Lecture two:

A Coinductive Calculus of Streams

Jan Rutten

CWI Amsterdam & Radboud University Nijmegen

IPM, Tehran - 13 January 2016

▲□▶ ▲□▶ ▲□▶ ▲□▶ = 三 のへで

Overview of this talk

▲□▶▲□▶▲□▶▲□▶ □ のQ@

- 1. Stream differential equations (SDEs)
- 2. Solving systems of SDEs
- 3. Formats for SDEs
- 4. Streams and coinduction
- 5. Discussion

1. Stream differential equations

Streams are the canonical example of a (final) coalgebra.

Stream differential equations:

- General framework for defining streams.
- Hand in hand with **coinduction** as main proof method.
- Ultimately leading to efficient algorithmics and automated proofs.

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

1. Stream differential equations

Streams are the canonical example of a (final) coalgebra.

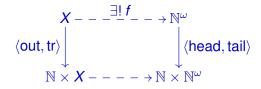
Stream differential equations:

- General framework for defining streams.
- Hand in hand with **coinduction** as main proof method.
- Ultimately leading to efficient algorithmics and automated proofs.

(日) (日) (日) (日) (日) (日) (日)

Stream Differential Equations (SDEs)

We shall explain how the following diagram



(ロ) (同) (三) (三) (三) (○) (○)

represents a system of stream differential equations

and its solution.

A stream system/coalgebra

$$\langle \mathsf{out},\mathsf{tr} \rangle \Big|_{\mathbb{N} \times X}$$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

For $x \in X$, one often writes

 $(\operatorname{out}(x) = n \text{ and } \operatorname{tr}(x) = y) \equiv x \xrightarrow{n} y$

(dynamical/transition system)



Another way of writing:

 $(\operatorname{out}(x) = n \text{ and } \operatorname{tr}(x) = y) \equiv (x(0) = n \text{ and } x' = y)$

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

initial value and derivative!

So we view any stream coalgebra

$$\mathsf{out},\mathsf{tr}
angle \downarrow \ \mathbb{N}\times \mathcal{Y}$$

as a system of stream differential equations (SDEs):

 $\{x(0) = out(x) \text{ and } x' = tr(x)\}_{x \in X}$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

We think of X as the set of variables.

So we view any stream coalgebra

$$\mathsf{out},\mathsf{tr}\rangle \Big|_{\mathbb{N}\times\mathcal{Y}}$$

as a system of stream differential equations (SDEs):

 $\{x(0) = out(x) \text{ and } x' = tr(x)\}_{x \in X}$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

We think of *X* as the set of **variables**.

Streams

$$\mathbb{N}^{\omega} \\ \downarrow \langle \mathsf{head}, \mathsf{tail} \rangle \\ \mathbb{N} \times \mathbb{N}^{\omega}$$

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● のへぐ

 $head(n_0, n_1, n_2, ...) = n_0$

 $tail(n_0, n_1, n_2, \ldots) = (n_1, n_2, \ldots)$

$$\begin{array}{c} \mathbb{N}^{\omega} \\ \downarrow \langle \mathsf{head}, \mathsf{tail} \rangle \\ \mathbb{N} \times \mathbb{N}^{\omega} \end{array}$$

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● のへぐ

Also here we shall write

 $(n_0, n_1, n_2, \ldots)(0) = n_0$

$$(n_0, n_1, n_2, \ldots)' = (n_1, n_2, n_3, \ldots)$$

Finality of streams

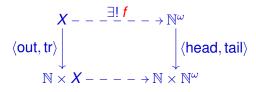
$$\begin{array}{c|c} X - - - \exists f \\ & \langle \mathsf{out}, \mathsf{tr} \rangle \\ & & \downarrow \\ & \mathbb{N} \times X - - - \to \mathbb{N} \times \mathbb{N}^{\omega} \end{array}$$

The function *f*, defined by

 $f(x) = (\operatorname{out}(x), \operatorname{out}(\operatorname{tr}(x)), \operatorname{out}(\operatorname{tr}(\operatorname{tr}(x))), \ldots)$

is the unique function making the diagram commute.

Solutions by finality



System of SDEs:

$\{x(0) = out(x) \text{ and } x' = tr(x)\}_{x \in X}$

The (unique) **solution** is given by the collection of streams:

 ${f(x)}_{x\in X}$

These streams **are** a solution of the SDEs, since

f(x)(0) = out(x) and f(x)' = tr(x)

A D F A 同 F A E F A E F A Q A

Solutions by finality

System of SDEs:

$\{x(0) = out(x) \text{ and } x' = tr(x)\}_{x \in X}$

The (unique) **solution** is given by the collection of streams: $\{f(x)\}_{x \in X}$

These streams **are** a solution of the SDEs, since

f(x)(0) = out(x) and f(x)' = tr(x)

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Solutions by finality

System of SDEs:

$$\{x(0) = out(x) \text{ and } x' = tr(x)\}_{x \in X}$$

The (unique) **solution** is given by the collection of streams:

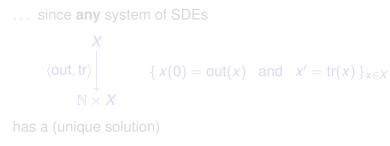
 ${f(x)}_{x\in X}$

These streams are a solution of the SDEs, since

f(x)(0) = out(x) and f(x)' = tr(x)

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Stream calculus is easy ...



 ${f(x)}_{x\in X}$

given by finality:

 $\begin{array}{c|c} X - - & \exists f \\ & \langle \text{out}, \text{tr} \rangle \\ & \downarrow \\ & \mathbb{N} \times X - - - \to \mathbb{N} \times \mathbb{N}^{\omega} \end{array}$

Stream calculus is easy ...

... since any system of SDEs

$$\begin{array}{c} X \\ \langle \mathsf{out}, \mathsf{tr} \rangle \\ \downarrow \\ \mathbb{N} \times X \end{array} \quad \{ x(0) = \mathsf{out}(x) \text{ and } x' = \mathsf{tr}(x) \}_{x \in X} \end{array}$$

has a (unique solution)

 ${f(x)}_{x\in X}$

given by finality:

$$\begin{array}{c|c} X - - & - \stackrel{\exists !}{-} \stackrel{f}{-} - \rightarrow \mathbb{N}^{\omega} \\ \langle \mathsf{out}, \mathsf{tr} \rangle & & \downarrow \langle \mathsf{head}, \mathsf{tail} \rangle \\ & & \mathbb{N} \times X - - - \rightarrow \mathbb{N} \times \mathbb{N}^{\omega} \end{array}$$

◆□▶ ◆□▶ ◆ □▶ ◆ □ ◆ ○ ◆ ○ ◆ ○ ◆

Stream calculus is easy ...

... since any system of SDEs

$$\begin{array}{c} X \\ \langle \mathsf{out},\mathsf{tr} \rangle \\ \downarrow \\ \mathbb{N} \times X \end{array} \quad \{ x(0) = \mathsf{out}(x) \text{ and } x' = \mathsf{tr}(x) \}_{x \in X} \end{array}$$

has a (unique solution)

 ${f(x)}_{x\in X}$

given by finality:

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

Example

$$\begin{cases} \{x, y\} - - -\exists ! f \\ - \to \mathbb{N}^{\omega} \\ \langle \mathsf{out}, \mathsf{tr} \rangle \\ \\ \mathbb{N} \times \{x, y\} - - - \to \mathbb{N} \times \mathbb{N}^{\omega} \end{cases}$$

SDEs: x(0) = 0, x' = y and y(0) = 1, y' = xSolution: f(x) = (0, 1, 0, 1, ...), f(y) = (1, 0, 1, 0, ...)

Example: infinite system of SDEs

$$\mathbb{N}^{\omega} \times \mathbb{N}^{\omega} - - - \stackrel{\exists!}{=} - - \to \mathbb{N}^{\omega}$$

(out, tr)
$$\mathbb{N} \times (\mathbb{N}^{\omega} \times \mathbb{N}^{\omega}) - - - \to \mathbb{N} \times \mathbb{N}^{\omega}$$

SDEs:

 $(\sigma, \tau)(\mathbf{0}) = \sigma(\mathbf{0}) + \tau(\mathbf{0}), \quad (\sigma, \tau)' = (\sigma', \tau') \quad (\forall \sigma, \tau \in \mathbb{N}^{\omega})$

▲□▶▲□▶▲□▶▲□▶ □ のQ@

Solution:

 $f(\sigma, \tau) = (\sigma(0) + \tau(0), \ \sigma(1) + \tau(1), \ \ldots)$

Example: infinite system of SDEs

SDEs: $(\sigma + \tau)(\mathbf{0}) = \sigma(\mathbf{0}) + \tau(\mathbf{0}), \quad (\sigma + \tau)' = \sigma' + \tau' \quad (\forall \sigma, \tau \in \mathbb{N}^{\omega})$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Solution:

 $\sigma + \tau = (\sigma(0) + \tau(0), \ \sigma(1) + \tau(1), \ \ldots)$

Example: infinite system of SDEs

SDEs: $(\sigma + \tau)(\mathbf{0}) = \sigma(\mathbf{0}) + \tau(\mathbf{0}), \quad (\sigma + \tau)' = \sigma' + \tau' \quad (\forall \sigma, \tau \in \mathbb{N}^{\omega})$

Solution: This formula is not really relevant. SDE says it all. Example: in the end . . .

... we simply will say: Let the function

 $+:\mathbb{N}^{\omega}\times\mathbb{N}^{\omega}\to\mathbb{N}^{\omega}$

be given by the following system of SDEs:

 $(\sigma + au)(\mathbf{0}) = \sigma(\mathbf{0}) + \tau(\mathbf{0}), \quad (\sigma + au)' = \sigma' + \tau' \quad (\forall \sigma, \tau \in \mathbb{N}^{\omega})$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Example: shuffle product

Let the function

 $\otimes:\mathbb{N}^{\omega}\times\mathbb{N}^{\omega}\to\mathbb{N}^{\omega}$

be given by the following system of SDEs:

 $(\sigma \otimes \tau)(\mathbf{0}) = \sigma(\mathbf{0})\tau(\mathbf{0}), \quad (\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQで

Solution: $(\sigma \otimes \tau)(n) = \sum_{k=0}^{n} {n \choose k} \cdot \sigma(k) \cdot \tau(n-k)$

Example: shuffle product

Let the function

 $\otimes:\mathbb{N}^{\omega}\times\mathbb{N}^{\omega}\to\mathbb{N}^{\omega}$

・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

be given by the following system of SDEs:

 $(\sigma \otimes \tau)(\mathbf{0}) = \sigma(\mathbf{0})\tau(\mathbf{0}), \quad (\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$

Again: this formula is not important. SDE says it all.

Proofs by coinduction

 $R \subseteq \mathbb{N}^{\omega} \times \mathbb{N}^{\omega}$ is a stream bisimulation if $\forall (\sigma, \tau) \in R : (i) \ \sigma(0) = \tau(0) \text{ and } (ii) \ (\sigma', \tau') \in R$

Theorem [Coinduction proof principle]:

 $(\sigma, \tau) \in \mathbf{R} \Rightarrow \sigma = \tau$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Proof: exercise.

Proofs by coinduction

 $R \subseteq \mathbb{N}^{\omega} \times \mathbb{N}^{\omega}$ is a stream bisimulation if $\forall (\sigma, \tau) \in R : (i) \ \sigma(0) = \tau(0) \text{ and } (ii) \ (\sigma', \tau') \in R$

Theorem [Coinduction proof principle]:

 $(\sigma, \tau) \in \mathbf{R} \Rightarrow \sigma = \tau$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Proof: exercise.

Coinduction: example

For all $\sigma, \tau, \rho \in \mathbb{N}^{\omega}$:

$$(\sigma \otimes \tau) \otimes \rho = \sigma \otimes (\tau \otimes \rho)$$

Proof:

 $\boldsymbol{R} = \{ ((\sigma \otimes \tau) \otimes \rho, \ \sigma \otimes (\tau \otimes \rho)) \mid \ \sigma, \tau, \rho \in \mathbb{N}^{\omega} \}$

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

is a stream bisimulation relation *up-to* +.

Coinduction: example

For all $\sigma, \tau, \rho \in \mathbb{N}^{\omega}$:

$$(\sigma\otimes au)\otimes
ho=\,\sigma\otimes(au\otimes
ho)$$

Proof:

$${m R}=\;\{\; ((\sigma\otimes au)\otimes
ho,\;\sigma\otimes(au\otimes
ho))\;\mid\;\sigma, au,
ho\in\mathbb{N}^{\omega}\;\}$$

is a stream bisimulation relation up-to +, since

$$((\sigma \otimes \tau) \otimes \rho)' = (\sigma' \otimes \tau) \otimes \rho + (\sigma \otimes \tau') \otimes \rho + (\sigma \otimes \tau) \otimes \rho'$$

 $(\sigma \otimes (\tau \otimes \rho))' = \sigma' \otimes (\tau \otimes \rho) + \sigma \otimes (\tau' \otimes \rho) + \sigma \otimes (\tau \otimes \rho')$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Coinduction: example

For all $\sigma, \tau, \rho \in \mathbb{N}^{\omega}$:

$$(\sigma \otimes au) \otimes
ho = \sigma \otimes (au \otimes
ho)$$

Exercise: try and give a proof using the formula

$$(\sigma \otimes \tau)(n) = \sum_{k=0}^{n} {n \choose k} \cdot \sigma(k) \cdot \tau(n-k)$$

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

Coinduction-up-to

Cf. Milner, Sangiorgi

Coinduction-up-to really is: Algebra + Coalgebra

Cf. Coalgebraic bisimulation-up-to J. Rot, M. Bonsangue, and J. Rutten LNCS 7741, 2013

Cf. Hacking nondeterminism with induction and coinduction Filippo Bonchi and Damien Pous Commun. ACM Vol. 58(2), 2015

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

More in Lecture four.

Coinduction-up-to

Cf. Milner, Sangiorgi

Coinduction-up-to really is: Algebra + Coalgebra

Cf. Coalgebraic bisimulation-up-to J. Rot, M. Bonsangue, and J. Rutten LNCS 7741, 2013

Cf. Hacking nondeterminism with induction and coinduction Filippo Bonchi and Damien Pous Commun. ACM Vol. 58(2), 2015

(日) (日) (日) (日) (日) (日) (日)

More in Lecture four.

2. Solving systems of SDEs

Previous definition of SDEs: semantical.

Next: syntax.

- Given: a **syntactically** presented system of SDEs.
- Goal: find its solution.
- Answer: use the **syntactic method** to construct a suitable stream coalgebra.

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

- Use finality (as before) to get the solution.

2. Solving systems of SDEs

Previous definition of SDEs: **semantical**. Next: **syntax**.

- Given: a **syntactically** presented system of SDEs.
- Goal: find its solution.
- Answer: use the **syntactic method** to construct a suitable stream coalgebra.

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

- Use finality (as before) to get the solution.

2. Solving systems of SDEs

Previous definition of SDEs: semantical.

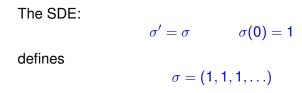
Next: syntax.

- Given: a **syntactically** presented system of SDEs.
- Goal: find its solution.
- Answer: use the **syntactic method** to construct a suitable stream coalgebra.

(日) (日) (日) (日) (日) (日) (日)

- Use finality (as before) to get the solution.

Examples



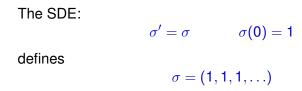
The SDE:

 $\sigma'' = \sigma' + \sigma$ $\sigma(0) = 1$ $\sigma'(0) = 1$

defines the Fibonacci numbers:

$$\sigma = (1, 1, 2, 3, 5, 8, \ldots)$$

▲□▶ ▲□▶ ▲□▶ ▲□▶ = 三 のへで



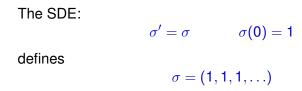
The SDE:

 $\sigma'' = \sigma' + \sigma$ $\sigma(0) = 1$ $\sigma'(0) = 1$

defines the Fibonacci numbers:

$$\sigma = (1, 1, 2, 3, 5, 8, \ldots)$$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>



The SDE:

 $\sigma'' = \sigma' + \sigma$ $\sigma(0) = 1$ $\sigma'(0) = 1$

defines the Fibonacci numbers:

$$\sigma = (1, 1, 2, 3, 5, 8, \ldots)$$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

The SDE: $(\sigma + \tau)' = \sigma' + \tau'$ $(\sigma + \tau)(0) = \sigma(0) + \tau(0)$

defines pointwise sum:

$$(\sigma + \tau)(\mathbf{n}) = \sigma(\mathbf{n}) + \tau(\mathbf{n})$$

The SDE:

 $(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$ $(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0)$ (where $[\sigma(0)] = (\sigma(0), 0, 0, 0, ...)$) defines convolution product:

$$(\sigma \times \tau)(n) = \sum_{k=0}^{n} \sigma(k) \cdot \tau(n-k)$$

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

The SDE: $(\sigma + \tau)' = \sigma' + \tau'$ $(\sigma + \tau)(0) = \sigma(0) + \tau(0)$

defines pointwise sum:

$$(\sigma + \tau)(\mathbf{n}) = \sigma(\mathbf{n}) + \tau(\mathbf{n})$$

The SDE:

 $(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$ $(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0)$ (where $[\sigma(0)] = (\sigma(0), 0, 0, 0, ...)$) defines convolution product:

$$(\sigma \times \tau)(n) = \sum_{k=0}^{n} \sigma(k) \cdot \tau(n-k)$$

A D F A 同 F A E F A E F A Q A

The SDE: $(\sigma + \tau)' = \sigma' + \tau'$ $(\sigma + \tau)(0) = \sigma(0) + \tau(0)$

defines pointwise sum:

$$(\sigma + \tau)(\mathbf{n}) = \sigma(\mathbf{n}) + \tau(\mathbf{n})$$

The SDE:

 $(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$ $(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0)$ (where $[\sigma(0)] = (\sigma(0), 0, 0, 0, ...)$) defines convolution product:

$$(\sigma \times \tau)(n) = \sum_{k=0}^{n} \sigma(k) \cdot \tau(n-k)$$

The SDE: $(\sigma + \tau)' = \sigma' + \tau'$ $(\sigma + \tau)(0) = \sigma(0) + \tau(0)$

defines pointwise sum:

$$(\sigma + \tau)(\mathbf{n}) = \sigma(\mathbf{n}) + \tau(\mathbf{n})$$

The SDE:

 $(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$ $(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0)$ (where $[\sigma(0)] = (\sigma(0), 0, 0, 0, ...)$) defines convolution product:

$$(\sigma \times \tau)(n) = \sum_{k=0}^{n} \sigma(k) \cdot \tau(n-k)$$

The syntactic method

A general method for solving systems of SDEs.

It works for a fairly large class of systems of SDEs.

We explain it by means of an example: the Hamming numbers.

(ロ) (同) (三) (三) (三) (○) (○)

The syntactic method

A general method for solving systems of SDEs.

It works for a fairly large class of systems of SDEs.

We explain it by means of an example: the Hamming numbers.

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

The Hamming numbers

Cf. Dijkstra's [EDW792].

All natural numbers, in increasing order, that have no other prime factors than 2 and 3 (and 5):

 $\begin{array}{lll} \gamma & = & (2^03^0,\, 2^13^0,\, 2^03^1,\, 2^23^0,\, 2^13^1,\, 2^33^0,\, 2^03^2,\, 2^23^1,\, \ldots) \\ & = & (1,2,3,4,6,8,9,12,\, \ldots) \end{array}$

We define γ by the stream differential equation

 $\gamma' = (2 \times \gamma) \parallel (3 \times \gamma) \qquad \gamma(0) = 1$

(日) (日) (日) (日) (日) (日) (日)

Note: this is not classical mathematics.

The Hamming numbers

Cf. Dijkstra's [EDW792].

All natural numbers, in increasing order, that have no other prime factors than 2 and 3 (and 5):

 $\begin{array}{lll} \gamma & = & (2^03^0,\, 2^13^0,\, 2^03^1,\, 2^23^0,\, 2^13^1,\, 2^33^0,\, 2^03^2,\, 2^23^1,\, \ldots) \\ & = & (1,2,3,4,6,8,9,12,\, \ldots) \end{array}$

We define γ by the stream differential equation

 $\gamma' = (2 \times \gamma) \parallel (3 \times \gamma) \qquad \gamma(0) = 1$

Note: this is not classical mathematics.

The stream differential equation

$$\gamma' = (2 \times \gamma) \parallel (3 \times \gamma) \qquad \gamma(0) = 1$$

Here the *ordered merge* $\|: \mathbb{N}^{\omega} \times \mathbb{N}^{\omega} \to \mathbb{N}^{\omega}$ is defined by

$$(\sigma \parallel \tau)' = \begin{cases} \sigma' \parallel \tau & \text{if } \sigma(0) < \tau(0) \\ \sigma' \parallel \tau' & \text{if } \sigma(0) = \tau(0) \\ \sigma \parallel \tau' & \text{if } \sigma(0) > \tau(0) \end{cases}$$
$$(\sigma \parallel \tau)(0) = \begin{cases} \sigma(0) & \text{if } \sigma(0) < \tau(0) \\ \tau(0) & \text{if } \sigma(0) \ge \tau(0) \end{cases}$$

and $2 \times \sigma$ (and similarly $3 \times \sigma$) is defined by

 $(2 \times \sigma)' = 2 \times (\sigma')$ $(2 \times \sigma)(0) = 2 \cdot \sigma(0)$

Syntactic solution method

Goal: to prove the unique existence of a solution for

 $\gamma' = (2 imes \gamma) \parallel (3 imes \gamma) \qquad \gamma(0) = 1$

Assuming the solution exists, we compute the first few derivatives of γ :

 $\begin{array}{lll} \gamma^{(1)} &=& (2 \times \gamma) \parallel (3 \times \gamma) \\ \gamma^{(2)} &=& (2 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \parallel (3 \times \gamma) \\ \gamma^{(3)} &=& (2 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \parallel (3 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \end{array}$

The idea: define **syntactic** terms for all possible such righthand sides.

(日) (日) (日) (日) (日) (日) (日)

Syntactic solution method

Goal: to prove the unique existence of a solution for

$$\gamma' = (\mathbf{2} \times \gamma) \parallel (\mathbf{3} \times \gamma) \qquad \gamma(\mathbf{0}) = \mathbf{1}$$

Assuming the solution exists, we compute the first few derivatives of γ :

$$\begin{array}{lll} \gamma^{(1)} &=& (2 \times \gamma) \parallel (3 \times \gamma) \\ \gamma^{(2)} &=& (2 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \parallel (3 \times \gamma) \\ \gamma^{(3)} &=& (2 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \parallel (3 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \end{array}$$

The idea: define **syntactic** terms for all possible such righthand sides.

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

Syntactic solution method

Goal: to prove the unique existence of a solution for

 $\gamma' = (2 \times \gamma) \parallel (3 \times \gamma) \qquad \gamma(0) = 1$

Assuming the solution exists, we compute the first few derivatives of γ :

$$\begin{array}{lll} \gamma^{(1)} &=& (2 \times \gamma) \parallel (3 \times \gamma) \\ \gamma^{(2)} &=& (2 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \parallel (3 \times \gamma) \\ \gamma^{(3)} &=& (2 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \parallel (3 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \end{array}$$

The idea: define **syntactic** terms for all possible such righthand sides.

The term coalgebra

Term $\ni t ::= c \mid \underline{\sigma} (\sigma \in \mathbb{N}^{\omega}) \mid 2 \text{times}(t) \mid 3 \text{times}(t) \mid \text{merge}(t_1, t_2)$

Next we turn the set Term into a stream coalgebra

 $\mathsf{Term} \xrightarrow{\langle \mathsf{out}, \mathsf{tr} \rangle} \mathbb{N} \times \mathsf{Term}$

by defining functions out : Term $\rightarrow \mathbb{N}$ and tr : Term \rightarrow Term by *induction* on the structure of terms, following the stream diff. eqn's.

(日) (日) (日) (日) (日) (日) (日)

The term coalgebra

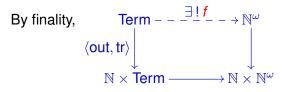
Term $\ni t ::= c \mid \underline{\sigma} (\sigma \in \mathbb{N}^{\omega}) \mid 2 \text{times}(t) \mid 3 \text{times}(t) \mid \text{merge}(t_1, t_2)$

Next we turn the set Term into a stream coalgebra

 $\mathsf{Term} \xrightarrow{\left<\mathsf{out}, \, \mathsf{tr}\right>} \mathbb{N} \times \mathsf{Term}$

by defining functions out : Term $\rightarrow \mathbb{N}$ and tr : Term \rightarrow Term by *induction* on the structure of terms, following the stream diff. eqn's.

The solution



Using *f*, we define

 $\gamma = f(c)$ $\sigma \parallel \tau = f(\text{merge}(\underline{\sigma}, \underline{\tau}))$

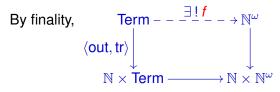
(and similarly for $2 \times \sigma$ and $3 \times \sigma$).

Finally one shows that, indeed,

$$\gamma' = (2 imes \gamma) \parallel (3 imes \gamma) \qquad \gamma(0) = 1$$

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

The solution



Using *f*, we define

 $\gamma = f(c)$ $\sigma \parallel \tau = f(merge(\underline{\sigma}, \underline{\tau}))$

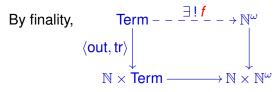
(and similarly for $2 \times \sigma$ and $3 \times \sigma$).

Finally one shows that, indeed,

$$\gamma' = (2 imes \gamma) \parallel (3 imes \gamma) \qquad \gamma(0) = 1$$

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

The solution



Using *f*, we define

 $\begin{array}{rcl} \gamma & = & \textbf{f}(\textbf{c}) \\ \sigma \parallel \tau & = & \textbf{f}(\mathsf{merge}(\underline{\sigma}, \underline{\tau})) \end{array}$

(and similarly for $2 \times \sigma$ and $3 \times \sigma$).

Finally one shows that, indeed,

$$\gamma' = (2 \times \gamma) \parallel (3 \times \gamma) \qquad \gamma(0) = 1$$

(ロ) (同) (三) (三) (三) (○) (○)

Let the function

 $\operatorname{even}:\mathbb{N}^\omega\to\mathbb{N}^\omega$

be given by the following system of SDEs:

 $(\operatorname{even}(\sigma))(0) = \sigma(0), \quad \operatorname{even}(\sigma)' = \operatorname{even}(\sigma'')$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

(Solution: even(σ) = (σ (0), σ (2), σ (4), ...).)

Let the function

 $\operatorname{even}:\mathbb{N}^\omega\to\mathbb{N}^\omega$

be given by the following system of SDEs:

 $(\operatorname{even}(\sigma))(0) = \sigma(0), \quad \operatorname{even}(\sigma)' = \operatorname{even}(\sigma'')$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

(Solution: even(σ) = (σ (0), σ (2), σ (4), ...).)

Now consider the following SDE:

x(0) = 0 x' = even(x)

It has **many** solutions, such as

 $x = (0, 0, 0, \ldots)$ $x = (0, 0, 1, 1, 1, \ldots)$

 $x = (0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, \ldots)$

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

Exercise: how many solutions are there?

Now consider the following SDE:

$$x(0) = 0$$
 $x' = \operatorname{even}(x)$

It has many solutions, such as

$$x = (0, 0, 0, \ldots)$$
 $x = (0, 0, 1, 1, 1, \ldots)$

 $x = (0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, \ldots)$

▲□▶ ▲□▶ ▲□▶ ▲□▶ □ のQで

Exercise: how many solutions are there?

Now consider the following SDE:

$$x(0) = 0$$
 $x' = \operatorname{even}(x)$

It has many solutions, such as

$$x = (0, 0, 0, \ldots)$$
 $x = (0, 0, 1, 1, 1, \ldots)$

 $x = (0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, \ldots)$

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

Exercise: how many solutions are there?

The syntactic format is important

The syntactic method does not work for

x(0) = 0 $x' = \operatorname{even}(x)$

The problem is that it does **not** translate uniquely to a corresponding stream coalgebra.

The technical problem is the second derivative in

 $\operatorname{even}(\sigma)' = \operatorname{even}(\sigma'')$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

The syntactic format is important

The syntactic method does not work for

x(0) = 0 $x' = \operatorname{even}(x)$

The problem is that it does **not** translate uniquely to a corresponding stream coalgebra.

The technical problem is the second derivative in

 $\operatorname{even}(\sigma)' = \operatorname{even}(\sigma'')$

The syntactic format is important

The syntactic method does not work for

x(0) = 0 $x' = \operatorname{even}(x)$

The problem is that it does **not** translate uniquely to a corresponding stream coalgebra.

The technical problem is the second derivative in

 $\operatorname{even}(\sigma)' = \operatorname{even}(\sigma'')$

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

3. Formats for SDEs

- A general format for the syntactic method

◆□▶ ◆□▶ ▲□▶ ▲□▶ □ のQ@

- Three well-known sub-classes:
 - Periodic streams
 - Rational streams
 - Context-free streams
- (Cf. formal languages.)

A useful set of operators on \mathbb{R}^{ω}

 $[r] = (r, 0, 0, 0, \ldots)$ for each $r \in \mathbb{R}$

 $X = (0, 1, 0, 0, 0, \ldots)$

 $(\sigma + \tau)(\mathbf{n}) = \sigma(\mathbf{n}) + \tau(\mathbf{n})$

$$(\sigma \times \tau)(n) = \sum_{k=0}^{n} \sigma(k) \cdot \tau(n-k)$$
$$\sigma \times \sigma^{-1} = [1] \qquad (\sigma(0) \neq 0)$$

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● のへぐ

The corresponding system of SDEs

derivative:	initial value:
[r]' = [0]	[r](0) = r
X' = [1]	X(0) = 0
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + au)(0) = \sigma(0) + \tau(0)$
$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$	$(\sigma imes au)(0) = \sigma(0) \cdot au(0)$
$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$

▲□▶ ▲□▶ ▲□▶ ▲□▶ = 三 のへで

derivative:	initial value:
[r]' = [0]	[r](0) = r
X' = [1]	X(0) = 0
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$
$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$	$(\sigma imes au)(0) = \sigma(0) \cdot au(0)$
$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$

The syntactic method applies in general to this kind of SDEs.

(日) (日) (日) (日) (日) (日) (日)

We shall explain "this kind".

derivative:	initial value:
[r]' = [0]	[r](0) = r
X' = [1]	X(0) = 0
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$
$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$	$(\sigma imes au)(0) = \sigma(0) \cdot au(0)$
$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$

The syntactic method applies in general to this kind of SDEs.

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

We shall explain "this kind".

derivative:

$$[r]' = [0]$$

$$X' = [1]$$

$$(\sigma + \tau)' = \sigma' + \tau'$$

$$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$$

$$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$$

On the left: terms with one operator (possibly a constant) ...

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

derivative:

$$\begin{aligned} [r]' &= [0] \\ X' &= [1] \\ (\sigma + \tau)' &= \sigma' + \tau' \\ (\sigma \times \tau)' &= (\sigma' \times \tau) + ([\sigma(0)] \times \tau') \\ (\sigma^{-1})' &= -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1} \end{aligned}$$

▲□▶ ▲□▶ ▲三▶ ▲三▶ 三三 のへで

On the left: ... and stream variables.

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

On the right: terms built from various operators

derivative:

$$[r]' = [0]$$

$$X' = [1]$$

$$(\sigma + \tau)' = \sigma' + \tau'$$

$$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$$

$$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$$

< □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □

On the right: ... and stream variables

Illustrating the format for our syntactic method

derivative:

$$[r]' = [0]$$

$$X' = [1]$$

$$(\sigma + \tau)' = \sigma' + \tau'$$

$$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$$

$$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$$

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

On the right: ... and derivatives of stream variables ...

(no double derivatives)

Illustrating the format for our syntactic method

derivative:

$$[r]' = [0]$$

$$X' = [1]$$

$$(\sigma + \tau)' = \sigma' + \tau'$$

$$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$$

$$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$$

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

On the right: ... and derivatives of stream variables ...

(no double derivatives)

Illustrating the format for our syntactic method

derivative:

$$[r]' = [0]$$

$$X' = [1]$$

$$(\sigma + \tau)' = \sigma' + \tau'$$

$$(\sigma \times \tau)' = \sigma' \times \tau) + ([\sigma(0)] \times \tau')$$

$$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$$

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

On the right: ... and initial values of stream variables.

The syntactic method

Theorem

Any system of SDEs such as

derivative:	initial value:
[r]' = [0]	[r](0) = r
X' = [1]	X(0) = 0
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$
$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$	$(\sigma imes au)(0) = \sigma(0) \cdot au(0)$
$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

has a unique solution.

Proof: By the syntactic method.

The syntactic method

Theorem

Any system of SDEs such as

derivative:	initial value:
[r]' = [0]	[r](0) = r
X' = [1]	X(0) = 0
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$
$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$	$(\sigma imes au)(0) = \sigma(0) \cdot au(0)$
$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$

◆□▶ ◆□▶ ▲□▶ ▲□▶ □ のQ@

has a unique solution.

Proof: By the syntactic method.

By **restricting** our format further, we obtain various concrete classes of streams.

◆□▶ ◆□▶ ▲□▶ ▲□▶ □ のQ@

We mention three of them:

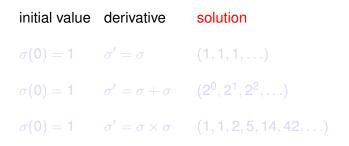
- Periodic streams
- Rational streams
- Context-free streams

By **restricting** our format further, we obtain various concrete classes of streams.

▲□▶ ▲□▶ ▲ 三▶ ▲ 三▶ - 三 - のへぐ

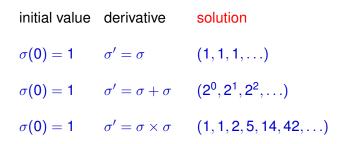
We mention three of them:

- Periodic streams
- Rational streams
- Context-free streams



Catalan numbers

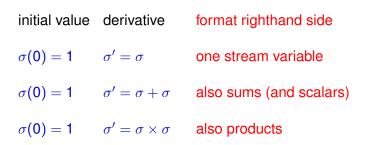
< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>



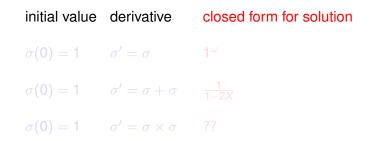
Catalan numbers



◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ◆ □ ▶ ◆ □ ▶

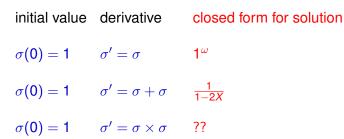


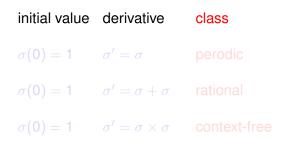
◆□▶ ◆□▶ ◆ □▶ ◆ □▶ ─ □ ─ の < @



・ロト・(四ト・(川下・(日下))

◆□▶ ◆□▶ ◆ □▶ ◆ □▶ ─ □ ─ の < @





◆□ ▶ ◆□ ▶ ◆ □ ▶ ◆ □ ▶ ● □ ● ● ● ●

initial value	derivative	class
$\sigma(0) = 1$	$\sigma'=\sigma$	perodic
$\sigma(0) = 1$	$\sigma'=\sigma+\sigma$	rational
$\sigma(0) = 1$	$\sigma' = \sigma \times \sigma$	context-free

4. Streams and coinduction

We saw an elementary example of coinduction (when proving the associativity of the shuffle product).

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

Time allowing, we will next illustrate the coinduction proof principle for streams with a non-trivial example.

4. Streams and coinduction

We saw an elementary example of coinduction (when proving the associativity of the shuffle product).

(日) (日) (日) (日) (日) (日) (日)

Time allowing, we will next illustrate the coinduction proof principle for streams with a non-trivial example.

A proof by coinduction: Moessner's theorem

- A. Moessner (1951), proof by O. Perron (1951) and I. Paasche (1952).
- Cf. Ralf Hinze: Scans and convolutions a calculational proof of Moessner's theorem (Oxford University, 2010).
- Our proof: by coinduction (Niqui & R., 2011) . . .
- . . . is a student's exercise.
- Cf. the original proof: serious binomial coefficient manipulation!!

(日) (日) (日) (日) (日) (日) (日)

nat	1	2	3	4	5	6	7	8	9	10	11	12	
Drop ₃	1	2		4	5		7			10	11		
Σ	1	3	7	12	19	27	37	48					
Drop ₂	1		7		19		37						
Σ	1		27	64									
nat ³	1 ³	2 ³	3 ³	4 ³									

nat	1	2	3	4	5	6	7	8	9	10	11	12	••••
Drop ₃	1	2		4	5		7	8		10	11		
Σ	1	3	7	12	19	27	37	48					
Drop ₂	1		7		19		37						
Σ	1		27	64									
nat ³	1 ³	2 ³	3 ³	4 ³									

nat	1	2	3	4	5	6	7	8	9	10	11	12	•••
Drop ₃	1	2		4	5		7	8		10	11		
Σ	1	3	7	12	19	27	37	48					
Drop ₂	1		7		19		37						
Σ	1		27	64									
not ³	13	ე 3	2 3	⊿3									

nat	1	2	3	4	5	6	7	8	9	10	11	12	••••
Drop ₃	1	2		4	5		7	8		10	11		
Σ	1	3	7	12	19	27	37	48					
Drop ₂	1		7		19		37						
Σ	1		27	64									
nat ³	1 ³	2 ³	3 ³	4 ³									

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

nat	1	2	3	4	5	6	7	8	9	10	11	12	••••
Drop ₃	1	2		4	5		7	8		10	11		
Σ	1	3	7	12	19	27	37	48					
Drop ₂	1		7		19		37						
Σ	1	8	27	64									
nat ³	1 ³	2 ³	3 ³	4 ³									

◆□▶ ◆□▶ ◆臣▶ ◆臣▶ 臣 のへぐ

nat	1	2	3	4	5	6	7	8	9	10	11	12	
Drop ₃	1	2		4	5		7	8		10	11		
Σ	1	3	7	12	19	27	37	48					
Drop ₂	1		7		19		37						
Σ	1	8	27	64									
	=												
nat ³	1 ³	2 ³	3 ³	4 ³									

nat	1	2	3	4	5	6	7	8	9	10	11	
Drop ₄	1	2	3		5	6	7		9	10	11	
Σ	1	3	6	11	17	24	33	43	54			
Drop ₃	1	3		11	17		33	43		67	81	
Σ	1	4	15	32	65	108	175					
Drop ₂	1		15		65		175					
Σ	1	16	81	256								
		14	2 ⁴	3 ⁴	4 ⁴							

nat	1	2	3	4	5	6	7	8	9	10	11	•••	
Drop ₄	1	2	3		5	6	7		9	10	11		
Σ	1	3	6	11	17	24	33	43	54				
Drop ₃	1	3		11	17		33	43		67	81		
Σ	1	4	15	32	65	108	175						
Drop ₂	1		15		65		175						
Σ	1	16	81	256									
		14	2 ⁴	3 ⁴	4 ⁴								

nat	1	2	3	4	5	6	7	8	9	10	11	
Drop ₄	1	2	3		5	6	7		9	10	11	
Σ	1	3	6	11	17	24	33	43	54			
Drop ₃	1	3		11	17		33	43		67	81	
Σ	1	4	15	32	65	108	175					
Drop ₂	1		15		65		175					
Σ	1	16	81	256								
		1 ⁴	2 ⁴	3 ⁴	4 ⁴							

nat	1	2	3	4	5	6	7	8	9	10	11	••••
Drop ₄	1	2	3		5	6	7		9	10	11	
Σ	1	3	6	11	17	24	33	43	54			
Drop ₃	1	3		11	17		33	43		67	81	
Σ	1	4	15	32	65	108	175					
Drop ₂	1		15		65		175					
Σ	1	16	81	256								
		14	2 ⁴	3 ⁴	4 ⁴							

nat	1	2	3	4	5	6	7	8	9	10	11	•••
Drop ₄	1	2	3		5	6	7		9	10	11	
Σ	1	3	6	11	17	24	33	43	54			
Drop ₃	1	3		11	17		33	43		67	81	
Σ	1	4	15	32	65	108	175	•••				
Σ Drop ₂												
	1		15		65							

・ロト・四ト・モート ヨー うへの

nat	1	2	3	4	5	6	7	8	9	10	11	••••
Drop ₄	1	2	3		5	6	7		9	10	11	
Σ	1	3	6	11	17	24	33	43	54			
Drop ₃	1	3		11	17		33	43		67	81	
Σ	1	4	15	32	65	108	175					
Drop ₂	1		15		65		175					
Σ	1	16	81	256								
		1 ⁴	2 ⁴	3 ⁴	4 ⁴							

・ロト・四ト・モート ヨー うへの

nat	1	2	3	4	5	6	7	8	9	10	11	••••
Drop ₄	1	2	3		5	6	7		9	10	11	
Σ	1	3	6	11	17	24	33	43	54			
Drop ₃	1	3		11	17		33	43		67	81	
Σ	1	4	15	32	65	108	175					
Drop ₂	1		15		65		175					
Σ	1	16	81	256								
		1 ⁴	2 ⁴	3 ⁴	4 ⁴							

・ロト・四ト・モート ヨー うへの

nat	1	2	3	4	5	6	7	8	9	10	11	
Drop ₄	1	2	3		5	6	7		9	10	11	
Σ	1	3	6	11	17	24	33	43	54			
Drop ₃	1	3		11	17		33	43		67	81	
Σ	1	4	15	32	65	108	175					
Drop ₂	1		15		65		175					
Σ	1	16	81	256								
	=	1 ⁴	2 ⁴	3 ⁴	4 ⁴							

nat	1	2	3	4	5	6	7	8	9	10	11	••••	
Drop ₅	1	2	3	4		6	7		9		11		
Σ	1	3	6	10	16	23	31	40	51				
Drop ₄	1	3	6		16	23	31		51				
etc.													
		15	25	3 5	4 5								

nat	1	2	3	4	5	6	7	8	9	10	11	••••	
Drop ₅	1	2	3	4		6	7	8	9		11		
Σ	1	3	6	10	16	23	31	40	51				
Drop ₄	1	3	6		16	23	31		51				
etc.													
		15	25	3 5	⊿ 5								

nat	1	2	3	4	5	6	7	8	9	10	11	••••	
Drop ₅	1	2	3	4		6	7	8	9		11		
Σ	1	3	6	10	16	23	31	40	51				
Drop ₄	1	3	6		16	23	31		51				
etc.													
		15	2 ⁵	3 ⁵	4 ⁵								

nat	1	2	3	4	5	6	7	8	9	10	11	••••
Drop ₅	1	2	3	4		6	7	8	9		11	
Σ	1	3	6	10	16	23	31	40	51			
Drop ₄	1	3	6		16	23	31		51			
etc.												
		15	2 ⁵	3 5	4 ⁵							

Moessner's theorem (k = 5)

nat	1	2	3	4	5	6	7	8	9	10	11	••••
Drop ₅	1	2	3	4		6	7	8	9		11	
Σ	1	3	6	10	16	23	31	40	51			
Drop ₄	1	3	6		16	23	31		51			
etc.												
		1 ⁵	2 ⁵	3 ⁵	4 ⁵							

Moessner's theorem (k = 5)

nat	1	2	3	4	5	6	7	8	9	10	11	••••	
Drop ₅	1	2	3	4		6	7	8	9		11		
Σ	1	3	6	10	16	23	31	40	51				
Drop ₄	1	3	6		16	23	31		51				
etc.													
	=	1 ⁵	2 ⁵	3 ⁵	4 ⁵								

Coinduction proof principle for streams:

 $(\sigma, \tau) \in \mathbf{R}$, bisimulation relation $\Rightarrow \sigma = \tau$

We formulate Moessner's theorem as an equality of two streams.

Next we shall prove that these streams are equal

... by showing that they **behave** the same.

That is, we show that they are related by a bisimulation.

(日) (日) (日) (日) (日) (日) (日)

Coinduction proof principle for streams:

 $(\sigma, \tau) \in \mathbf{R}$, bisimulation relation $\Rightarrow \sigma = \tau$

We formulate Moessner's theorem as an equality of two streams.

Next we shall prove that these streams are equal

... by showing that they **behave** the same.

That is, we show that they are related by a bisimulation.

(日) (日) (日) (日) (日) (日) (日)

Coinduction proof principle for streams:

 $(\sigma, \tau) \in \mathbf{R}$, bisimulation relation $\Rightarrow \sigma = \tau$

We formulate Moessner's theorem as an equality of two streams.

Next we shall prove that these streams are equal

... by showing that they **behave** the same.

That is, we show that they are related by a bisimulation.

(ロ) (同) (三) (三) (三) (○) (○)

Coinduction proof principle for streams:

 $(\sigma, \tau) \in \mathbf{R}$, bisimulation relation $\Rightarrow \sigma = \tau$

We formulate Moessner's theorem as an equality of two streams.

Next we shall prove that these streams are equal

... by showing that they **behave** the same.

That is, we show that they are related by a bisimulation.

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

We will define all of the above ingredients using

stream differential equations

< □ > < 同 > < Ξ > < Ξ > < Ξ > < Ξ < </p>

This will

- make the inherent circularity explicit, and
- help us contruct a suitable **bisimulation** relation!

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

We will define all of the above ingredients using

stream differential equations

< □ > < 同 > < Ξ > < Ξ > < Ξ > < Ξ < </p>

This will

- make the inherent circularity explicit, and
- help us contruct a suitable **bisimulation** relation!

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

```
where nat = (1,2,3,...) satisfies

nat(0) = 1 nat' = nat + ones

with ones = (1,1,1,...); and

nat<sup>3</sup> = (1^3,2^3,3^3,...) = nat \odot nat \odot nat

with
```

 $(\sigma \odot \tau)(\mathbf{0}) = \sigma(\mathbf{0}) \cdot \tau(\mathbf{0}) \qquad (\sigma \odot \tau)' = \sigma' \odot \tau'$

◆□▶ ◆□▶ ◆三▶ ◆三▶ ● ● ●

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

where nat = (1, 2, 3, ...) satisfies nat(0) = 1 nat' = nat + ones with ones = (1, 1, 1, ...); and nat³ = $(1^3, 2^3, 3^3, ...)$ = nat \odot nat \odot nat with

 $(\sigma \odot \tau)(\mathbf{0}) = \sigma(\mathbf{0}) \cdot \tau(\mathbf{0}) \qquad (\sigma \odot \tau)' = \sigma' \odot \tau'$

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

and where

 $\Sigma(\sigma) = (\sigma(0), \ \sigma(0) + \sigma(1), \ \sigma(0) + \sigma(1) + \sigma(2), \ldots)$

・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

 $Drop_2(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$

 $Drop_3(\sigma) = (\sigma(0), \sigma(1), \sigma(3), \sigma(4), \sigma(6), \sigma(7), \ldots)$

can all be specified by elementary stream diff. equations.

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

and where

 $\Sigma(\sigma) = (\sigma(0), \sigma(0) + \sigma(1), \sigma(0) + \sigma(1) + \sigma(2), \ldots)$

A D F A 同 F A E F A E F A Q A

 $Drop_2(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$

 $Drop_3(\sigma) = (\sigma(0), \sigma(1), \sigma(3), \sigma(4), \sigma(6), \sigma(7), \ldots)$

can all be specified by elementary stream diff. equations.

$nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

It now suffices to construct a bisimulation R with $\langle nat^3, \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat) \rangle \in R$

Easy, using the previous stream differential equations . . .

・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

$nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

It now suffices to construct a bisimulation R with $\langle nat^3, \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat) \rangle \in R$

Easy, using the previous stream differential equations . . .

< □ > < 同 > < Ξ > < Ξ > < Ξ > < Ξ < </p>

$nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

It now suffices to construct a bisimulation R with $\langle nat^3, \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat) \rangle \in R$

Easy, using the previous stream differential equations . . .

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

$nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

Proof: We define *R* as the smallest set such that (i) $\langle nat^3, \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat) \rangle \in R$ (ii) $\langle nat \odot (nat + ones)^2, \Sigma \circ Drop_2^0 \circ \Sigma \circ Drop_3^1(nat) \rangle \in R$ (iii) if $\langle \sigma_1, \sigma_2 \rangle \in R$ and $\langle \tau_1, \tau_2 \rangle \in R$ then $\langle \sigma_1 + \tau_1, \sigma_2 + \tau_2 \rangle \in R$ (iv) $\langle \sigma, \sigma \rangle \in R$ (all σ)

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

Proof: We define *R* as the smallest set such that (i) $\langle \operatorname{nat}^3, \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(\operatorname{nat}) \rangle \in R$ (ii) $\langle \operatorname{nat} \odot (\operatorname{nat} + \operatorname{ones})^2, \Sigma \circ Drop_2^0 \circ \Sigma \circ Drop_3^1(\operatorname{nat}) \rangle \in R$ (iii) if $\langle \sigma_1, \sigma_2 \rangle \in R$ and $\langle \tau_1, \tau_2 \rangle \in R$ then $\langle \sigma_1 + \tau_1, \sigma_2 + \tau_2 \rangle \in R$ (iv) $\langle \sigma, \sigma \rangle \in R$ (all σ)

・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・
 ・

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

Proof: We define *R* as the smallest set such that (i) $\langle \operatorname{nat}^3, \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(\operatorname{nat}) \rangle \in R$ (ii) $\langle \operatorname{nat} \odot (\operatorname{nat} + \operatorname{ones})^2, \Sigma \circ Drop_2^0 \circ \Sigma \circ Drop_3^1(\operatorname{nat}) \rangle \in R$ (iii) if $\langle \sigma_1, \sigma_2 \rangle \in R$ and $\langle \tau_1, \tau_2 \rangle \in R$ then $\langle \sigma_1 + \tau_1, \sigma_2 + \tau_2 \rangle \in R$ (iv) $\langle \sigma, \sigma \rangle \in R$ (all σ)

< □ > < 同 > < 三 > < 三 > < 三 > < ○ < ○ </p>

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

Proof: We define *R* as the smallest set such that (i) $\langle \operatorname{nat}^3, \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(\operatorname{nat}) \rangle \in R$ (ii) $\langle \operatorname{nat} \odot (\operatorname{nat} + \operatorname{ones})^2, \Sigma \circ Drop_2^0 \circ \Sigma \circ Drop_3^1(\operatorname{nat}) \rangle \in R$ (iii) if $\langle \sigma_1, \sigma_2 \rangle \in R$ and $\langle \tau_1, \tau_2 \rangle \in R$ then $\langle \sigma_1 + \tau_1, \sigma_2 + \tau_2 \rangle \in R$ (iv) $\langle \sigma, \sigma \rangle \in R$ (all σ)

A D F A 同 F A E F A E F A Q A

 $nat^3 = \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat)$

Proof: We define *R* as the smallest set such that (i) $\langle nat^3, \Sigma \circ Drop_2 \circ \Sigma \circ Drop_3(nat) \rangle \in R$ (ii) $\langle nat \odot (nat + ones)^2, \Sigma \circ Drop_2^0 \circ \Sigma \circ Drop_3^1(nat) \rangle \in R$ (iii) if $\langle \sigma_1, \sigma_2 \rangle \in R$ and $\langle \tau_1, \tau_2 \rangle \in R$ then $\langle \sigma_1 + \tau_1, \sigma_2 + \tau_2 \rangle \in R$ (iv) $\langle \sigma, \sigma \rangle \in R$ (all σ)

◆□▶ ◆□▶ ▲□▶ ▲□▶ □ のQ@

The proof for all *k*: make one **big bisimulation**

Proof has been verified in theorem prover (COQ), by Krebbers, Parlant, Silva.

◆□▶ ◆□▶ ▲□▶ ▲□▶ □ のQ@

The proof for all *k*: make one **big bisimulation**

Proof has been verified in theorem prover (COQ), by Krebbers, Parlant, Silva.

- We take streams σ as **basic entities**, instead of focussing on their individual **elements** σ(n).
- This prevents lots of unnecessary bookkeeping (cf. binomial coefficients).
- The (final) coalgebra structure of the set of streams has a natural interpretation in terms of a **calculus**, in analogy to classical calculus.
- There is initial evidence that this leads to efficient proofs that can be easily automated.

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

- We take streams σ as basic entities, instead of focussing on their individual elements σ(n).
- This prevents lots of unnecessary bookkeeping (cf. binomial coefficients).
- The (final) coalgebra structure of the set of streams has a natural interpretation in terms of a **calculus**, in analogy to classical calculus.
- There is initial evidence that this leads to efficient proofs that can be easily automated.

◆□▶ ◆□▶ ▲□▶ ▲□▶ ■ ののの

- We take streams σ as basic entities, instead of focussing on their individual elements σ(n).
- This prevents lots of unnecessary bookkeeping (cf. binomial coefficients).
- The (final) coalgebra structure of the set of streams has a natural interpretation in terms of a **calculus**, in analogy to classical calculus.
- There is initial evidence that this leads to efficient proofs that can be easily automated.

(日) (日) (日) (日) (日) (日) (日)

- We take streams σ as basic entities, instead of focussing on their individual elements σ(n).
- This prevents lots of unnecessary bookkeeping (cf. binomial coefficients).
- The (final) coalgebra structure of the set of streams has a natural interpretation in terms of a **calculus**, in analogy to classical calculus.
- There is initial evidence that this leads to efficient proofs that can be easily automated.

(日) (日) (日) (日) (日) (日) (日)

- We take streams σ as basic entities, instead of focussing on their individual elements σ(n).
- This prevents lots of unnecessary bookkeeping (cf. binomial coefficients).
- The (final) coalgebra structure of the set of streams has a natural interpretation in terms of a **calculus**, in analogy to classical calculus.
- There is initial evidence that this leads to efficient proofs that can be easily automated.