Streams are infinite sequences of values of a type A

$$\mathbb{S}_A \ni a_0 \lhd a_1 \lhd a_2 \lhd a_3 \lhd \cdots$$

A function from streams to a (discrete) type B

$$f : \mathbb{S}_A \longrightarrow B$$

is continuous if it depends only on a finite prefix for each input:

$$\forall \alpha, \exists n, \forall \alpha', \alpha' =_n \alpha \Rightarrow f\alpha' = f\alpha$$

$\alpha' =_n \alpha$ means the first n elements are the same

n is called modulus of continuity of f for α

Brouwer's Continuity Principle:

All functions

$$f : \mathbb{S}_{\mathbb{N}} \longrightarrow \mathbb{N}$$

are continuous

Computational Justification:

If f is a computable function/program it must always terminate in finite time;
in that time it can only read a finite number of input elements.

Can we add it to Type Theory?

NO (M. Escardó)

Brouwer's Continuity Principle is inconsistent in Type Theory.

(We can use the principle to define an evil function that proves $0=1$)

Can we fix it? Reformulate in a safe form?

- Escardó: use a weak existential

  $\underline{\exists n}$ I can state the existence but not extract the modulus

- I propose a different way:

  different notion of stream

  monadic streams
  the elements of the sequence are generated by monadic actions/side effects.

A monad is a type operator

$$M : \mathrm{Set} \longrightarrow \mathrm{Set}$$

with operations

$$\mathrm{return}_X : X \longrightarrow MX$$

$$\mathrm{mult}_X : M(MX) \longrightarrow MX$$

satisfying some properties

Idea: MX is a way of producing elements of X, involving some side effects, possibly failing or producing several elements

A monad is a type operator

$$M : \mathrm{Set} \longrightarrow \mathrm{Set}$$

with operations

$$\mathrm{return}_X : X \longrightarrow MX$$

$$\mathrm{mult}_X : M(MX) \longrightarrow MX$$

satisfying some properties

Idea: MX is a way of producing elements of X, involving some side effects, possibly failing or producing several elements

Examples:

- The Maybe monad

  Maybe X has elements

  Just x for $x : X$

  Nothing

-

A monad is a type operator

$$M : \mathrm{Set} \longrightarrow \mathrm{Set}$$

with operations

$$\mathrm{return}_X : X \longrightarrow MX$$

$$\mathrm{mult}_X : M(MX) \longrightarrow MX$$

satisfying some properties

Idea: MX is a way of producing elements of X, involving some side effects, possibly failing or producing several elements

Examples:

- The state monad (with set of states S)

$$\mathrm{State}\, X = S \longrightarrow X \times S$$

an element of $\mathrm{State}\, X$ takes the present state $s : S$, uses it to produce an $x : X$ an the updated state $s' : S$

$$\mathrm{return}\ x = \lambda s.\ \langle x, s \rangle$$

$$\mathrm{mult} = \ldots$$

A monad is a type operator

$$M : \mathrm{Set} \longrightarrow \mathrm{Set}$$

with operations

$$\mathrm{return}_X : X \longrightarrow MX$$

$$\mathrm{mult}_X : M(MX) \longrightarrow MX$$

satisfying some properties

Idea: MX is a way of producing elements of X, involving some side effects, possibly failing or producing several elements

Examples:

- The writer monad (state that can only be written)

$\langle I, e, * \rangle$ monoid

$$\mathrm{Writer}_I X = X \times I$$

Idea: initial state e, every action $\langle x, i \rangle : \mathrm{Writer}_I Y$ multiplies the state by i

A monad is a type operator

$$M : \mathrm{Set} \longrightarrow \mathrm{Set}$$

with operations

$$\mathrm{return}_X : X \longrightarrow MX$$

$$\mathrm{mult}_X : M(MX) \longrightarrow MX$$

satisfying some properties

Idea: MX is a way of producing elements of X, involving some side effects, possibly failing or producing several elements

Examples:

- The writer monad (state that can only be written)

  $\langle I, e, * \rangle$ monoid

  $$\mathrm{Writer}_I X = X \times I$$

  Idea: initial state e,
  every action $\langle x, i \rangle : \mathrm{Writer}_I X$
  multiplies the state by i

- Haskell IO monad

  $\mathrm{IO}\, X =$ interactive input/output actions that eventually produce an X

Streams

CoInductive $\$_A : \mathrm{Set}$

cons: $A \times \$_A \to \$_A$

Monadic Streams

CoInductive $\mathcal{S}_{M,A} : \mathrm{Set}$

cons: $M(A \times \mathcal{S}_{M,A}) \to \mathcal{S}_{M,A}$

Monadic Streams

CoInductive $\$_{M,A}$: Set

cons : $M(A \times \$_{M,A}) \to \$_{M,A}$

Examples

- $\$_{Id,A}$ pure streams

Monadic Streams

CoInductive $\$_{M,A}$: Set

cons : $M(A \times \$_{M,A}) \to \$_{M,A}$

Examples

- $\$_{Id,A}$ pure streams
- $\$_{Maybe,A}$ colists
  finite and infinite sequences
  empty list (cons Nothing)

Monadic Streams

Coinductive $\$_{M,A} : Set$

mcons : $M(A \times \$_{M,A}) \to \$_{M,A}$

Examples

- $\$_{Id,A}$ pure streams

- $\$_{Maybe,A}$ colists
  finite and infinite sequences
  empty list (cons Nothing)

- $\$_{State_S,A}$
  state dependent/changing
  infinite sequences

$\alpha : \$_{State_S,A}$ $\quad \alpha = mcons\ h$

$h : S \to A \times \$_{..} \times S$

Monadic Streams

CoInductive $\$_{M,A} : Set$

mcons : $M(A \times \$_{M,A}) \to \$_{M,A}$

Examples

- $\$_{Id,A}$ pure streams

- $\$_{Maybe,A}$ colists
  finite and infinite sequences
  empty list (cons Nothing)

- $\$_{State_S,A}$
  state dependent/changing
  infinite sequences

$\alpha : \$_{State_S,A}$ $\quad \alpha = mcons\ h$

$h : S \to A \times \$_{..} \times S$

α

s_0

Monadic Streams

Coinductive $\$_{M,A} : \mathrm{Set}$

$\mathrm{mcons} : M(A \times \$_{M,A}) \to \$_{M,A}$

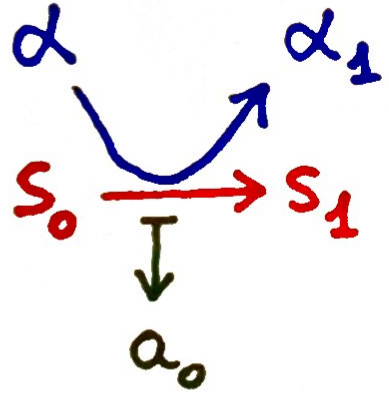
Examples

- $\$_{\mathrm{Id},A}$ pure streams
- $\$_{\mathrm{Maybe},A}$ colists
  finite and infinite sequences
  empty list (cons Nothing)

- $\$_{\mathrm{State}_S,A}$
  state dependent/changing
  infinite sequences

$\alpha : \$_{\mathrm{State}_S,A}$ $\quad \alpha = \mathrm{mcons}\ h$

$h : S \to A \times \$_{\ldots} \times S$



Monadic Streams

CoInductive $\mathfrak{S}_{M,A} : \mathrm{Set}$

$\mathrm{mcons} : M(A \times \mathfrak{S}_{M,A}) \to \mathfrak{S}_{M,A}$

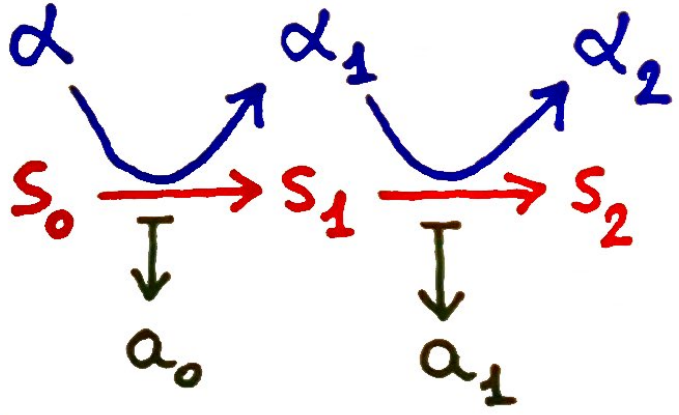
Examples

- $\mathfrak{S}_{\mathrm{Id},A}$ pure streams
- $\mathfrak{S}_{\mathrm{Maybe},A}$ colists
  finite and infinite sequences
  empty list (cons Nothing)

- $\mathfrak{S}_{\mathrm{State}_S,A}$
  state dependent/changing
  infinite sequences

$\alpha : \mathfrak{S}_{\mathrm{State}_S,A}$ $\quad \alpha = \mathrm{mcons}\ h$

$h : S \to A \times \mathfrak{S}_{\ldots} \times S$



Monadic Streams

Coinductive $\$_{M,A} : \mathrm{Set}$

$\mathrm{mcons} : M(A \times \$_{M,A}) \to \$_{M,A}$

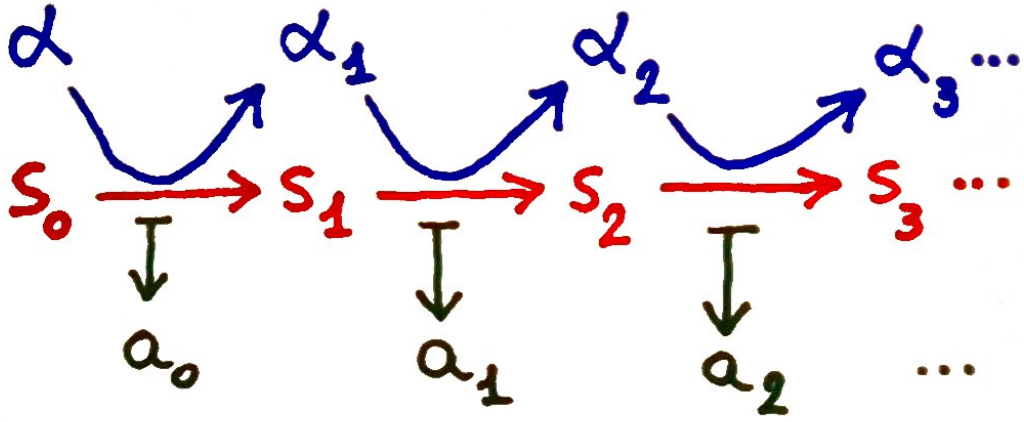
Examples

- $\$_{\mathrm{Id},A}$ pure streams
- $\$_{\mathrm{Maybe},A}$ colists
  finite and infinite sequences
  empty list (cons Nothing)

- $\$_{\mathrm{State}_S,A}$
  state dependent/changing
  infinite sequences

$\alpha : \$_{\mathrm{State}_S,A}$ $\quad \alpha = \mathrm{mcons}\ h$

$h : S \to A \times \$_{\_} \times S$



Monadic Streams

CoInductive $\mathfrak{S}_{M,A} : \mathrm{Set}$

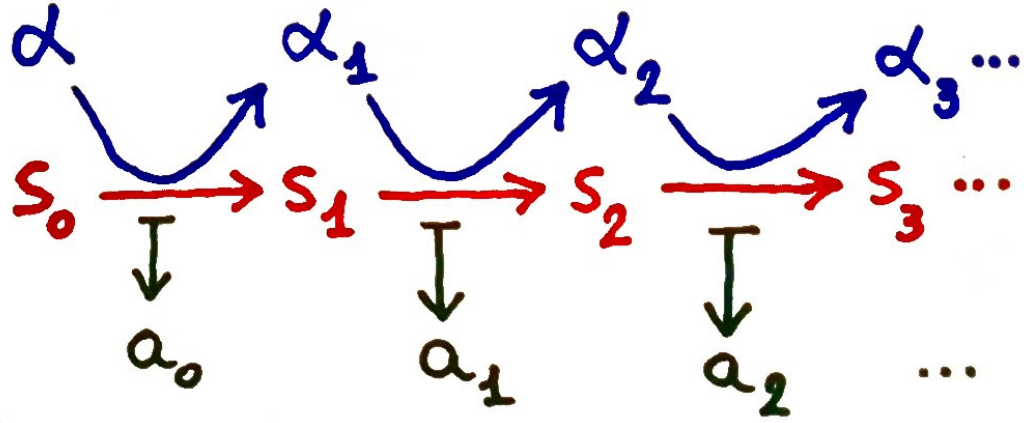
mcons : $M(A \times \mathfrak{S}_{M,A}) \to \mathfrak{S}_{M,A}$

Examples

- $\mathfrak{S}_{Id,A}$ pure streams
- $\mathfrak{S}_{Maybe,A}$ colists
  finite and infinite sequences
  empty list (cons Nothing)
- $\mathfrak{S}_{State_S,A}$
  state dependent/changing
  infinite sequences

$\alpha : \mathfrak{S}_{State_S,A}$ $\quad \alpha = \mathrm{mcons}\ h$

$h : S \to A \times \mathfrak{S}_{\cdots} \times S$



- $\mathfrak{S}_{List,A}$ finitely branching trees

Functions on Monadic Streams

$$f : \forall M, \mathcal{S}_{M,A} \longrightarrow MB$$

f must be "uniform":

compute the same operations independently of M,

it works independently of how the streams are generated.

Functions on Monadic Streams

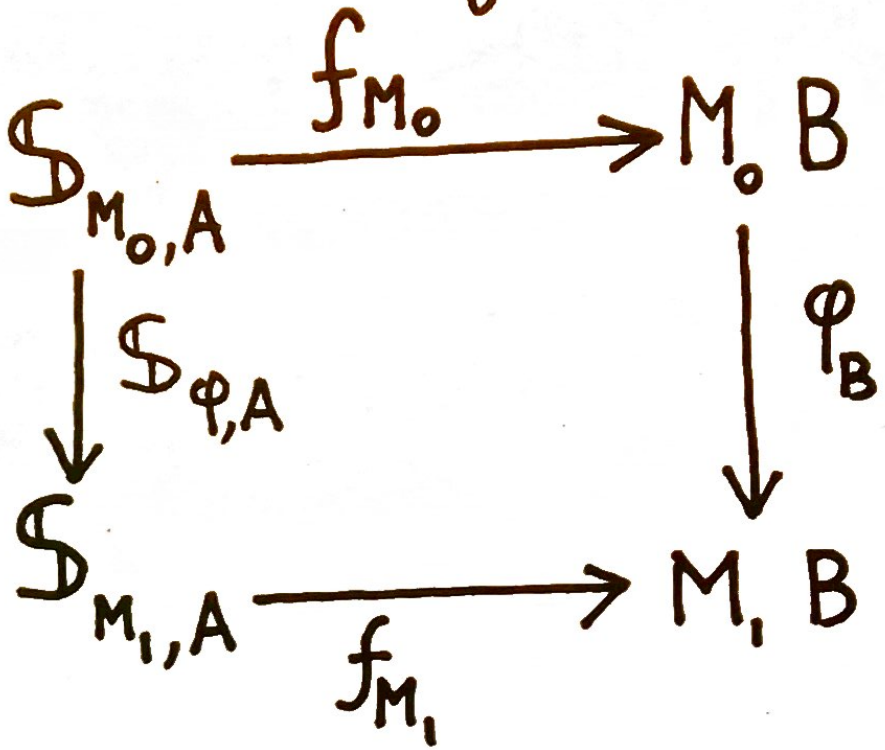
$$f : \forall M, \mathcal{S}_{M,A} \longrightarrow M\,B$$

f must be "uniform":
compute the same operations independently of M,
it works independently of how the streams are generated.

Formally: f is natural in M

if $\varphi : M_0 \dot{\longrightarrow} M_1$
is a morphism of monads,

then this diagram commutes



$\mathcal{S}_{\varphi,A}$ lifting of φ to streams
(apply φ to all monadic actions in the stream)

Test for (syntactic) constants

$$f : \forall M, \mathcal{S}_{M,A} \longrightarrow MB$$

Test for (syntactic) constants

$$f : \forall M, \mathcal{S}_{M,A} \longrightarrow MB$$

$$\forall M, \forall \alpha, \; f_M \, \alpha = \text{return } b$$

Test for (syntactic) constants

$$f : \forall M, \mathcal{S}_{M,A} \longrightarrow M\,B$$

$$f_{\text{Maybe}}\ (\text{incons Nothing}) = \text{Just } b$$

iff

$$\forall M, \forall \alpha,\ f_M\ \alpha = \text{return } b$$

Test for (syntactic) constants

$$f : \forall M,\ \mathbb{S}_{M,A} \longrightarrow M\,B$$

Theorem

$$f_{\text{Maybe}}\ (\text{incons Nothing}) = \text{Just } b$$

iff

$$\forall M, \forall \alpha,\ f_M\ \alpha = \text{return } b$$

Test for (syntactic) constants

$$f : \forall M, \mathcal{S}_{M,A} \longrightarrow M\,B$$

Theorem

$$f_{Maybe}\ (incons\ Nothing) = Just\ b$$

iff

$$\forall M, \forall \alpha,\ f_M\ \alpha = return\ b$$

Theorem

f_{Id} is continuous

Test for (syntactic) constants

$$f : \forall M, \mathbb{S}_{M,A} \longrightarrow M\,B$$

Theorem

$$f_{\text{Maybe}}\ (\text{incons Nothing}) = \text{Just } b$$

iff

$$\forall M, \forall \alpha,\ f_M\,\alpha = \text{return } b$$

Theorem

f_{Id} is continuous

Computation of the modulus of continuity

Use writer monad with monoid $\langle \mathbb{N}, 0, \max \rangle$

$$\alpha = a_0 \lhd a_1 \lhd a_2 \lhd \cdots : \mathbb{S}_A$$

Test for (syntactic) constants

$$f : \forall M, \mathbb{S}_{M,A} \longrightarrow MB$$

Theorem

$f_{\text{Maybe}}\,(\text{incons Nothing}) = \text{Just } b$

iff

$\forall M, \forall \alpha,\ f_M\,\alpha = \text{return } b$

Theorem

f_{Id} is continuous

Computation of the modulus of continuity

Use writer monad with monoid $\langle \mathbb{N}, 0, \max \rangle$

$\alpha = a_0 \lhd a_1 \lhd a_2 \lhd \cdots : \mathbb{S}_A$

$\alpha_\iota = \langle a_0, 0 \rangle \lhd \langle a_1, 1 \rangle \lhd \cdots : \mathbb{S}_{Wr_{\mathbb{N}}, A}$

Test for (syntactic) constants

$$f : \forall M, \mathcal{S}_{M,A} \longrightarrow M\,B$$

Theorem

$$f_{\text{Maybe}}\ (\text{incons Nothing}) = \text{Just } b$$

iff

$$\forall M, \forall \alpha,\ f_M\ \alpha = \text{return } b$$

Theorem

f_{Id} is continuous

Computation of the modulus of continuity

Use writer monad with monoid $\langle \mathbb{N}, 0, \max \rangle$

$$\alpha = a_0 \lhd a_1 \lhd a_2 \lhd \cdots : \mathcal{S}_A$$

$$\alpha_i = \langle a_0, 0 \rangle \lhd \langle a_1, 1 \rangle \lhd \cdots : \mathcal{S}_{Wr_{\mathbb{N}}, A}$$

$$f_{Writer_{\mathbb{N}}, A}\ \alpha_i = \langle b, n \rangle$$

Test for (syntactic) constants

$$f : \forall M, \mathfrak{S}_{M,A} \longrightarrow MB$$

Theorem

$f_{Maybe}\ (incons\ Nothing) = Just\ b$

iff

$\forall M, \forall \alpha,\ f_M\ \alpha = return\ b$

Theorem

f_{Id} is continuous

Computation of the modulus of continuity

Use writer monad with monoid $\langle \mathbb{N}, 0, max \rangle$

$\alpha = a_0 \lhd a_1 \lhd a_2 \lhd \cdots : \mathfrak{S}_A$

$\alpha_i = \langle a_0, 0 \rangle \lhd \langle a_1, 1 \rangle \lhd \cdots : \mathfrak{S}_{Writer_{\mathbb{N}}, A}$

$f_{Writer_{\mathbb{N}}, A}\ \alpha_i = \langle b, n \rangle$

n: modulus

Significance

- A case study for coinduction
  - Polymorphic definition of $\$_{M,A}$
  - Only for strictly positive monads (containers)
  - Need reflection on coinductive types
  - Good programming/reasoning for codata
  - Naturality condition technically tricky
- Important Subject
  $\$_{M,A}$ used for Functional Reactive Programs

Significance

- A case study for coinduction
  - Polymorphic definition of $\$_{M,A}$
  - Only for strictly positive monads (containers)
  - Need reflection on coinductive types
  - Good programming/reasoning for codata
  - Naturality condition technically tricky
- Important Subject
  $\$_{M,A}$ used for Functional Reactive Programs
- Real Numbers Computing ?
  (Brouwer's original application)

Significance

- Foundational Issue

Significance

- Foundational Issue

Disagreement on Foundations

Significance

- Foundational Issue

Disagreement on Foundations

NOT about logic rules
(strong vs weak $\exists$)

Significance

- **Foundational Issue**

Disagreement on Foundations

NOT about logic rules
(strong vs weak $\exists$)

BUT about the nature
of mathematical reality
(what is an infinite sequence)

Significance

- **Foundational Issue**

Disagreement on Foundations

NOT about logic rules
(strong vs weak $\exists$)

BUT about the nature
of mathematical reality
(what is an infinite sequence)

THANK You