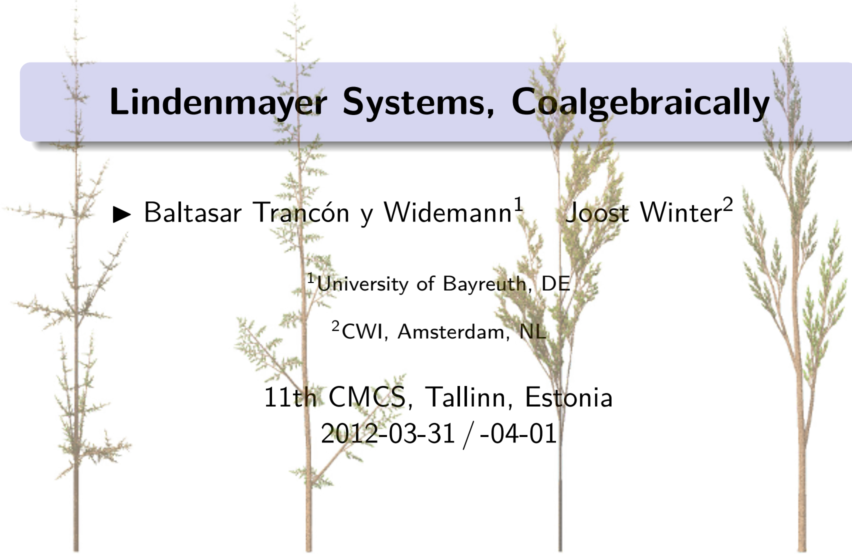


Lindenmayer Systems, Coalgebraically

- Baltasar Trancón y Widemann[1] Joost Winter[2]

[1]University of Bayreuth, DE

[2]CWI, Amsterdam, NL

11th CMCS, Tallinn, Estonia
2012-03-31 / -04-01

Context of Our Research

Work not quite in progress. . .

- Lindenmayer Systems
  - as example of behavioral environmental modelling in a lecture (2010, Bayreuth)
  - as running example for an invited tutorial on categories, algebra and coalgebra
    (2011 Workshop Young Modellers in Ecology, Wallenfels, DE)
- Context-free Grammars, Coalgebraically
  (2011 CALCO, Winchester, UK)
- How are the two related?
  (2011 CALCO Coffee Break)

Context of Our Research

Work not quite in progress. . .

- Lindenmayer Systems
  - as example of behavioral environmental modelling in a lecture (2010, Bayreuth)
  - as running example for an invited tutorial on categories, algebra and coalgebra
    (2011 Workshop Young Modellers in Ecology, Wallenfels, DE)
- Context-free Grammars, Coalgebraically
  (2011 CALCO, Winchester, UK)
- How are the two related?
  (2011 CALCO Coffee Break)

Context of Our Research

Work not quite in progress. . .

- Lindenmayer Systems
  - as example of behavioral environmental modelling in a lecture (2010, Bayreuth)
  - as running example for an invited tutorial on categories, algebra and coalgebra (2011 Workshop Young Modellers in Ecology, Wallenfels, DE)
- Context-free Grammars, Coalgebraically (2011 CALCO, Winchester, UK)
- How are the two related? (2011 CALCO Coffee Break)

History of Lindenmayer Systems

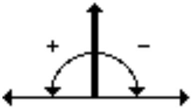
- Mathematical model for the growth of simple multicellular organisms: *yeasts, algae, fungi* (Lindenmayer 1968)
  - following the example of formal grammars (Chomsky 1957)
- Later generalized to complex organisms: *vascular plants*
- Graphical interpretation
  - from simple turtle graphics for theoreticians and children
  - to state-of-the-art photorealistic image synthesis

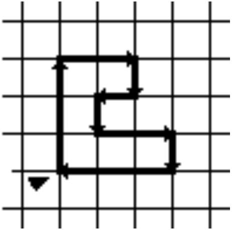
History of Lindenmayer Systems

- Mathematical model for the growth of simple multicellular organisms: *yeasts, algae, fungi* (Lindenmayer 1968)
  - following the example of formal grammars (Chomsky 1957)
- Later generalized to complex organisms: *vascular plants*
- Graphical interpretation
  - from simple turtle graphics for theoreticians and children
  - to state-of-the-art photorealistic image synthesis

History of Lindenmayer Systems

- Mathematical model for the growth of simple multicellular organisms: *yeasts, algae, fungi* (Lindenmayer 1968)
  - following the example of formal grammars (Chomsky 1957)
- Later generalized to complex organisms: *vascular plants*
- Graphical interpretation
  - from simple turtle graphics for theoreticians and children
  - to state-of-the-art photorealistic image synthesis





History of Lindenmayer Systems

- Mathematical model for the growth of simple multicellular organisms: *yeasts, algae, fungi* (Lindenmayer 1968)
  - following the example of formal grammars (Chomsky 1957)
- Later generalized to complex organisms: *vascular plants*
- Graphical interpretation
  - from simple turtle graphics for theoreticians and children
  - to state-of-the-art photorealistic image synthesis

Philosophy of Lindenmayer Systems

Growth is...

Replacement of building blocks by **more** building blocks

Decentral with **local** rules of replacement

Discrete with steps of **simultaneous** growth,
proceeding from one **global** stage to the next

Creation of form by establishing **neighbourship** between blocks,
in the simplest case **linear**

Lindenmayer Systems in Literature

The standard reference is **The Algorithmic Beauty of Plants** (Prusinkiewicz and Lindenmayer 1990, free high-quality PDF edition avaliable).

See also `http://algorithmicbotany.org/`.

Deterministic Context-free Lindenmayer Systems I

Classical Definition (Syntactic)

- A deterministic context-free L-System is a triple (V, ω, P) with
  - V a **finite** set
  - $\omega \in V^+$ an axiom
  - $P \subseteq V \times V^*$ a functional *rewrite* relation
- A derivation **step** of (V, ω, P) replaces each symbol v_i in a word $v_1 \cdots v_n \in V^*$ **simultaneously** by the subword w_i such that $(v_i, w_i) \in P$.
- The derivation **sequence** of (V, ω, P) is the infinite sequence of steps starting from ω.

Comparison to Grammars

- Parallel instead of serial rewriting
- No final state: *the journey is the reward*

Deterministic Context-free Lindenmayer Systems I

Classical Definition (Syntactic)

- A deterministic context-free L-System is a triple (V, ω, P) with
  - V a **finite** set
  - $\omega \in V^+$ an axiom
  - $P \subseteq V \times V^*$ a functional *rewrite* relation
- A derivation **step** of (V, ω, P) replaces each symbol v_i in a word $v_1 \cdots v_n \in V^*$ **simultaneously** by the subword w_i such that $(v_i, w_i) \in P$.
- The derivation **sequence** of (V, ω, P) is the infinite sequence of steps starting from ω.

Comparison to Grammars

- Parallel instead of serial rewriting
- No final state: *the journey is the reward*

Deterministic Context-free Lindenmayer Systems II

Coalgebraic Definition (Semantic)

- P is the graph of a function $p : V \to V^* = \mathcal{L}V$
  - trivial pairs (v, v) are omitted in writing
- Unpointed L-Systems are coalgebras (V, p) of the list functor $\mathcal{L}$
- Derivation steps apply p elementwise,
- and forget boundaries between subwords

Bottom Line

- L-Systems are essentially list coalgebras.
- L-System derivation is Kleisli extension in the list monad.

Deterministic Context-free Lindenmayer Systems II

Coalgebraic Definition (Semantic)

- P is the graph of a function $p : V \to V^* = \mathcal{L}V$
  - trivial pairs (v, v) are omitted in writing
- Unpointed L-Systems are coalgebras (V, p) of the list functor $\mathcal{L}$
- Derivation steps apply p elementwise,
- and forget boundaries between subwords

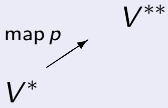
Bottom Line

- L-Systems are essentially list coalgebras.
- L-System derivation is Kleisli extension in the list monad.

Deterministic Context-free Lindenmayer Systems II

Coalgebraic Definition (Semantic)

- P is the graph of a function $p : V \to V^* = \mathcal{L}V$
  - trivial pairs (v, v) are omitted in writing
- Unpointed L-Systems are coalgebras (V, p) of the list functor $\mathcal{L}$
- Derivation steps apply p elementwise,
- and forget boundaries between subwords



Bottom Line

- L-Systems are essentially list coalgebras.
- L-System derivation is Kleisli extension in the list monad.

Deterministic Context-free Lindenmayer Systems II

Coalgebraic Definition (Semantic)

- P is the graph of a function $p : V \to V^* = \mathcal{L}V$
  - trivial pairs (v, v) are omitted in writing
- Unpointed L-Systems are coalgebras (V, p) of the list functor $\mathcal{L}$
- Derivation steps apply p elementwise,
- and forget boundaries between subwords

$$V^* \xrightarrow{\text{map } p} V^{**} \xrightarrow{\text{flatten}} V^*$$

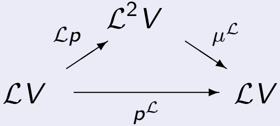
Bottom Line

- L-Systems are essentially list coalgebras.
- L-System derivation is Kleisli extension in the list monad.

Deterministic Context-free Lindenmayer Systems II

Coalgebraic Definition (Semantic)

- P is the graph of a function $p : V \to V^* = \mathcal{L}V$
  - trivial pairs (v, v) are omitted in writing
- Unpointed L-Systems are coalgebras (V, p) of the list functor $\mathcal{L}$
- Derivation steps apply p elementwise,
- and forget boundaries between subwords



Bottom Line

- L-Systems are essentially list coalgebras.
- L-System derivation is Kleisli extension in the list monad.

Deterministic Context-free Lindenmayer Systems II

Coalgebraic Definition (Semantic)

- P is the graph of a function $p : V \to V^* = \mathcal{L}V$
  - trivial pairs (v, v) are omitted in writing
- Unpointed L-Systems are coalgebras (V, p) of the list functor $\mathcal{L}$
- Derivation steps apply p elementwise,
- and forget boundaries between subwords

$$\begin{array}{ccc} & \mathcal{L}^2 V & \\ {\scriptstyle \mathcal{L}p} \nearrow & & \searrow {\scriptstyle \mu^{\mathcal{L}}} \\ \mathcal{L}V & \xrightarrow[p^{\mathcal{L}}]{} & \mathcal{L}V \end{array}$$

Bottom Line

- L-Systems are essentially list coalgebras.
- L-System derivation is Kleisli extension in the list monad.

Recipe: Composite Monads

General Idea

Define L-System extension *components* as monadic functors, to be composed (left or right) with $\mathcal{L}$.

Composite Monads

Bad News Two monadic functors S, T do not generally give rise to a monad for ST.

Good News A distributive law of T over S does the job.

$$\begin{matrix} \lambda : TS \Rightarrow ST \\ \cdots \end{matrix} \implies \begin{matrix} \eta^{ST} = \eta^S \eta^T \\ \mu^{ST} = \mu^S \mu^T \circ S\lambda T \end{matrix}$$

Excellent News The obvious distributive laws for L-System component monads are *exactly* the missing semantical links.

Recipe: Composite Monads

General Idea

Define L-System extension *components* as monadic functors, to be composed (left or right) with $\mathcal{L}$.

Composite Monads

Bad News Two monadic functors S, T do not generally give rise to a monad for ST.

Good News A distributive law of T over S does the job.

$$\begin{array}{c} \lambda : TS \Rightarrow ST \\ \dots \end{array} \implies \begin{array}{l} \eta^{ST} = \eta^S \eta^T \\ \mu^{ST} = \mu^S \mu^T \circ S\lambda T \end{array}$$

Excellent News The obvious distributive laws for L-System component monads are *exactly* the missing semantical links.

Recipe: Composite Monads

General Idea

Define L-System extension *components* as monadic functors, to be composed (left or right) with $\mathcal{L}$.

Composite Monads

Bad News Two monadic functors S, T do not generally give rise to a monad for ST.

Good News A distributive law of T over S does the job.

$$\begin{matrix} \lambda : TS \Rightarrow ST \\ \ldots \end{matrix} \implies \begin{matrix} \eta^{ST} = \eta^S \eta^T \\ \mu^{ST} = \mu^S \mu^T \circ S\lambda T \end{matrix}$$

Excellent News The obvious distributive laws for L-System component monads are *exactly* the missing semantical links.

Recipe: Composite Monads

General Idea

Define L-System extension *components* as monadic functors, to be composed (left or right) with $\mathcal{L}$.

Composite Monads

Bad News Two monadic functors S, T do not generally give rise to a monad for ST.

Good News A distributive law of T over S does the job.

$$\begin{array}{c} \lambda : TS \Rightarrow ST \\ \dots \end{array} \implies \begin{array}{l} \eta^{ST} = \eta^S \eta^T \\ \mu^{ST} = \mu^S \mu^T \circ S\lambda T \end{array}$$

Excellent News The obvious distributive laws for L-System component monads are *exactly* the missing semantical links.

Even More Composite Monads

What about multiple extensions?

- Fix order of nesting
- For a finite sequence of monads $S_1, \ldots, S_n$
  - find a "triangular matrix" of distributive laws

$$\lambda^{ij} : S_j S_i \Rightarrow S_i S_j \quad \text{for all } i < j$$

  - giving rise to compositions in any order of parentheses
  - which are all equivalent $\implies$ monad composition is associative

Examples of Extensions

Ordinary	$A \to AB$	$B \to A$	
+Terminals	$F \to F{+}F{-}{-}F{+}F$		
+Nondeterminism	$A \to AB$	$A \to BA$	$B \to A$
+Probabilism	$A \overset{1/3}{\to} AB$	$A \overset{2/3}{\to} BA$	$B \overset{1}{\to} A$
+Parameters	$I(t) \overset{t>0}{\to} I(t-1)$	$I(t) \overset{t=0}{\to} S$	

Terminals

Coproduct (Error) Monad

$$\mathcal{C}_A = (-) + A \qquad \eta^{\mathcal{C}_A} = \iota_1 \qquad \mu^{\mathcal{C}_A} = [\mathsf{id}, \iota_2]$$

- Fixed as innermost extension (right of $\mathcal{L}$)
- Universal distributive law over any monad:

$$[S\iota_1, \eta^S \circ \iota_2] : \mathcal{C}_A S \Rightarrow S\mathcal{C}_A$$

Nondeterminism

Finite Power Monad

$$\mathcal{P}_{\mathsf{f}}X = \{Y \subseteq X \mid Y \text{ finite}\} \quad \mathcal{P}_{\mathsf{f}}h(Y) = \{f(y) \mid y \in Y\}$$

$$\eta^{\mathcal{P}_{\mathsf{f}}}(x) = \{x\} \quad \mu^{\mathcal{P}_{\mathsf{f}}} = \bigcup$$

- Fixed as outer extension (left of $\mathcal{L}$)
- Distributive law: **Cartesian product**

$$\prod : \mathcal{L}\mathcal{P}_{\mathsf{f}} \Rightarrow \mathcal{P}_{\mathsf{f}}\mathcal{L}$$

Probabilism

Finitely Supported Distribution Monad

$$\mathcal{D}_\mathrm{f}X = \{p : Y \to [0,1] \mid Y \in \mathcal{P}_\mathrm{f}X; \sum_x p(x) = 1\}$$

$$\mathcal{D}_\mathrm{f}h(p)(y) = \sum_x p(x)\,\delta_{h(x),y}$$

$$\eta^{\mathcal{D}_\mathrm{f}}(x)(y) = \delta_{x,y} \qquad \mu^{\mathcal{D}_\mathrm{f}}(p)(y) = \sum_{q,x} p(q)\,q(x)\,\delta_{x,y}$$

- Fixed as outer extension (alternative to $\mathcal{P}_\mathrm{f}$)
- Distributive law: **independent product**

$$\psi : \mathcal{L}\mathcal{D}_\mathrm{f} \Rightarrow \mathcal{D}_\mathrm{f}\mathcal{L}$$

$$\psi(p_1 \cdots p_n)(y_1 \cdots y_n) = \sum_{x_1 \cdots x_n} p_1(x_1) \cdots p_n(x_n)\,\delta_{x_1,y_1} \cdots \delta_{x_n,y_n}$$

- ***No*** mention of stochastic independence in (Prusinkiewicz and Lindenmayer 1990)!

Probabilism

Finitely Supported Distribution Monad

$$\mathcal{D}_\mathsf{f} X = \{p : Y \to [0,1] \mid Y \in \mathcal{P}_\mathsf{f} X; \sum_x p(x) = 1\}$$

$$\mathcal{D}_\mathsf{f} h(p)(y) = \sum_x p(x)\, \delta_{h(x),y}$$

$$\eta^{\mathcal{D}_\mathsf{f}}(x)(y) = \delta_{x,y} \qquad \mu^{\mathcal{D}_\mathsf{f}}(p)(y) = \sum_{q,x} p(q)\, q(x)\, \delta_{x,y}$$

- Fixed as outer extension (alternative to $\mathcal{P}_\mathsf{f}$)
- Distributive law: **independent product**

$$\psi : \mathcal{L}\mathcal{D}_\mathsf{f} \Rightarrow \mathcal{D}_\mathsf{f}\mathcal{L}$$

$$\psi(p_1 \cdots p_n)(y_1 \cdots y_n) = \sum_{x_1 \cdots x_n} p_1(x_1) \cdots p_n(x_n)\, \delta_{x_1,y_1} \cdots \delta_{x_n,y_n}$$

- ***No*** mention of stochastic independence in (Prusinkiewicz and Lindenmayer 1990)!

Parameters

Classical Definition (Syntactic)

- ad-hoc datatypes and expression language
- formal parameters, guards, actual parameters
- long-winded informal description of evaluation

Coproduct-structured Carrier

For each symbol $v \in V$ fix a parameter space A_v

$$V' = \coprod_v A_v$$

- Parametrized L-Systems as coalgebras (V', p)
- Parameter spaces may be infinite
  - restore "essential finiteness" by requiring a **homomorphism** to a finite coalgebra (W, q), respecting coproduct structure
  - optionally rule out guards by requiring $W \cong V$

Parameters

Classical Definition (Syntactic)

- ad-hoc datatypes and expression language
- formal parameters, guards, actual parameters
- long-winded informal description of evaluation

Coproduct-structured Carrier

For each symbol $v \in V$ fix a parameter space A_v

$$V' = \coprod_v A_v$$

- Parametrized L-Systems as coalgebras (V', p)
- Parameter spaces may be infinite
  - restore "essential finiteness" by requiring a **homomorphism** to a finite coalgebra (W, q), respecting coproduct structure
  - optionally rule out guards by requiring $W \cong V$

Outlook

- Lindenmayer Systems vs. grammars and languages
  - build on existing work

    Trace Semantics (Hasuo and Jacobs 2005)
    Weighted Automata (Honkala 2009)
    BDEs, RegExps (Winter, Bonsangue, and Rutten 2011)
- Final coalgebras, bisimulations
  - relationship to fractals
  - "botanic equivalence", turtle graphics equivalence
- Context-sensitive Lindenmayer Systems
  - possibly bialgebraic?
  - analogous to cellular automata
    (Trancón y Widemann and Hauhs 2011)
- Lindenmayer Systems as Model Coalgebras
  - explore didactic potential
  - contributions welcome! wiki?

Take-Home Messages

- Lindenmayer Systems are, in their basic form,
  finite coalgebras of the list monad
- Dynamics by iteration in the Kleisli category
- Extensions interact with the basics by monadic distributive laws
- Further coalgebraic notions likely to be applicable
- Nice intuitive demonstration of coalgebra for non-experts

Bibliography I

Chomsky, N. (1957). *Syntactic Structures*. Mouton & Co.

Hasuo, I. and B. Jacobs (2005). "Context-Free Languages via Coalgebraic Trace Semantics". In: *Algebra and Coalgebra in Computer Science (CALCO 2005)*. Ed. by J. L. Fiadeiro et al. Vol. 3629. Lecture Notes in Computer Science. Springer, pp. 213–231. ISBN: 3-540-28620-9. DOI: 10.1007/11548133_14.

Honkala, J. (2009). "Lindenmayer Systems". In: *Handbook of Weighted Automata*. Ed. by M. Droste, W. Kuich, and H. Vogler. Springer. Chap. 8. ISBN: 978-3-642-01491-8.

Lindenmayer, A. (1968). "Mathematical models for cellular interaction in development". In: *J. Theoret. Biology* 18, pp. 280–315.

Prusinkiewicz, P. and A. Lindenmayer (1990). *The Algorithmic Beauty of Plants*. Springer. ISBN: 978-0387972978.

Bibliography II

Trancón y Widemann, B. and M. Hauhs (2011). "Distributive-Law Semantics for Cellular Automata and Agent-Based Models". In: *Proceedings 4th International Conference on Algebra and Coalgebra (CALCO 2011)*. Ed. by A. Corradini, B. Klin, and C. Cîrstea. Vol. 6859. Lecture Notes in Computer Science. Springer, pp. 344–358. DOI: 10.1007/978-3-642-22944-2_24.

Winter, J., M. M. Bonsangue, and J. Rutten (2011). "Context-Free Languages, Coalgebraically". In: *Proceedings 4th International Conference on Algebra and Coalgebra (CALCO 2011)*. Ed. by A. Corradini, B. Klin, and C. Cîrstea. Vol. 6859. Lecture Notes in Computer Science. Springer, pp. 359–376. DOI: 10.1007/978-3-642-22944-2_25.