Finite Dynamical Systems

Yves Jäckle

2023

Contents

1 Finite Dynamical Systems

1.1 Basics of FDS

Finite Dynamical Systems:

A **finite dynamical system (FDS)** is described by its **state space** X, which is a finite set, and its **transition function** $f : X \to X$.

In a **synchronous FDS**, the dynamics is described by the images $f^n(X)$ (image of the n-th composite of f). The **trajectory** of $x \in X$ is the sequence $(f^n(x))_{n\in\mathbb{N}}$. We associate a **synchronous transition (di-) graph** $STG(f)$ to such a system, who's vertices are $V = X$ and who's arc are $A = \{(x, y) : y = f(x)\}$. This digraph may therefore contain loops, corresponding to the case $f(x) = x$.

In many cases, one deals with $X = \prod_{i=1}^{[d]} X_i$ for finite sets X_i. A very common such type of systems are those for which $X_i = \{0, 1\}$, which are called **Boolean networks**.

The case where $X = \{0, 1\}^L$ where $L \subsetneq \mathbb{Z}^d$ can represent a certain type of **cellular automata**. We can picture these as a part of a grid/lattice in d-space, in which grid-cells may be in one of two states. The transitions f_i for $i \in L$ may then depend on a neighbourhood of i in L, for example the transitions can have expression $f_i(x) = g(x_{B_1(x,1)})$ where $B_1(x, 1) = \{y \in L : \|x - y\|_1 \leqslant 1\}$.

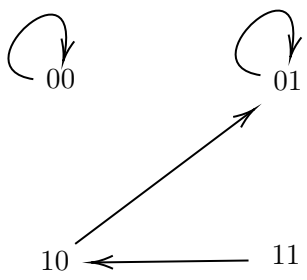
There is a different type of update for FDS:

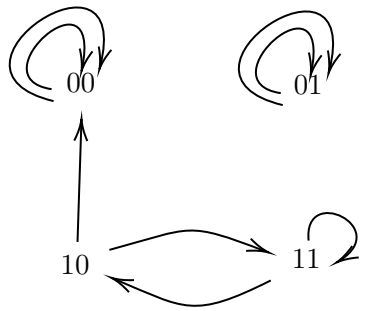
Asynchronous FDS:

In a **asynchronous FDS**, the state space is of form $X = \prod_{i=1}^{[d]} X_i$ for finite sets X_i, and we're interested in component-wise updates. By writing $x = (x_1, ..., x_d)$ with $x_i \in X_i$, the transition functions are $\tilde{f}_i(x) = (x_1, ..., x_{i-1}, f_i(x), x_{i+1}, ..., x_d)$. The dynamics is then described by the **asynchronous transition (di-)graph** $ATG(f)$: its vertices are $V = X$ and its arcs are $A = \left\{(x, y) : y = \tilde{f}_i(x)\right\}$. Such a graph may have loops, and even loops as multi-edges.

One can think of $ATG(f)$ as representing all the transitions that could occur, without information on which component will get updated in what order.

Examples:
We consider the boolean network with state space $X = \{0, 1\}^2$ and transition function f where f_1 is bit-conjunction "and" and f_2 is bit-exlcuive-disjunction "xor". On the left of the following figure, we depicted $STG(f)$, and on the right, we have $ATG(f)$.





One can study an asynchronous FDS with additional information on the order in which the transitions occur:

Sequential dynamical system:

A **sequential dynamical system (SDS)**, is a synchronous FDS based on an **update order/schedule** described by a permutation π of $[|X|]$. The transition function of the SDS is $F_\pi = \tilde{f}_{\pi(|X|)} \circ ... \circ \tilde{f}_{\pi(1)}$. Note that in general, $F_\pi \neq f$.

As with normal dynamical systems, we're interested in the long term behaviour of the system.

Fixed points and attractors:

For an FDS, we say that a state $x \in X$ is **periodic** if there is a iterate f^n with $n \geqslant 1$ so that $f^n(x) = x$. The smallest such n is the **period** of x. Points of period 1 are called **fixed points** or **steady states**. we denote their set by $Fix(f)$.

A set of states A is an **attractor** of the FDS if in the respective transition graph G, there are no arcs from a vertex of A to a vertex of $X \backslash A$, and A is inclusion-minimal for this property.

Characterisation of attractors in synchronous FDS:

There are always attractors of a synchronous FDS and they are made of periodic points and are precisely the cycles (possibly loops) of its transition graph $STG(f)$.

Proof: If x is on a cycle of $STG(f)$, then $(f^n(x))$ runs over the cycle, and x is periodic (with period the length of the cycle). The arcs of the cycle connect only cycle-vertices and this property fails if we remove at least one vertex of the cycle (as the arc leading to one of those vertices no leads outside of A). Therefore cycles, are attractors.

Conversely, if A is an attractor of a synchronous FDS, then for all $x \in A$, $f(x) \in A$. As A is finite, there must be a repetition in the sequence $(f^n(x))$ after a finite number of iterates. We consider the first such repetition at states $f^a(x) = f^b(x)$: this means that $y := f^a(x)$ is on a cycle of $STG(f)$. If $y \neq x$, then A wouldn't be an attractor, as the cycle on y that is its subset has the attractor property, so that A wouldn't be inclusion-minimal. So A contains a cycle in x, implying that x is periodic, and since A is inclusion-minimal, it must in fact be that cycle.
Finally, for $A = X$, the beginning of the argument implies that a attractor must exist.

Remark: We can use indicator vector $x \in \{0,1\}^{|X|}$ to represent the states. Transitions can then be handled with matrix M_f who'se entries are $m_{y,x} = \begin{cases} 1 : y = f(x) \\ 0 : else \end{cases}$, so that $M_f x$ indicates the image of the state that x indicates, under f. One can then study the dynamics with linear algebra methods, such as the Jordan normal form, or simply Gaussian elimination, as one can find fixed points by solving $M_f x = x$. However, when $|X|$ is large this may become a computationally difficult task. Also, computing M_f implicitly requires computing the transition graph, so that solving $M_f x = x$ is a inefficient way of finding fixed points.
Note that this method links FDS to the field of Markov chains, as we're dealing with a Markov chain with deterministic transition probabilities.

1.2 Applications in Biology

1.3 Dynamics from the interaction graph

In biological applications, the transition function f may be unknown. However, certain experiments can provide information on f. One such information is **dependency**: in the case of a state space $X = \prod_{i=1}^{d} X_i$, we say that dimension (/criteria/parameter) $j \in [d]$ is dependent on $i \in [d]$ if we can find states $x, y \in X$ that differ only in i (in the sense that $x_k = y_k$ for $k \in [d] \backslash i$ and $x_i \neq y_i$), so that $f_j(x) \neq f_j(y)$. Independence therefore means that we can change parameter i of the biological system and expect no changes to parameter j in the next step of the biological system. Dependence means that a change parameter i may (it doesn't have to) yield a change to parameter j in the next step of the biological system. We can visualize this with:

Interaction graph of an FDS:

The **interaction (di-)graph** $IG(f)$ of an FDS on state space $X = \prod_{i=1}^{d} X_i$ is a digraph with vertex set $V = [d]$ and arcs $A = \{(i,j) : \exists x, y \in X, x_k = y_k \; for \; k \in [d] \backslash i \; and \; x_i \neq y_i, \; and \; f_j(x) \neq f_j(y)\}$, meaning that we add an arc if j depends on i. This graph may have loops.

Vocabulary: in the literature, another name for the interaction graph is the dependency graph, and it appears under different definitions, which may or not be equivalent.

As one can obtain an interaction graph from experiments, we'll study what can be said about the FDS from this graph only, ignoring the true transitions function f.

This task can seem rather difficult, as there may be many different FDS with different behaviours that have the same interaction graph. For example, we can consider all the Boolean networks who's interaction graph is a cycle C_n. If we peak ahead in these notes to the characterisation of the interaction graphs for PDS, we'll see that this means that the transition functions have form $f_i(x_{i-1 \mod n})$ and aren't constant, so that the only possible transitions are $f_i(x_{i-1 \mod n}) = x_{i-1 \mod n}$ or $f_i(x_{i-1 \mod n}) = 1 - x_{i-1 \mod n}$. This represents one of two choices per i, so that 2^n transitions f exist that have C_n as interaction graph.

However, one can still make general qualitative statements based on the interaction graph:

Robert's Theorem:

For an FDS with transition function f and state space $X = \prod_{i=1}^{d} X_i$, then the interaction graph $IG(f)$ is acyclic (also, loopless) $\Rightarrow f^n$ is the same constant function for all $n \geqslant d$. In such a case, f has a unique fixed point and it's the only attractor.

Proof: We'll prove this by induction on d. The case $d = 1$ corresponds to a constant $f = f^1$ as no changes to x_1 affect f.
For dimension d, the acyclic $IG(f)$ must have a source: a vertex i to which no arc points (consider one of the endpoints of a largest dipath in the graph). This means that f_i is a constant k_i, as it takes the same value for all states. To see this, note that we can apply the notion of independence to any sequence of states $x = c_0, c_1, ..., c_{n-1}, c_n = y$ where c_{j+1} differs from c_j only by coordinate $j+1$, where $c_{j,j+1} = x_{j+1}$

and $c_{j+1,j+1} = y_{j+1}$, to see that $f_i(x) = f_i(y)$.
This now implies that in the sequence of states $x^{(n)} = f^n\left(x^{(0)}\right)$, we have $x_i^{(n)} = k_i$ for $n \geqslant 1$. Next, by defining $g(x_{[d]\setminus i}) = f(x_1, ..., k_i, ..., x_d)$, we see that for $n \geqslant 2$, we have $x^{(n)} = f^n\left(x^{(0)}\right) = g^{n-1}\left(x^{(1)}_{[d]\setminus i}\right)$. If we then restrain the image of g to the dimensions of $[d]\setminus i$, knowing that $g_i = f_i = k_i$, then g is the transition function of a FDS on a space of dimension $d-1$. the interaction graph of g is a subgraph of that of f, so that it remains acyclic. We can then use the induction hypothesis that g^n is constant for all $n \geqslant d-1$, but then $f^n\left(x^{(0)}\right) = g^{n-1}\left(x^{(1)}_{[d]\setminus i}\right)$ is constant for all $n \geqslant d$, which is the conclusion of the induction step. (A. Richard's notes may have a cleaner version of this proof).
Finally we show that in such a case the constant $k = f^d$ is a fixed point, and it's the only attractor of f. Since $f^{d+1}(k) = k$ and $f^{d+1}(k) = f\left(f^d(k)\right) = f(k)$, we see that k is a fixed point. It's the only fixed point, as for any other fixed points q, $q = f^d(q) = k$. To see that cyclic attractors are impossible, note that the cycle would get stuck at k from d iterations onward.

We'll see later on that there is a pseudo-converse to Roberts theorem, in the sense that for graphs with a special property which includes having a cycle, one can build Boolean networks the have none or multiple fixed points that have this graph as interaction graph.

2 Boolean Networks

2.1 Interaction graphs and BN

Here is the pseudo-converse to Roberts theorem we promised:

Small Aracena-Richard-Salinas Theorem:

For a digraph G on vertices $[d]$ that has the property $|\delta^-(j)| \geqslant 1$ for all $i \in V$ and posses a cycle, there is a BN that has no fixed points, and one that has at least two fixed points, so that both have G as interaction graph.

Proof: First, we build a BN that has at least 2 fixed points. We'll study the BN given by $f_j(x) = \bigvee_{i \in \delta^-(j)} x_i$ which has already the fixed points 0 and 1. Note that the interaction graph of this BN is indeed the one we assumed: j does depend on i when $i \in \delta^-(j)$, as one can see by computing the images of 0 and e_i under f_j and noting that they differ. Other dependecies don't exist, as the variables don't appear in the functions expression.
If we only have the property that the graph has a cycle and we set the disjunction over an empty set to be 0, then 0 might not be a fixed point anymore if the graph has a source.
But we can still obtain a fixed point in that case ! Indeed, we only need to set $x_i = 0$ when i can not be reached from the cycle C (this includes the potential sources), and $x_i = 1$ when i can be reached from the cycle C. This will produce a dicut in the graph, as there may be no arcs from a vertex reachable from C to one that isn't. The vertices reachable from C, j, all have a vertex i of the same partition as predecessor: this is true for the vertices on the cycle (take their predecessor on the cycle), and then for those at the end of a path starting from a cycle vertex. Since the variable corresponding to that predecessor i has value 1, $f_j(x) = 1 = x_j$.
The other vertices j can have no predecessor (if any) reachable from C, so that all their predecessors i (if any) have value 1, so that $f_j(x) = 0 = x_j$. We therefore get a fixed point which is different from 1 when there's at least one source.

We'll now build the BN that has no fixed points. Our construction will have the desired property if a certain set underlying it is an FVS:

Feedback vertex set:

A **feedback vertex set (FVS)** I in a graph (V, A) is a set of vertices I so that each cycle of (V, A) contains a vertex of I. In particular, the induced graph on $V \backslash I$ is acyclic. One can require that I be inclusion minimal for this property in the definition, which we do.

When the graph has a cycle C, FVSs exist and aren't empty, as they must contain an element of C.
We'll study the BN given by $f_j(x) = \bigvee_{i \in \delta^-(j)} \overline{x_i}$ for $j \in F$ and $f_j(x) = \bigwedge_{i \in \delta^-(j)} x_i$ for $j \notin F$ and see that it has no fixed point when F is a FVS of the graph.
Note again that the interaction graph of this BN is indeed the one we assumed: j does depend on i when $i \in \delta^-(j)$, as one can see by computing the images of 1 and $1 - e_i$ under f_j and noting that they differ. Other dependencies don't exist, as the variables don't appear in the functions expression.
We'll show this by contradiction: assume that we have a fixed point $x = f(x)$. We now disjoin cases on

whether there is an $i \in F$ so that $x_i = 0$ or not. In the first case, we consider a cycle C of the graph so that $F \cap C = \{i\}$: it exists as otherwise, all cycles of the graph passing through i would pass through another point of F, so that $F \backslash i$ has the FVS property, contradicting the inclusion-minimality of F. If we then follow the vertices along the cycle, starting from the successor j to i, since $j \notin F$, we compute $f_j(x) = 0$ due to $x_i = 0$ and $i \in \delta^-(j)$. For the next vertices $k \notin F$, we compute $f_k(x) = 0$ due to $x_j = 0$ and $j \in \delta^-(k)$, until we arrive back to i. There, we compute $f_i(x) = 1$ due to it's predecessor k on the cycle having $x_k = 0$ and $i \in F$. This is a contradiction to the fixed point property $f_i(x) = x_i$.
In the second case of the case disjunction, all $i \in F$ verify $x_i = 1$. Then with $f_i(x) = x_i$ and $i \in F$, we find that there must be a $j \in \delta^-(i)$ so that $x_j = 0$, so that in particular $j \notin F$ in this case. Now $f_j(x) = x_j$ and $j \notin F$ imply that there must exist a $k \in \delta^-(j)$ so that that $x_k = 0$, so that in particular $k \notin F$. We can keep reiterating this argument, backtracking a path of vertices in $V \backslash F$ for which $x_j = 0$ in the process. Due to finiteness, the path will eventually loop: we will then have found a cycle in $V \backslash F$ contradicting the FVS property of F.
So all cases for fixed points lead to contradiction, so that none may exist.

In fact, the number of fixed points of BN with a given interaction graph is a topic of interest:

Fixed points under interaction graph constraints:

We define bounds on the number of fixed points of a BN with interaction graph G by:
$\min_{BN}(G) = \min(|Fix(f)| : IG(f) = G,\ f\ a\ BN) \geqslant 0$ and
$\max_{BN}(G) = \max(|Fix(f)| : IG(f) = G,\ f\ a\ BN) \leqslant 2^{|V|}$.
Of course, $\min_{BN}(G) \leqslant \max_{BN}(G)$.

We mention bounds on these quantities based on the following graph properties:

Relevant graph properties:

The **packing number** $\nu(G)$ of G is the maximum size of a family of vertex-disjoint cycles of G.

The **transversal number** $\tau(G)$ of G, defined as the minimum size of a FVS of G.

INTERNAL NOTE: FIND ALGORITHMS FOR COMPUTING THEM
With them, one can bound the maximum number of fixed points:

Aracena-Richard-Salinas Theorem:

$$\max_{BN}(G) \geqslant \nu(G) + 1$$

Feedback bound Theorem (Riis-Aracena):

$$\max_{BN}(G) \leqslant 2^{\tau(G)}$$

2.2 Algorithms for finding attractors of BN

Complexity for fixed points:

Finding a/the fixed point(/s) of a BN is NP-complete.

Proof: We reduce a SAT instance to the task of finding a fixed point of a particular BN.
For a SAT instance on variables $x_1, ..., x_n$ and clauses $C_1, ..., C_m$, we build a BN on $\{0,1\}^{n+m}$ where we keep variables $x_1, ..., x_n$ associate a variable y_i to clause C_i. The key remark is that in a Boolean equation of form $z = E(...) \vee \overline{z}$, we must have $z = 1$, so that $E(...) = 1$ as well. Since we want the clauses to be true, we can let $E(...) = C_i$ and by letting z be a variable in our BN, $z = E(...) \vee \overline{z}$ can be related to fixed points by defining transition $f_i(x_1, ..., x_n, y_1, ..., y_m) = C_i(x_1, ..., x_n) \vee \overline{y_i}$ where i indexes clauses. By setting transitions $f_j(x_1, ..., x_n) = x_j$ for j indexing variables, fixed points of the BN solve $x_j = x_j$ and $y_i = C_i(x_1, ..., x_n) \vee \overline{y_i}$, the second part implying that the x-variables of that point satisfy all clauses. Conversely, any assignment of x-variables satisfying the clauses can be extended to a fixed point of the BN by setting the remaining y-variables to 1. Since the number of variables and functions of the BN we built is polynomial in the input size of SAT, this is a polynomial time reduction to an NP-complete problem. Finally, one can verify that a given point is a fixed point by computing its image under f, so that we obtain NP-completeness of the problem.

Remark: If the f_i are Boolean functions, one can reduce the problem of finding fixed points to a SAT. First, we observe that $x_i = f_i(x)$ is equivalent to $1 = \overline{f_i(x) \; xor \; x_i}$. Next, one first expresses the Boolean functions $f_i(x) \; xor \; x_i$ in their conjunctive normal form (CNF) CNF_i. This step is where the reduction may fail to be polynomial, as we could obtain a CNF with exponentially clauses. We then see that all equations are satisfied when the CNF $CNF_1 \wedge ... \wedge CNF_n$ is satisfied, that is to say when all clauses are satisfied, which is a SAT instance.
Seeing as it's relatively easy to pass from searching fixed points to solving an SAT instance, we'll briefly discuss how SAT can be solved with integer programming, as it's easier to find IP-solvers then to find implementations of efficient exponential exact algorithms for SAT.
We can associate indicator variables to disjunctions and conjunctions as follows.
If z indicates disjuction $z = \bigvee_{i \in I} x_i$, then we can represent it by constraints $\begin{cases} z \geqslant x_i \\ z \leqslant \sum_{i \in I} x_i \\ z \in \{0,1\} \end{cases}$ and if it indicates conjunction $z = \bigwedge_{i \in I} x_i$, then we can represent it by $\begin{cases} z \leqslant x_i \\ z \geqslant 1 - |I| + \sum_{i \in I} x_i \\ z \in \{0,1\} \end{cases}$.
Finally, negation can be expressed as $\overline{x_i} = 1 - x_i$, so that by nesting these variables, we'll end up with variables z_j indicating clause j of m clauses of the SAT instance. We can then solve the IP $\max \left(\sum_{j=1}^{m} z_j \right)$ over these constraints. If the optimum returned by the IP solver is m, then the corresponding feasible solutions x-variables correspond to a satisfying assignment, and otherwise, no satisfying assignment exists.

3 Sequential Dynamical Systems

"An introduction to Sequential Dynamical Systems" by Henning Mortveit and Christian Reidys

4 Polynomial Dynamical Systems

4.1 Basics on PDS

A particular type of FDS are:

Polynomial Dynamical Systems:

A **polynomial dynamical systems (PDS)** is a synchronous FDS for which the state space is K^d where K is a finite field and for the transition function f, the components f_i are a polynomials of $K[X]$.

These FDS are of interest from a computational viewpoint. For example, finding their fixed points is equivalent to solving a system of polynomial equations over the field.

Indeed, if $x \in K^d$ is a fixed point, this mean that it solves $\begin{cases} f_1(x) - x_1 = 0 \\ ... \\ f_d(x) - x_d = 0 \end{cases}$ over K^d.

If the state space is large, computing the transition graph becomes extremely time consuming compared to the task of solving systems of polynomial equations. This the strength of PDS. Therefore, we'd like to know when we can use them:

Representing FDS as PDS:

An FDS can be represented by a PDS if it's state space has the size p^q where p is a prime.

We remark that this is the case for Boolean networks. There reason for the constraint on the size is that the only finite fields that exist have size (/order) of this form. They can be represented by $(\mathbb{Z}/p\mathbb{Z})[X]/\langle P \rangle$ for some monic irreductible P over $\mathbb{Z}/p\mathbb{Z}$.
As a reminder, $\mathbb{Z}/p\mathbb{Z}$ are the integers modulo p and $(\mathbb{Z}/p\mathbb{Z})[X]/\langle P \rangle$ are the polynomials over $\mathbb{Z}/p\mathbb{Z}$ modulo P. As we'll consider polynomials over the field $(\mathbb{Z}/p\mathbb{Z})[X]/\langle P \rangle$, it's convenient to not think of its elements as polynomials, but as "symbolic expressions" of form $\sum_{i=0}^{\deg(P)-1} c_i X^i$ for an indeterminate X.
We can see by this expression that the field has $p^{\deg(P)}$ elements. So the K^d we hope to represent the state space with has size $p^{\deg(P).d}$, hence the necessity of the condition. To be more precise, it can be shown that one can find a desired polynomial of arbitrary degree, so that *all* prime-power-sized state spaces are candidates
FLESH OUT THE ACTUAL ALGEBRA ON THIS PREVIOUS PART.

So we know that one can represent the state space as K^d where K is a finite field, when it's size is the power of a prime. How do we represent the transition function ?

Lagrange interpolation:

For any map $f_i : K^d \to K$, there is a polynomial P_{f_i} of degree at most $d(|K| - 1)$ that coincides with f_i, so that the transition function can be represented by $(P_{f_1}, ..., P_{f_d})$.

Proof: We'll construct such a polynomial recursively (inductively), depending on the number of variables d. When $d = 1$, we can use 1-dimensional Lagrange interpolation and obtain $P_g(X) = \sum_{x \in K} g(x) \frac{\prod_{y \in K \setminus x}(X - y)}{\prod_{y \in K \setminus x}(x - y)}$ which has dregee $|K| - 1$ at most.
For the step, we use the idea of factorisation into the coefficients. We consider the $|K|$ functions $y \mapsto g(y_1, ..., y_d, x_{d+1})$, indexed by x_{d+1} ranging over K. By induction, we know that there are polynomials $P_{x_{d+1}}(y_1, ..., y_d)$ interpolating $g(y_1, ..., y_d, x_{d+1})$.
We can then define $P_g(y_1, ..., y_d, y_{d+1}) = \sum_{x_{d+1} \in K} P_{x_{d+1}}(y_1, ..., y_d) \frac{\prod_{z \in K \setminus x_{d+1}}(y_{d+1} - z)}{\prod_{z \in K \setminus x_{d+1}}(x_{d+1} - z)}$.
It interpolates g, as in $(x_1, ..., x_d, x_{d+1})$, the only term remaining is $P_{x_{d+1}}(x_1, ..., x_d)$, which interpolated $y \mapsto g(y_1, ..., y_d, x_{d+1})$ by induction and has value $g(x_1, ..., x_d, x_{d+1})$. Finally, the degree of P_g has increased by at most $|K| - 1$, so that it's degree is $(d + 1)(|K| - 1)$ at most.

We still have to explain how one obtains f_i from $f : X \to X$. If $\sigma : X \to K^d$ is the one-to-one map used for representation. then $f_i = \left(\sigma \circ f \circ \sigma^{-1}\right)_i$.
To conclude with the representation theorem, we still have to use the isomorphism $\phi : K^d \to F$ from the product of fields to the field F, mapping coordinate to expression (polynomial) coefficients. The transition function is then $\phi \circ (P_{f_1}, ..., P_{f_d}) \circ \phi^{-1}$.

If $X = \prod_{i=1}^{d} X_i$ where all $|X_i|$ are the same prime and f is given by it's coordinate functions, then it seems better adapted to the problem to consider the representation in K^d instead of that of F.
For Boolean networks, for example, we keep the represent $\{0, 1\}^d$ by rather then by $\mathbb{Z}_2^d$ expressions of form $\sum_{i=0}^{d-1} c_i b^i$ where $c_i \in \mathbb{Z}_2$ and $b^d = b$. To find a polynomial transition function corresponding to the Boolean one, one can use a nested application of tricks, some of which we present:

- Negation: represent $\neg x$ as $1 - x$.
- Conjunction: represent $x \wedge y$ as xy.
- Disjunction: represent $x \vee y$ as $1 - (1 - x)(1 - y) = x + y - xy$.
- Implication: represent $x \Rightarrow y$ as $1 - x(1 - y) = 1 + xy - x$.

For example $f(x, y) = (x \wedge \neg y, x)$ will be represented by $f_P(x, y) = (x(1 - y), x)$. The transition function on the field F associated to the Boolean network would have been $c_0 + bc_1 \mapsto c_0(1 - c_1) + bc_0$.

Remark: For the finite field F and the FDS given by $f : F \to F$, a theoretically interesting result is that f can be represented by a polynomial. Indeed, the Lagrange polynomial interpolating $f(F)$ does the job.

Remark: One can "extend" the representation theorem to FDS with arbitrary state space size by adding artificial states to it. By considering a state space $Y = X \cup Z$ of size a prime power greater then $|X|$, with artificial sates Z (so $Z \cap X \neq 0$), and extending the transition function $f : X \to X$ to $F : Y \to Y$ by setting $F|_X = f$ and $F|_Z = id_Z$, we get a FDS that can be represented by a PDS. Then, the study of the FDS given by f can be acheived through that of the PDS representataive of F, followed by the exclusion of results involving Z.

Note that the representation theorem is only for theoretical purposes. Interpolation requires us to know transition function and to compute all it's values, that is to say that we have to compute the entire transition graph of the FDS.

4.2 PDS and interaction graphs

Is there an algebraic way of describing the interaction graph of a PDS ?
IS THERE AN ACTUAL ALEGRAIC WAY OF TESTING DEPENDENCE ?

Characterisation of the interaction graph of a PDS:

We call the **support** of f the inclusion minimal set $supp(f_j) \subseteq [d]$ so that f_j coincides with a polynomial with variables indexed by $supp(f_j)$. The arcs of the interaction graph of a PDS are $A = \{(i,j) : i \in supp(f_j)\}$.

Proof: We'll show that $i \notin supp(f_j) \Leftrightarrow \forall x, y \in K^d$, $x_k = y_k$ for $k \in [d]\backslash i$ and $x_i \neq y_i$ imply $f_j(x) = f_j(y)$. Contraposition then yields the equivalence of the notions of dependence.
If $i \notin supp(f_j)$, f_j's value stays the same if only the i-th variable is changed, as can be seen from evaluations of its polynomial equivalent in the variables indexed by $supp(f_j)$. Conversely, if $\forall x, y \in K^d$, $x_k = y_k$ for $k \in [d]\backslash i$ and $x_i \neq y_i$ imply $f_j(x) = f_j(y)$, then we can consider the Lagrange polynomial on variables indexed by $[d]\backslash i$ interpolating the points of K^{d-1} with values $f_j(x)$: this polynomial will coincide with f, so that $supp(f_j) \subseteq [d]\backslash i$, meaning that $i \notin supp(f_j)$.

Note that the method used in the proof to tell if j depends on i is computationally inefficient: using Lagrange interpolation to check if $i \in supp(f_j)$ required computing all values of f_j, which already provides an answer to the question of dependency.
Note also that the condition $i \in supp(f_j)$ can in all generality not be read off from f_j. For example, for $f_j(x_i) = x_i^{|K|} - x_i$, j remains independent of i, as f_j coincides with the zero polynomial over the field K.

What kind of interaction graphs can we expect to see from PDS ?

Possibilities of the interaction graph of a PDS:

All digraphs on d vertices may be interactions graph of a PDS.

Proof: For a graph $([d], A)$, we consider the PDS over $\{0,1\}^d$ with transition functions $f_j(x) = \sum_{i \in \delta^-(j)} x_i$, where $\delta^-(j)$ denotes the in-neighbourhood of i (recall that the sum over an empty set is taken to be 0). When $(i,j) \in A$, then the points 0 and e_i differ on i only and have different inmages under f_j so that j depends on i. Otherwise, $(i,j) \notin A$ so that $i \notin supp(f_j)$, and j is independent of i. So $([d], A)$ really is the interaction graph of f.

Note we could have used a different field then $\mathbb{Z}/2\mathbb{Z}$ in the proof and got the same result. Even when the field is fixed, their may be multiple functions with the same interaction graph.

4.3 Reverse engineering and data fitting

4.4 Linear and affine PDS

Linear and affine PDS:

A **linear PDS** is given by a trasition $f : K^d \to K^d$ for a finite field K of form $f(x) = Mx$ where $M \in K^{d\times d}$ is a matrix with coefficients in K.
An **affine PDS** is given by a trasition $f : K^d \to K^d$ for a finite field K of form $f(x) = Ax + b$ where $A \in K^{d\times d}$ and $b \in K^d$.

Under certain conditions, on can use strong tools from linear algebra to describe linear and affine PDS completely. Indeed, we can determine the characteristic polynomial of M or A, use a factorisation algorithm to check if it factors into linear terms, in which case we can use the Jordan canonical/normal form of M or A.
The powers Jordan normal form J can be expressed as formulas of the power. This way, $f^n(x) = M^n x = P^{-1}J^nPx$ and $f^n(x) = P^{-1}J^nPx + P^{-1}\left(\sum_{i=0}^{n-1} J^i\right)Pb$ can be expressed as explicit formulas in n. We can then find the fixed points and cycles of the PDS by solving $f^n(x) = x$ for $n = 1, ..., |K|^d$ with linear algebra. For this approach to be efficient, the time of finding the Jordan form has to outpeform matric multiplication.

4.5 Monomial PDS

Monomial PDS:

A **monomial PDS** is given by a trasition $f : K^d \to K^d$ for a finite field K of form $f_i(x) = c_i x^{a_i}$ for $i \in [d]$, where $a_i \in \mathbb{N}^d$ and $c_i \in K$, using multiindex notation.

If one of the $c_i = 0$, then in the first iteration $x_i^{(1)} = 0$ and in the second, for all j for which $a_{ji} \geqslant 1$, we have $x_i^{(2)} = 0$, and so on. In fact, after a finite number of steps, all x_j for j in the connected component of the interaction/dependency graph of i will stay at value 0. So we'll assume that $c_i \neq 0$, or focus on the sub-PDS for which this is the case.

We can reduce the study of monomial PDS to that of affine PDS as follows.
We find a generator g of K^* by using a finding a prime factorisation of $|K^*|$ and using algorithm 4.80 of $[HAC96]$, recalling that K^* is a cyclic multiplicative group. With this generator, we can use the discrete logarithm $\log_g : K^* \to \mathbb{Z}_{|K^*|-1}$ for which $k = g^{\log_g(k)}$ for all $k \in K^*$, which can be computed using the Pohlig-Hellman algorithm or the baby-step giant-step algorithm.

Then we have $\log_g(f_i(x)) = \log_g(c_i) + \sum_{j\in[d]} \overline{a_{ij}} \log_g(x_j)$ for all $x \in K^d$ and since
$\log_g(f_i(f(x))) = \log_g\left(c_i f_1(x)^{a_{i1}} ... f_d(x)^{a_{id}}\right) = \log_g(c_i) + \sum_{j\in[d]} a_{ij} \log_g(f_j(x))$, we can seen that for matric A with coefficients $\overline{a_{ij}} \in \mathbb{Z}_{|K^*|-1}$, we have $\log_g(f(x)) = \log_g(c) + A \log_g(x)$ and hence $\log_g\left(f^n(x)\right) = \log_g(c) \left(\sum_{i=0}^{n-1} A^i\right) + A^n \log_g(x)$. So we've reduced the study to that of an affine system, all be it one on a ring instead of a field.

To get the fixed points and cycles in this context, we need to perform repeated matrix multiplication as we have no Jordan canonical form available, and solve a system of form $Mx = b$ without having Gaussian elimination availble.
The last steps can be done as follows. Note that if we choose representatives of the coefficients in $\mathbb{Z}$, then we can solve $Mx = b$ for the representatives and since $\bar{\cdot} : \mathbb{Z} \to \mathbb{Z}_{|K^*|-1}$ is a ring morphism, for a solution x' of $Mx' = b$, $\overline{x'}$ will solve $\overline{M}x = \overline{b}$.
Now, we can use the Herminte normal form or the Smith normal for as $\mathbb{Z}$ is a PID (it's even Euclidean) to solve $Mx = b$.

5 Cellular automata

5.1 Basics of CA

We'll study a specific type of cellular automata:

Lattice cellular automata:

A **lattice-cellular automaton (lCA)** has state space $S^{(\mathbb{Z}^d)}$, where S is a finite set describing the state of the "cells" of $\mathbb{Z}^d$. The state of cells will change under a **local transition** function $f_l : S^{[n]} \to S$ depending on the sates of the cells in a neighborhood D, a finite set such as $B_1(0,1) \cap \mathbb{Z}^d$, of size n, so that in state vector $u \in S^{(\mathbb{Z}^d)}$, cell $z \in \mathbb{Z}^d$ will update its state to $f_l(u_{z+D})$. The global transition function is therefore described by $f = u \mapsto (f_l(u_{z+D}))_{z\in\mathbb{Z}^d}$.

Note that this isn't an FDS anymore, as the state space is infinite. We'll study lCA anyway.
An FDS version/generalization can be found in:

Graph cellular automata:

A **graph cellular automaton (gCA)** has state space S^V where the cells V are the vertex set of a graph (V, E). We equip this graph with positive edge weights $w : E \to \mathbb{R}_+$, which provides a metric d on the graph's vertices, where $d(u, v)$ is the length of a shortest path from u to v. The state of cells v will change under a **local transition** function $f_v : S^{D(v)} \to S$ depending on the sates of the cells in a neighborhood $D(v)$, a finite set such as $B_d(v, 1)$, so that in state vector $U \in S^V$, cell $v \in V$ will update its state to $f_v(U_{D(v)})$. The global transition function is therefore described by $f = u \mapsto (f_v(U_{D(v)}))_{v\in V}$.

The generalization come from considering $\mathbb{Z}^d$ as an infinite graph with vertices $\mathbb{Z}^d$ and edges linking vertices that differ by one coordinate only.

One can make a connection from graph cellular automata to graph coloring. If we choose states $E = [|V|]$, weights $w = 1$ so that the neighbourhoods are $D(v) = \delta(v) \cup \{v\} = B_d(v, 1)$, and local transition $f_v(x_{D(v)}) = \min([|V|]\backslash\{x_{\delta(v)}\})$ (smallest label/color not present in the neighborhood), then a fixed point of this system is a valid coloring of the graph. Indeed, if $x_v = \min([|V|]\backslash\{x_{\delta(v)}\})$, then in particular $x_v \neq x_w$ for all $w \in \delta(v)$, and that for all v.
However, the coloring may not be a minimum one. A counterexample can be built as follows: take copies if $K_1, K_2, ..., K_n$ and color them with a valid coloring, where K_q is colored with colors from $[q]$, then add a vertex and connect it to the higthest labeled/colored vertices of these copies and color it with $n + 1$. This produces a valid coloring, and it's even a fixed point of the graph. Indeed, in the complete graphs, the available colors in the iteration map are just the color of the vertex itself, and since the vertex linking the complete graphs has neighbours of all colors of $[n]$, it will stay at $n + 1$.
This coloring isn't minimum, however. We could have permuted the colors in the complete graphs so that the vertex linking them is connected to only vertices of with color 1, so that we could get a minimum coloring by coloring the central vertex with 2 (minimum since the complete graph K_n needs n colors).
COMPLETE: is this a approximation algorithm ? How fare off can the fixed-point-coloring be from a minimum one ?

It may be tempting do say that $(f(x))_v = f_v(x_{D(v)}) \leqslant x_v$.
However, for $f_v(x_{D(v)}) \leqslant x_v$ to occur, the color x_v must not be present in the neighbourhood of v, as

otherwise, it may happen that all lower colors are present in the neighbourhood of v, so that $f_v(x_{D(v)}) > x_v$.

Even if the initial coloring is a valid one, we have no guarantee that in the colorings of the iterates are valid, so we can't expect to always have $f_v(x_{D(v)}) \leqslant x_v$. This is for example the case of a 6-cycle in which we color the vertices from 1 to 6: two vertices will have their colors oscillate between 1 and 3 indefinitely, as can be seen after performing 4 iterations.

The study of CA we now give is similar to that of continuous dynamical systems.
We'll be interested in notions of convergence, which require a metric on the state-space:

Metrics:

For a lCA, for state vectors $x, y \in S^{(\mathbb{Z}^d)}$, we define $k(x,y) = \inf(q : x_{B_1(0,q)} \neq y_{B_1(0,q)})$ (the radius of the first Manhattan distance ball around the origin where the states disagree). Then $d_C(x,y) = \begin{cases} \frac{1}{k(x,y)+1} : \ k(x,y) \ finite \\ 0 : \ k(x,y) \ infinite \end{cases}$ is a metric on $S^{(\mathbb{Z}^d)}$. It's called the **Cantor metric** of the lCA.

For a gCA, for state vectors $x, y \in S^V$, we define the **Hamming distance** $d_H(x,y) = |\{v : x_v \neq y_v\}|$ (the number of cells who's states differ).

Proof: We only prove the triangular inequalities.
We have $\min(k(x,y)-1, k(y,z)-1) \leqslant k(x,z)-1$, as $\min(k(x,y)-1, k(y,z)-1) < k(x,z)$, for otherwise, $x_{B_1(0,k(x,z))} = y_{B_1(0,k(x,z))}$ and $y_{B_1(0,k(x,z))} = z_{B_1(0,k(x,z))}$, so that $x_{B_1(0,k(x,z))} = z_{B_1(0,k(x,z))}$, contradicting the definition of $k(x,z)$.
Since $\min(a,b) \leqslant c \Rightarrow \min(a+r, b+r) \leqslant c+r$ for real numbers, we have $\min(k(x,y)+1, k(y,z)+1) \leqslant k(x,z)+1$.
Next, note that $\frac{(k(x,y)+1)(k(y,z)+1)}{k(x,z)+1} \leqslant \frac{\min(k(x,y)+1, k(y,z)+1)\max(k(x,y)+1, k(y,z)+1)}{k(x,z)+1} \leqslant \max(k(x,y)+1, k(y,z)+1)$.
Finally, as $\max(k(x,y)+1, k(y,z)+1) \leqslant k(x,y)+1+k(y,z)+1$ (positivity), we get $\frac{(k(x,y)+1)(k(y,z)+1)}{k(x,z)+1} \leqslant k(x,y)+1+k(y,z)+1$ from which dividing by $(k(x,y)+1)(k(y,z)+1)$ and simplifying yields the triangular inequality.
For the Hamming distance note that if $x_v \neq z_v$, then at least one of $x_v \neq y_v$ or $y_v \neq z_v$ must hold (as equality is transitive).

Continuity:

The global transition functions of lCA and gCA are continuous wrt. the Cantor and Hamming distance respectively.

Proof: For any state u and any $\varepsilon > 0$, we which to find a $\delta > 0$ so that $d_C(u,v) < \delta \Rightarrow d_C(f(u), f(v)) < \varepsilon$ for all states v.
Note that $d_C(u,v) < \delta \Leftrightarrow k(u,v) > \frac{1-\delta}{\delta}$ so that for integers $a \leqslant \frac{1-\delta}{\delta} \Leftrightarrow \frac{1}{1+a} \geqslant \delta$, we have $u_{B(0,a)} = v_{B(0,a)}$. If we set $\frac{1}{1+a} = \delta$ for an a large enough so that $z+D \subseteq B(0,a)$ for all $z \in B(0,b)$ (for an integer b to be chosen soon), then $u_{z+D} = v_{z+D}$ and therefore $f_l(u_{z+D}) = f_l(v_{z+D})$, aka $(f(u))_z = (f(v))_z$

for all $z \in B(0,b)$, so that $k(f(u),f(v)) > b \Leftrightarrow d_C(f(u),f(v)) < \dfrac{1}{1+b}$. Choosing b so that $\dfrac{1}{1+b} \leqslant \varepsilon$ and a so that $z + D \subseteq B(0,a)$ for all $z \in B(0,b)$, which is possible as D is finite and hence bounded, does the job.
For the gCA, $\delta = \dfrac{1}{2}$ always does the job.

With the notions of distance and continuity in this discrete setting, we can use results from functional analysis or continuous dynamical systems.

For example, by noticing that the gCA metric space is complete (Cauchy sequences eventually become constant), we can use the Banach fixed point theorem on a certain class of transition functions to obtain the existence of a single fixed point to which all initial states converge.
Indeed, we contractions for which $d_H((f(u),f(v)) < d_H(x,y)$ fall in this category: as the Hamming distance is integer valued, $d_H((f(u),f(v)) < d_H(x,y) \Leftrightarrow d_H((f(u),f(v)) - 1 \leqslant d_H(x,y)$ and since $z \mapsto \dfrac{z-1}{z}$ is increasing, we have $\dfrac{d_H((f(u),f(v))-1}{d_H(x,y)} \leqslant \dfrac{d_H(x,y)-1}{d_H(x,y)} \leqslant \dfrac{|V|-1}{|V|}$, so that d_H is $\dfrac{|V|-1}{|V|}$-Lipschitz, and therefore a contraction.
However, finding non-trivial examples of f for which $d_H((f(u),f(v)) < d_H(x,y)$ is difficult.

5.2 Attractors

Attractors:

A set of states $Y \subseteq E^{(\mathbb{Z}^d)}$ or $Y \subseteq E^V$ is a **(pre-) attractor** if $f(Y) \subseteq Y$.

5.3 Greenberg-Hastings automaton

6 Data

6.1 References

- [$HAC96$] "Handbook of Applied Cryptography", by A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996