

A Mathematical Foundation for Computer Simulation

Reinhard Laubenbacher

Virginia Bioinformatics Institute at Virginia Tech
reinhard@vbi.vt.edu

July 16, 2004

Goal: Contribute to a mathematical foundation for agent-based computer simulation.

SOFTWARE ← THEORY → ANALYSIS

A mathematical specification for simulations can help with:

- software verification;
- simulation design;
- analysis of dynamics;
- relationship between structure and dynamics;
- control theory.

General framework:

Deterministic/stochastic, time-discrete dynamical system

$$f : X \longrightarrow X$$

on a finite set X of possible system states.

General framework:

Deterministic/stochastic, time-discrete dynamical system

$$f : X \longrightarrow X$$

on a finite set X of possible system states.

Examples: Cellular automata, Boolean networks, Probabilistic Boolean networks, Sequential dynamical systems.

Consider only deterministic systems from now on.

General Framework

- Variables x_1, \dots, x_n , which take values in a finite set X ;
- functions f_1, f_2, \dots, f_n such that

$$f_i : X^n \longrightarrow X$$

determines the state of variable x_i (local update functions);

Iteration of

$$f = (f_1, \dots, f_n) : X^n \longrightarrow X^n$$

defines a time-discrete, state-discrete dynamical system.

General Framework

- Variables x_1, \dots, x_n , which take values in a finite set X ;
- functions f_1, f_2, \dots, f_n such that

$$f_i : X^n \longrightarrow X$$

determines the state of variable x_i (local update functions);

Iteration of

$$f = (f_1, \dots, f_n) : X^n \longrightarrow X^n$$

defines a time-discrete, state-discrete dynamical system.

Problem: No tools are available to analyze such systems since there is no mathematical structure.

Possible additional structure:

- Structure on X , e.g., X is a finite field. (Finite version of a Cartesian coordinate system.)

Consequence: the f_i are algebraic functions!

- Asynchronous update of variables. (Get *sequential dynamical systems*).

Assume (1). Change notation: $k := X$, $f_i \in k[x_1, \dots, x_n]$.

Example: Boolean networks.

$X = \{0, 1\} = \mathbf{F}_2$. Then Boolean functions can be represented as polynomial functions:

- $x \wedge y = x * y$;
- $x \vee y = x + y + x * y$;
- $\neg x = x + 1$.

Finite Dynamical Systems.

Let k be a finite field. A *finite dynamical system (FDS)* is a function

$$f = (f_1, \dots, f_n) : k^n \longrightarrow k^n,$$

with dynamics generated by iteration.

Dynamics.

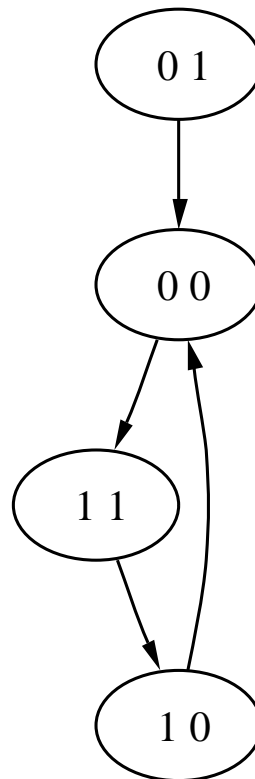
The *state space* of the FDS $f : k^n \longrightarrow k^n$ is the directed graph with vertices the elements of k^n . There is an edge $a \rightarrow b$ iff $f(a) = b$.

Two types of points: *transient* and *periodic* points

Dynamics (cont.).

Example:

$$X = \mathbf{F}_2, f_1 = 1 + x_1 + x_2, f_2 = 1 + x_1 + x_2 + x_1 * x_2$$



Question: Does the structure of the f_i contain information about the dynamics of f ?

Cellular automata: (Wolfram, Odlyzko) For a 1-dim. additive binary CA, all trees at the nodes of the limit cycles are identical.

Boolean networks: (Elspas) The number and length of limit cycles of a linear BN can be determined from the characteristic polynomial of a representing matrix.

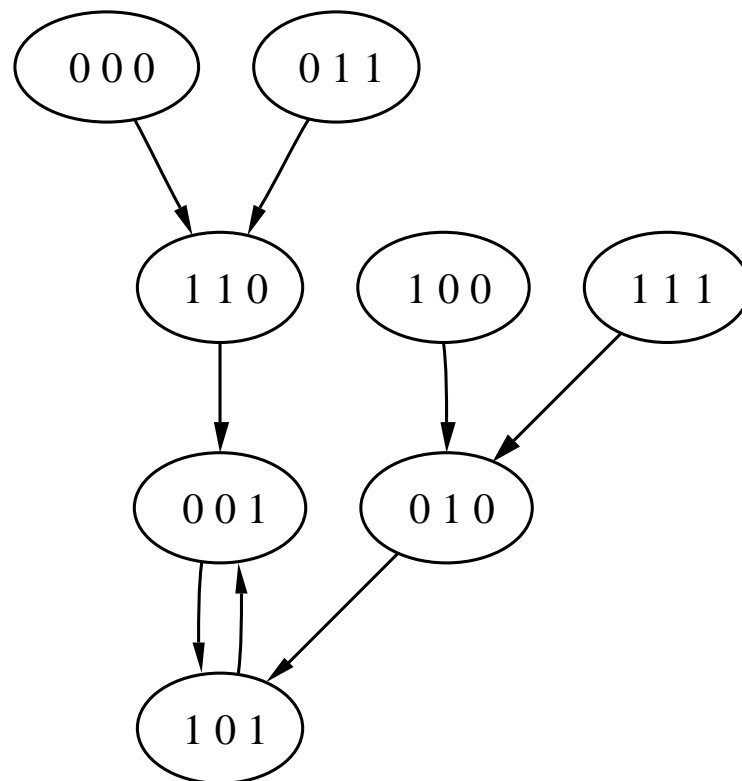
Linear Systems over arbitrary finite fields:(Hernandez) The number of components of the state space, the length of each limit cycle, and the tree structure of the transients can be determined from the factorization of the characteristic polynomial of a representing matrix into irreducible factors, together with the number of field elements. In particular, the structure of the tree of transients at each point in each limit cycle is identical.

(This recovers the Wolfram/Odlyzko result.)

Example.

$$f_1 = 1 + x_1, f_2 = 1 + x_2 + x_3, f_3 = x_2 + x_3$$

$$f = (f_1, f_2, f_3) : \mathbf{F}_2^3 \longrightarrow \mathbf{F}_2^3.$$



Nonlinear systems

Dimension 1: $f : k \longrightarrow k$.

Well-studied question: When is f a permutation polynomial, that is, when is f a bijective function?

Theorem. (Cohen) If p is a sufficiently large prime and $f(x)$ of degree ≥ 2 permutes \mathbf{F}_p , then for all $0 < a < p$, $f(x) + ax$ is not a permutation polynomial.

Monomial systems over \mathbf{F}_2 .

Let $f = (f_1, \dots, f_n) : \mathbf{F}_2^n \longrightarrow \mathbf{F}_2^n$ be such that each $f_i \in k[x_1, \dots, x_n]$ is a (square-free) *monomial*.

Problem: Characterize those monomial systems that are *fixed point systems*, that is, have only limit cycles of length 1.

Monomial systems (cont.).

Associate to f a directed graph X_f , its *dependency graph*.

- Vertices: $a_1, \dots, a_n, \epsilon$;
- Edges: There is an edge $a_i \rightarrow a_j$ if x_j appears in f_i . There is an edge $a_i \rightarrow \epsilon$ if $f_i = 0$.

Monomial systems (cont.).

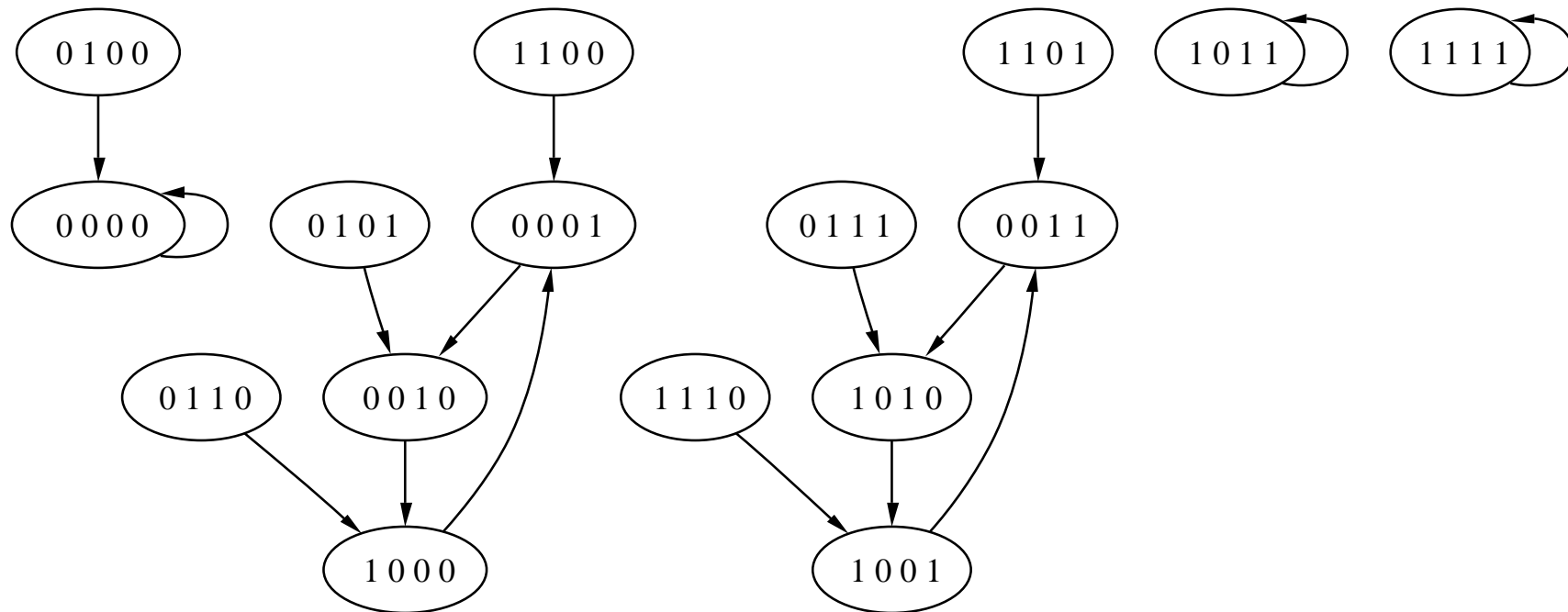
Example.

$$f = (x_3, x_1x_4, x_4, x_1) : \mathbf{F}_2^4 \longrightarrow \mathbf{F}_2^4.$$

Monomial systems (cont.).

Example.

$$f = (x_3, x_1x_4, x_4, x_1) : \mathbf{F}_2^4 \longrightarrow \mathbf{F}_2^4.$$



Monomial systems (cont.).

Let G be a strongly connected directed graph. Define the *loop number* of G as follows. Let v be any vertex in G . Consider all directed paths from v to itself, including the constant path at v , and form pairwise differences of their lengths. The loop number of G is the minimum of all those differences that are nonzero.

This is the same as the gcd of all loop lengths, the *index of imprimitivity* of the graph.

Monomial systems (cont.).

Theorem. (Colòn-Reyes, L., Pareigis) Let $f : \mathbf{F}_2^n \longrightarrow \mathbf{F}_2^n$ be a monomial system with dependency graph G . Then the following are equivalent.

1. f is a fixed point system;
2. for every vertex a in G one of the following holds:
 - (a) the connected component of a has loop number 1,
 - (b) a is connected with a walk to ϵ ,
 - (c) there is no walk of length ≥ 1 from a to itself.

Proof. reduce to the case of a strongly connected dependency graph.

The problem for general finite fields can be reduced to a Boolean and a linear system (Colòn-Reyes, Jarrah, L., Sturmfels).

Monomial systems (cont.).

Definition. Let X and Y be dependency graphs of functions $f : \mathbf{F}_2^r \rightarrow \mathbf{F}_2^r$ and $g : \mathbf{F}_2^s \rightarrow \mathbf{F}_2^s$, respectively. A *glueing* $X \# Y$ of Y to X consists of a digraph with vertices $V_X \dot{\cup} V_Y$ and edges $E_X \dot{\cup} E_Y$ (disjoint union), together with a set of additional directed edges from vertices in Y to vertices in X , and a new set of coordinate functions for g that reflect the new inputs.

Get a new system $f \# g : \mathbf{F}_2^{r+s} \longrightarrow \mathbf{F}_2^{r+s}$.

Monomial systems (cont.).

Theorem. (Colòn-Reyes, L. Pareigis) Let X be a fixed point system and let Y be strongly connected of loop length ≥ 1 . Let $X \# Y$ be a glueing. The following are equivalent:

1. The glueing $X \# Y$ is a fixed point system.
2. (a) There is a vertex $a \in Y$ that is connected with a walk to a zero in X , or
(b) Y is a fixed point system.

Reverse question:

How to design a FDS with specified (partial) state space structure?

This can be solved as an interpolation problem, using tools from computational algebra (L., Stigler, *J. Theor. Biol.*, 2004).

(of interest in biology, e.g., neural networks, gene regulatory networks)

Sequential dynamical systems (SDS)

Update vertices sequentially instead of in parallel.

Let $\pi \in S_n$ be a permutation, and let $f_i : k^n \rightarrow k^n$, $i = 1, \dots, n$, be a function that changes only the i th coordinate. Form

$$f = f_{\pi^{-1}(n)} \circ f_{\pi^{-1}(n-1)} \circ \cdots \circ f_{\pi^{-1}(1)} : k^n \rightarrow k^n.$$

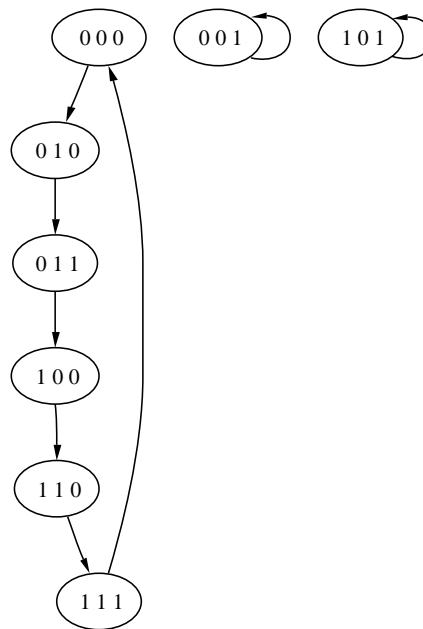
Then f is a finite dynamical system, determined by the f_i and the *update schedule* π .

SDS (cont.)

Example.

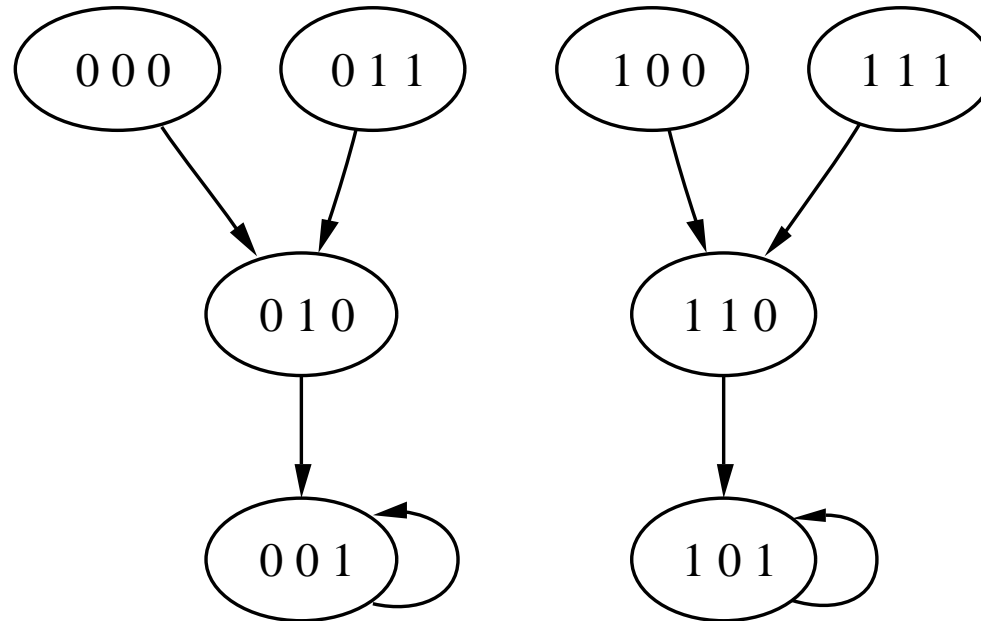
$$f_1 = x_2 * x_3 + x_1, f_2 = x_3 + 1, f_3 = x_3 + x_2.$$

Use parallel update.



SDS (cont.)

Update first x_1 , then x_2 , finally x_3 :



SDS (cont.)

Assume that the f_i are symmetric in their inputs. Associate to the f_i the graph G with vertices a_1, \dots, a_n . There is an (undirected) edge between a_i and a_j if x_i appears in f_j .

SDS (cont.)

Assume that the f_i are symmetric in their inputs. Associate to the f_i the graph G with vertices a_1, \dots, a_n . There is an (undirected) edge between a_i and a_j if x_i appears in f_j .

Theorem. (Barrett, Mortveit, Reidys) The number of different SDS one obtains by only varying the update schedule π is bounded above by the number of acyclic orientations of G . Furthermore, this upper bound is sharp, realized by choosing all f_i to be *NOR*.

SDS (cont.)

Theorem. (Garcia, Jarrah, L.) The same result is true if the update schedule is generalized to be a word in $\{1, \dots, n\}$.

SDS (cont.)

Theorem. (Garcia, Jarrah, L.) The same result is true if the update schedule is generalized to be a word in $\{1, \dots, n\}$.

Can generalize the update schedule to be a partially ordered set (L., Pareigis). Is the theorem still true?

Transformations of FDS and SDS.

GOAL: Want categories SDS , FDS , and state spaces S , with “good” properties, together with functors:

$$SDS \longrightarrow FDS,$$

$$SDS \longrightarrow S,$$

$$FDS \longrightarrow S.$$

A transformation

$$\varphi : (f : k^n \rightarrow k^n) \longrightarrow (g : k^m \rightarrow k^m)$$

is defined as a function $\phi : k^n \rightarrow k^m$ such that

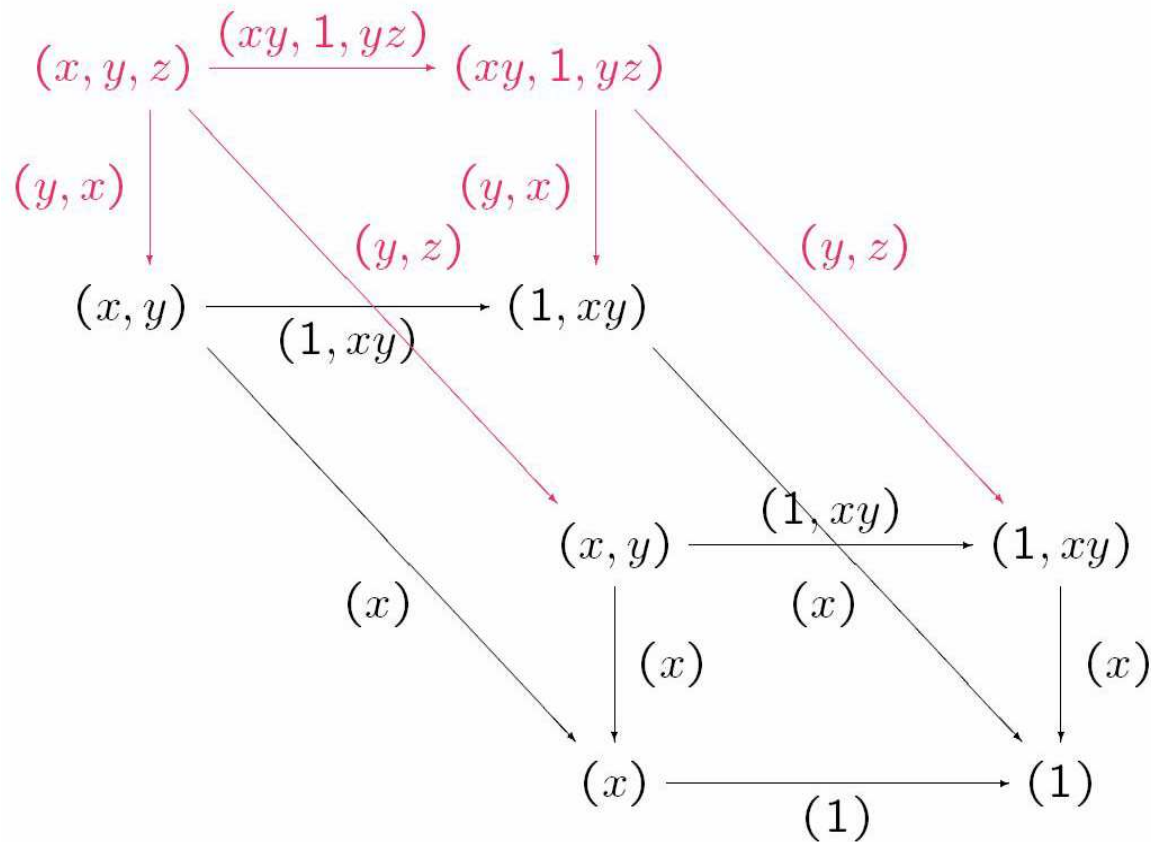
- ϕ is constructed from injections, projections, and “scalar multiplications;”
- $\phi \circ f = g \circ \phi$, locally (hence globally).

A rigorous definition involves the dependency graphs of f and g .

Theorem. (L., Pareigis) Any such transformation induces a transformation of state spaces. Furthermore, the composition of transformations is again a transformation.

Fiber products

Example.



An Application

Goal: Construct a dynamic model for PathSim (stochastic, agent-based model) and use it to solve optimal control problems.

Strategy:

- Generate time series from the simulation. Use them to average stochastic effects. Generate perturbation time series.
- Apply system identification algorithm in (L., Stigler, *J. Theor. Biol.*, 2004) to get polynomial model.
- Then apply optimal control theory methods for polynomial systems over finite fields, developed by LeBorgne, Marchand, and others. Uses tools from computational algebraic geometry.

Problem: System identification algorithm cannot deal with thousands of variables. Need dimension reduction techniques.

Solution: Apply the algorithm locally and “glue together” local models via fiber products.

Need to understand how fiber products behave with respect to dynamics.

Open Problems:

- binomial systems;
- can one use tools from computational algebra to, e.g., compute fixed points of special types of systems, such as special functions or special dependency graphs. (The general problem is NP hard.);
- understand transformations of SDS, FDS and related categorical constructions;
- understand the role of the update schedule and the functor $SDS \longrightarrow FDS$.

Software for simulation of FDS and SDS:

`http://dvd.vbi.vt.edu`

Summary

- A mathematical foundation is needed to deal with agent-based simulations.
- Pieces of such a foundation already exist.
- A collaboration between math/cs and applications is required.
- Lots of interesting math problems.
- Lots of interesting applications.

Collaborators and Students:

Chris Barrett (LANL), Omar Colòn-Reyes (VT), Nick Eriksson (UC Berkeley), Luis Garcia (VT), Abdul Jarrah (VBI), Paola Vera Licona (VT), Madhav Marathe (LANL), Henning Mortveit (LANL), Bodo Pareigis (Univ of Munich), Christian Reidys (LANL), Brandilyn Stigler (VT), Michael Stillman (Cornell), Bernd Sturmfels (UC Berkeley), Hussein Vastani (VT).

Financial support: NSF, NIH, Los Alamos National Laboratory.