A finite field deterministic genetic network model

Oscar Moreno, Humberto Ortiz-Zuazaga, Carlos J. Corrada Bravo,
María Alicia Aviño-Diaz, and Dorothy Bollman.

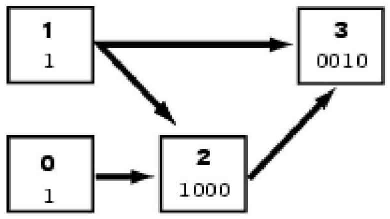
Abstract

Several research groups [1, 2, 3] have described genetic networks as networks of Boolean variables. For example, in [3] Ideker, Thorsson and Karp present a deterministic Boolean network model for genetic networks. We will show that these Boolean network models are examples of Finite Dynamical Systems ([13]–[19]), and we will generalize the deterministic Boolean network to finite fields. We also show how these Finite Dynamical System models over finite fields may be constructed from microarray experimental data, similar to the method employed by [3] to construct their models, using the results from [3, 4]. Our generalization, however, allows for a more natural treatment of microarray data than Boolean variables that have only two possible values, to a full range of discrete values.

Introduction: Dbnm and the reverse engineering problem.

In [3] Ideker, Thorsson and Karp describe a genetic network model based on Boolean networks, the deterministic Boolean network model or dBnm. In their model, gene levels or external biological stimuli can have two values (on or off). This is a greatly simplified biological model, which has the advantage of allowing simple computational analysis. Taking the model definition from this paper, we can describe a genetic network as:

1. A graph consisting of N numbered nodes and, $1 \leq n \leq N$.
2. A set of directed edges between nodes.
3. A Boolean function f_n for each node.
4. An edge from a node to another represents an influence of the first node on the expression of the second.



x_1	1	0	1	0
x_2	1	1	0	0
x_3	0	0	1	0

$$\begin{aligned} x_0 &:= 1 \\ x_1 &:= 1 \\ x_2 &:= x_0 \text{ and } x_1 \\ x_3 &:= x_1 \text{ and not } x_2 \end{aligned}$$

Figure 1: Example of the Boolean steady-state network model: (a) a directed graph structure with numbered nodes connected by edges, (b) the truth table (shown for node 3 only) and (c) the logic equations for each node.

Figure 1 reproduced from [3] represents a small example genetic network with 4 nodes and 4 edges. For a set of genes or stimuli and a set of perturbation experiments the authors construct an expression matrix as in Figure 2.

$$E = \begin{array}{|cccc|c} x_0 & x_1 & x_2 & x_3 & \\ 1 & 1 & 1 & 0 & p_0 \\ - & 1 & 0 & 1 & p_1 \\ 1 & - & 0 & 0 & p_2 \\ 1 & 1 & - & 1 & p_3 \\ 1 & 1 & 1 & + & p_4 \end{array}$$

Figure 2: Example expression matrix generated from the generic network in Figure 1.

The symbols + and- in the expression matrix represent genes or stimuli that are forced to 1 or 0, respectively. From this expression data, the challenge is to reconstruct or reverse engineer the genetic network. Ideker et al describe two inference procedures can reconstruct the genetic network interactively, suggesting new biological experiments to complement the existing data. From the expression matrix, the Predictor generates (possibly several) network hypothesis. The Chooser selects a new perturbation experiment, that would best discriminate between available hypotheses. Infering the network from the data in this manner means solving what is called the reverse engineering problem.

Our Definition of the dBnm.

We understand that the following is a formalization of the model presented in [3].

Definition 1. A Boolean variable assumes the values 0,1.

Definition 2. A Boolean function is a function involving Boolean variables and the operations and, or, not with the following definitions:

X	Y	X and Y	X or Y	Not X
1	1	1	1	0
1	0	0	1	0
0	1	0	1	1
0	0	0	0	1

Definition 3. A dBnm is a set of n Boolean variables $(x_1, \ldots, x_n)$ which are inputs, and a set of n Boolean functions which are the outputs $(f_1, \ldots, f_n)$. The Boolean variables represent genes or stimuli, and the Boolean function f_i represents how the gene i is determined by all the other genes.

Lemma 1 *Given n Boolean variables $(x_1, \ldots, x_n)$ and define $f(x_1, \ldots, x_n)$ for all possible values. Then there is a Boolean function that coincides with f as defined.*

Proof: See any book on computer architecture ([9]) for realizing a Boolean function as sums of products and products of sums.

Boolean network models are Finite Dynamical Systems.

Definition 4. + and * in Z_2 are defined as follows:

X	Y	$X * Y$	$X + Y$
1	1	1	0
0	1	0	1
1	0	0	1
0	0	0	0

Remark: Z_2 is a field with those two operations.

Definition 5. A polynomial function in the variables $X_1, \ldots, X_n$ over Z_2 is a multivariable polynomial in the variables $X_1, \ldots, X_n$.

Lemma 2 *Given n variables over Z_2 and define $f(x_1, \ldots, x_n)$ for all possible values then there is a function over Z_2 that coincides with f as defined.*

Note that for any two Boolean variables X, Y we have:

$$\begin{aligned} X \text{ and } Y &= X \cdot Y \\ X \text{ or } Y &= X + Y + X \cdot Y \\ \text{NOT } X &= 1 + X \\ X \text{ exclusive or } Y &= X + Y \end{aligned}$$

Now if we are given the function f we first invoke Lemma 1 and realize it as a Boolean function. Now using the above it is easy to see how we can realize it as a polynomial function over Z_2.

Example 1. To illustrate let us work the example of Figure 1, that was given in [3]. We have there $f_0 = 1, f_1 = 1, f_2 = x_0 \cdot x_1, f_3 = x_1 \cdot (x_2 + 1)$. Note now they are multivariable polynomials over the finite field Z_2.

Lemma 3 *The set of Boolean functions coincides with the set of functions over Z_2.*

Proof: This follows from Lemma 1 and Lemma 2.

Definition 6. A finite dynamical system (FDS) is a pair (V, f) where V is the set of vectors over a finite field GF(p^n) and $f : V \to V$.

Theorem 1 *The dBnm defined in [3] is a Finite Dynamical System.*

Proof: Note first, that the dBnm can be seen as a set of n functions over Z_2, by Lemma 3. The FDS over Z_2 can also be seen as a set of n functions (in the n variables $x_1, \ldots, x_n$) from Z_2^n to Z_2. Then, it is easy to see that they are in fact equivalent.

Example 2. Let us illustrate our theorem in the case of the example of Figure 1 (from [3]) and we will see that it is a finite dynamical system. Let $V = Z_2^4$ in our definition of F.D.S and let $f = (f0, f1, f2, f3)$ where $f_i, i = 0, 1, 2, 3$ are as it was given in the previous example. Then it should be clear that $f : Z_2^4 \to Z_2^4$ and that the example of Figure 1 is a F.D.S. The proof of the theorem follows the same method.

Genetic networks over finite fields and the reverse engineering problem.

In a natural manner, we have generalized in [4] FDS from the prime field GF(p) to a general finite field GF(p^n). Similarly, it is very natural to consider a generalization of dBnm where the variables will now no longer be Boolean variables but they vary over GF(p^n).

Definition 7. A 2^i-finite field variable is a variable over the finite field GF(2^i).

Definition 8. A polynomial function over GF(2^i). Is a multivariable polynomial function in the variables $X_1, \ldots, X_n$ over the finite field GF(2^i).

Definition 9. A 2^i deterministic finite field genetic network model (2^i-dFGnm) is a set of n 2^i-finite field variables (inputs), and a set n polynomial functions over GF(2^i) (outputs). As before the variables are the genes (or stimuli) and the polynomial functions give the gene network.

We will clarify this with the following example.

Example 3. Let us illustrate our generalization of dBnm from boolean variables to finite fields variables with an example. Consider again the example from Figure 1 and we will use the same graph and also we will define $f_0 = 1, f_1 = 1, f_2 =_0 \cdot x_1, f_3 = x_1 \cdot (x_2 + 1)$. The only difference will be that the variables will vary now not over Z_2 but over the finite field with 4 elements GF(4). Note that GF(4) can be represented as: GF(4) $= \{0, 1, \alpha, \alpha^2\}$ where α is a root of the polynomial $z^2 + z + 1$ (with coefficients in Z_2). By taking $1, \alpha$ as a basis, we can say that $0 = (0, 0), 1 = (0, 1), \alpha = (1, 0), \alpha^2 = (1, 1)$.

We now have the following theorem:

Theorem 2 *Our models over GF(p^n) will also be Finite Dynamical Systems.*

The proof of Theorem 2 is similar to that of Theorem 1 above.

Remark: genetic network over finite fields allow many more variations in expression levels of the genes or the input stimuli, compared to dBnm which only allow two expression levels of two levels of stimuli (present or absent). In particular, the current limitations of microarray analysis technology only allow for 2^{12} or 2^{16} detectable levels of expression. Our model would allow us to represent these microarray experiments with finite fields with 2^{12} or 2^{16} elements.

Theorem 3 *For any fixed basis $a_1, \ldots, a_r$ of GF(p^r) there is a natural one-one correspondence between the FDS over $(GF(p^r))^n$ and those over $(Z_p^r)^n$ (where Z_p is the field of p elements).*

Example 4. Note that $f = (f_0, f_1, f_2, f_3)$ will give us a function $f : (GF(4))^4 \to (GF(4))^4$ and we obtain a F.D.S. over a finite field. Therefore our generalized models are also F.D.S.

Theorem 3 has been proven in [4]. Now, using Theorem 2 and 1, and the methods described in [3], we can infer the genetic network, assuming we can force any node to any of the values of the finite field, and we have the following theorem:

Theorem 4 *If we have a 2^i-dFGnm and we assume we can force to any of the values of the finite field then we can solve the reverse engineering problem.*

Proof: Using Theorem 3 above from [4] a 2^i-dFGnm is a dBnm with $2^i n$ variables. The assumption on forcing the values means that the hypothesis that [3] uses to solve the reverse engineering problem are satisfied, and we can do the same and this proves our result.

Acknowledgment

We want to acknowledge that this work was inspired by the work of Prof. Solomon Golomb in [10], [11] and [12].

References

[1] T. Akutsu, S. Kuahara, O. Maruyama, S. Miyano, Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions. Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, H. Karloff, ed. (ACM Press, 1998).

[2] R. Somogyi and C. Sniegoski, Modeling the complexity of genetic networks: Understanding multigenic and pleiotropic regulation. Complexity 1, 45-63 (1996).

[3] T.E. Ideker, V. Thorsson, and R.M. Karp. Discovery of Regulatory Interactions Through Perturbation: Inference and Experimental Design, Pacific Symposium on Biocomputing 5:302-313 (2000).

[4] O. Moreno, D. Bollman, M. Aviño-Diaz, Finite Dynamical Systems, Linear Automata, and Finite Fields, Conference Proceedings, WSEAS Transactions, (2002).

[5] A. J. Viterbi, Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm, IEEE Trans. Inf. Theory, 260-269, April 1967.

[6] G. D. Forney, Jr., The Viterbi Algorithm, Proc. IEEE, 268-278, March 1973.

[7] G. D. Forney, Jr., Convolutional Codes II: Maximum Likelihood Decoding,Inf. Control, 222-266, July 1974.

[8] R. Lidl and H. Niederreiter, Finite Fields, Cambridge University Press, 1997.

[9] D. A. Patterson and J.L. Hennessy, Computer organization and design, Morgan Kaufmann Publishers, San Francisco 1997

[10] S. Golomb and E.C. Posner. Rook Domains, Latin Squares, Affine Planes, and Error-Distributing Codes. IEEE Transaction and Information Theory, July 1964, IT-10, 116-118.

[11] S. Golomb. Error-correcting codes and the genome project. Computers Chem. Vol 16, No 2 pp 183-186, 1992.

[12] S. Golomb, Genetic coding. Eng. And Sci., pp 9-14. April 1962.

[13] C. Barret, and C. Reidys, Elements of a theory of simulation I: Sequential CA Over Random Graphs, Appl. Math. Comput. 98pp 241-259, (1999).

[14] C. Barret, H. Mortveit, and C. Reidys, Elements of a theory of simulation II: sequential dynamical systems, Appl. Math. Comput. 107 (2000).

[15] C. Barret, H. Mortveit, and C. Reidys, Elements of a theory of simulation III: Equivalence of SDS, Appl. Math. Comput. 122 pp 325-340,(2001).

[16] D. Bollman, Some periodicity properties of transformations on vector spaces over residue class rings, J. Soc. Indust. Appl. Math. Vol 13, No3, September, 1965, pp 902-912.

[17] R. Laubenbacher and B. Pareigis, Equivalence relations on finite dynamical systems. Adv, in Appl. Math 26 (2001), 237-251.

[18] R. Laubenbacher and B. Pareigis, Decomposition and simulation of sequential dynamical systems, Preprint, (2002).

[19] R. Laubenbacher and B. Pareigis, Update schedules of Sequential Dynamical Systems preprint, (2002).