (u \otimes \mathbb{1}) \\ &= (\mathrm{tr}(f,S,s_0))^\circ ; (u \otimes \mathbb{1}) \end{aligned}$$

by coinduction, $u \cdot (\mathrm{tr}(g,T,t_0))^+ \sim (\mathrm{tr}(f,S,s_0))^+$

□

OUTLINE

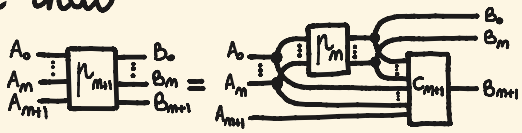
- effectful categories
- effectful streams
- effectful trace semantics
- causal processes

CAUSAL PROCESSES

A causal process $p : A \to B$ in a copy-discard category $\mathcal{C}$ is a family of morphisms

$$p_n : A_0 \otimes \cdots \otimes A_n \longrightarrow B_0 \otimes \cdots \otimes B_n$$

such that



for some $c_{n+1} : B_0 \otimes \cdots \otimes B_n \otimes A_0 \otimes \cdots \otimes A_n \otimes A_{n+1} \to B_{n+1}$

THEOREM

causal processes form a monoidal category $\mathbf{Proc}$ when $\mathcal{C}$ has quasi-total conditionals.

[cf. Raney 1958 ; Sprunger & Katsumata 2019]

CAUSAL PROCESSES ARE STREAMS

THEOREM

Consider $(\mathsf{fun}\mathcal{C}, \mathsf{tot}\mathcal{C}, \mathcal{C})$.

If $\mathcal{C}$ has quasi-total conditionals and ranges,

$$\mathsf{Proc} \simeq \mathsf{Stream} \ .$$

[cf. Coecke & Spekkens 2012; Cho & Jacobs 2019; Fritz 2020; Corradini & Gadducci 1999; Fritz, Gadducci, Perrone, Trotta 2023]

EXAMPLES

- Set
- Rel
- pStoch
- Par
- Stoch

TRACES ARE EFFECTFUL TRACES

Compute the traces of a Mealy machine

$$(f, S, s) : A \to B$$

in some known cases.

$(b_0, \dots, b_n)$ is a trace of $(a_0, \dots, a_n)$

Set if $s_0 = s$ and $\forall k \leq n \ (s_{k+1}, b_k) = f(s_k, a_k)$

Rel if $\exists (s_0, \dots, s_{n+1}) \quad s_0 \in s$
and $\forall k \leq n \ (s_{k+1}, b_k) \in f(s_k, a_k)$

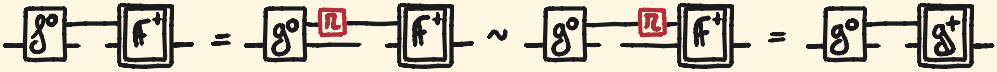
pStoch with probability $\sum_{(s_0, \dots, s_{n+1})} s(s_0 \mid *) \cdot \prod_{k \leq n} f(s_{k+1}, b_k \mid s_k, a_k)$

SUMMARY

- formal compositional semantics for effectful stream computations
- trace equivalence and bisimulation of effectful Mealy machines
- characterisation as causal stream processes

FUTURE WORK

- coinduction up-to dinaturality



- Rel with explicit failure
- equality in St-C implies bisimulation

- distance instead of equivalence relation for security