# Building model prototypes from time-course data

Alan Veliz-Cuba
University of Dayton

Stephen Randal Voss
University of Kentucky

David Murrugarra
University of Kentucky

July 22, 2022

**Abstract**

A primary challenge in building predictive models from temporal data is selecting the appropriate model topology and the regulatory functions that describe the data. Software packages are available for equation learning of continuous models, but not for discrete models. In this paper we introduce a method for building model prototypes. These model prototypes consist of a wiring diagram and a set of discrete functions that can explain the time course data. The method takes as input a collection of time course data or discretized measurements over time. After network inference, we use our toolbox to simulate the prototype model as a stochastic Boolean model. Our method provides a model that can qualitatively reproduce the patterns of the original data and can further be used for model analysis, making predictions, and designing interventions. We applied our method to a time-course, gene-expression data that were collected during salamander tail regeneration under control and intervention conditions. The inferred model captures important regulations that were previously validated in the research literature and gives novel interactions for future testing. The toolbox for inference and simulations is freely available at `github.com/alanavc/prototype-model`.

## 1 Introduction

The process of constructing discrete models from experimental data has several steps that have been studied in parallel. The main steps involved in this process are discretization, network inference, network selection, model interpolation, and deterministic/stochastic simulations [4,6,8,10,14,21,23,29]. Although some tools exist that address the global process [3, 12, 22], either the code is unavailable, not editable, or not in a ready-to-use format.

Equation learning (EQ) methods for differential equation (DE) models start with a collection of time course data and then "recovers" the governing equations using a library of functions [2, 9]. Many methods for EQ of DE models are based on formulating the inference problem as a parameter estimation problem

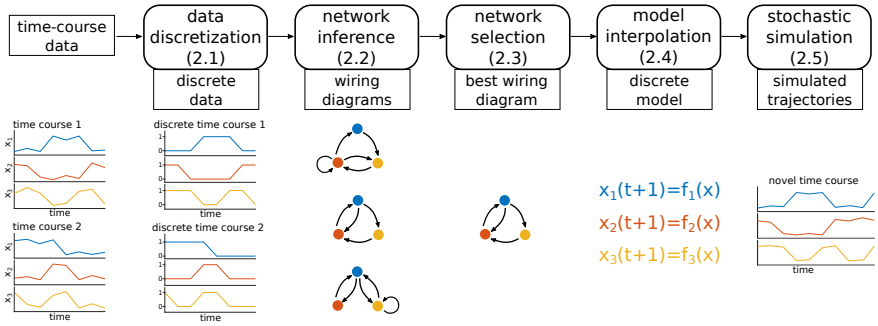



Figure 1: Flowchart showing the steps in model creation from data and the sections where each step is described. Starting from experimental time courses, we first transform the data into discrete values (in this case Boolean). Using algebraic techniques, we find wiring diagrams that explain the data. Each wiring diagram found will be consistent with all discrete time courses. We select the best wiring diagram and then find a discrete model that fits all the discrete data. This will result in a discrete model that can be simulated and compared with the original data. The model can also be run with new initial conditions or for longer time to create novel time courses that can be used to make predictions.

that can be solved via optimization techniques [2, 9]. Analogue methods for equation learning of discrete models that can learn both the network topology and the functions are still under development. Some of these existing methods can provide network candidates (i.e., possible wiring diagrams) that can explain the data. Other methods can provide candidate functions based on interpolating the data.

The main contribution of the paper is the combination of methods and the concrete toolbox that any user can use without familiarity with algebraic techniques. Our toolbox is modular, so that any step in the flowchart can be modified by the user without any restrictions. Importantly, it is also open-source and is freely available through a GitHub site. It works in Octave, so it is available for use in any operating system without the need of any license costs due to proprietary software. This makes our results fully reproducible.

The starting point of our method is experimental time-course data. Our focus is the construction of Boolean models, but we show with a toy model how our method also works for mixed-state models where variables can have different number of states or levels. As an application, we construct a model prototype using gene expression data for several time points which was collected during tail regeneration experiments in axolotls. We also use a synthetic network to illustrate the effect of data size, noise, and number of levels.

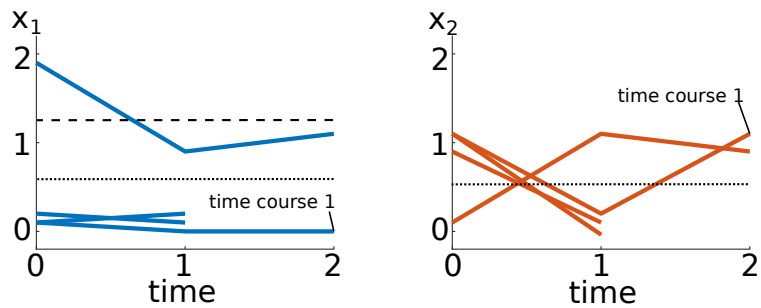



Figure 2: Values of $x_1$ and $x_2$ for the time courses. Variable $x_1$ can be considered as having 3 levels, whereas variable $x_2$ has 2 levels. The dashed lines show how the range of the data can be divided into regions (3 regions for $x_1$ and 2 for $x_2$), which will determine the discretization.

# 2 Methods

We will use "network" to refer to the correct wiring diagram and set of functions, and "model" to refer to the wiring diagram and set of functions that are obtained using data generated by the network.

Here we describe the methods for model selection (i.e., wiring diagram and regulatory functions) and the framework for simulations. We assume that we are given time courses of the form $s^1 \to s^2 \to \cdots \to s^r$, where $s^i = (s^i_1, \ldots, s^i_n) \in S = S_1 \times \cdots \times S_n$. Here $S_i$ is a finite set of all the values that the $i$-th variable can take. Note that if $S_i = \{0, 1\}$, then we have a Boolean model.

**Example 2.1.** *To illustrate the methods, we use an example with the following four time courses.*

**(1)** $(0.1, 1.1, 1.9, 0.9, 0.2) \to (0.0, 0.2, 0.2, 0.1, 0.1) \to (0.0, 1.1, 0.1, 1.9, 2.1)$
**(2)** $(1.9, 0.1, 0.9, 0.1, 0.0) \to (0.9, 1.1, 0.1, 1.9, 2.1) \to (1.1, 0.9, 0.1, 1.9, 2.0)$
**(3)** $(0.2, 1.1, 1.9, 0.9, 1.1) \to (0.1, 0.0, 0.2, 0.1, 0.1)$
**(4)** $(0.1, 0.9, 2.1, 1.1, 2.1) \to (0.2, 0.1, 0.2, 0.1, 1.1)$

## 2.1 Discretization

We implemented a simple discretization method based on binning data by dividing the range of the data into equally spaced regions. The time courses suggest that the number of levels for variables $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, are 3, 2, 3, 3, 3, respectively. For example, by plotting the values of $x_1$ and $x_2$ for each trajectory (Fig. 2), we see that $x_1$ has 3 distinctive levels and $x_2$ has 2 distinctive levels. For $x_1$, all values below the dotted line will be mapped to 0 (low); all values between the dotted and dashed lines will get mapped to 1 (medium); and all values above the dashed line will get mapped to 2 (high). For $x_2$, all values below the dotted line will be mapped to 0 (low); and all values above the dotted line will get mapped to 1 (high).

Then, the discrete time courses are given below.
**(1)** $01210 \to 00000 \to 01022$
**(2)** $20100 \to 11022 \to 11022$
**(3)** $01211 \to 00000$
**(4)** $01212 \to 00001$
In this case $S = \{0,1,2\} \times \{0,1\} \times \{0,1,2\} \times \{0,1,2\} \times \{0,1,2\}$.

## 2.2 Network Inference

To find the wiring diagrams that are consistent with a collection of time courses of the form $s^1 \to s^2 \to \cdots \to s^r$ we use the algebraic framework introduced in [23]. This framework takes partial information about the evolution of a network $s \to f(s)$ and returns all the minimal wiring diagrams that are consistent with the data. This approach guarantees that for each minimal wiring diagram there exists a model that fits the data such that each interaction is activation or inhibition [23] .

To use the framework in [23], we first note that each time course $s^1 \to s^2 \to \cdots \to s^r$ implies that $s^{j+1} = f(s^j)$ for $j = 1, \ldots, r-1$, where $f$ is the network one is trying to infer. This results in a set $D \subseteq S$ such that $f(s)$ is known for every $s \in D$. That is, $D$ is the set of inputs for which we know the outputs.

**Example 2.2.** *In Example 2.1,* $D = \{01210, 00000, 20100, 11022, 01211, 01212\}$. *Then, the partial information we have is given by the table*

| $x$ | $f(x)$ |
|---|---|
| 01210 | 00000 |
| 00000 | 01022 |
| 20100 | 11022 |
| 11022 | 11022 |
| 01211 | 00000 |
| 01212 | 00001 |

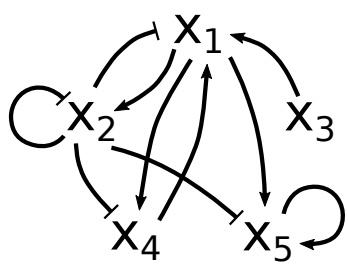
Table 1: Partial information for example.

*Then, using the algebraic techniques in [23] we can find all minimal wiring diagrams that are consistent with the data. For each variable* $x_i$ *in the network, the algebraic framework returns* $W_1, \ldots, W_k$, *where each* $W_j$ *is a minimal set of inputs for variable* $i$. *For our example we obtain Table 2.*

By selecting one wiring diagram for each $x_i$, we obtain a (global) wiring diagram that is consistent with the data. For example, if we select $\{x_2^-, x_3^+, x_4^+\}$ for $x_1$, $\{x_1^+, x_2^-\}$ for $x_2$, $\{\ \}$ for $x_3$, $\{x_1^+, x_2^-\}$ for $x_4$, and $\{x_1^+, x_2^-, x_5^+\}$ for $x_5$, we obtain the wiring diagram shown in Fig. 3. To compare different wiring diagrams we can use the adjacency matrix representation.

| $x_i$ | minimal wiring diagrams for $x_i$ |
|---|---|
| $x_1$ | $\{x_1^+\}$, $\{x_2^-, x_3^+, x_4^+\}$ |
| $x_2$ | $\{x_3^-\}$, $\{x_2^-, x_4^+\}$, $\{x_1^+, x_4^-\}$, $\{x_1^+, x_2^-\}$ |
| $x_3$ | $\{\}$ |
| $x_4$ | $\{x_3^-\}$, $\{x_2^-, x_4^+\}$, $\{x_1^+, x_4^-\}$, $\{x_1^+, x_2^-\}$ |
| $x_5$ | $\{x_3^-, x_5^+\}$, $\{x_2^-, x_4^+, x_5^+\}$, $\{x_1^+, x_4^-, x_5^+\}$, $\{x_1^+, x_2^-, x_5^+\}$ |

Table 2: Minimal wiring diagrams. The $+/-$ superscripts indicate activation/inhibition. For example, the set $\{x_2^-, x_3^+, x_4^+\}$ indicates that one way to explain the data is for $x_2$ to be an inhibitor of $x_1$ and $x_3$ and $x_4$ to be activators of $x_1$. By choosing one set for each variable, one obtains a wiring diagram that is consistent with the data. Note that no variable affects $x_3$ (i.e., constant function).




$$\begin{pmatrix} 0 & -1 & 1 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix}$$

Figure 3: Example of wiring diagram consistent with the data. Left: Wiring diagram. Right: Adjacency matrix representation.

## 2.3 Wiring diagram selection

The network inference described in Section 2.2 could return several minimal network candidates for each variable. That is, for a given time course data, there might be several models that explain the data and that are minimal. The method will return all candidate wiring diagrams. In order to select one model out of all possible options, we calculate the "best wiring diagram" by including only the most frequent interactions from the wiring diagrams found. For each variable, $x_i$, we quantified the frequency $q_{ji}^+$ of positive interactions $x_j \longrightarrow x_i$ across all possible wiring diagrams and the frequency $q_{ji}^-$ of negative interaction $x_j \dashv x_i$ across all possible wiring diagrams for all $j = 1, \ldots, n$. That is, the parameter $q_{ji}^+$ (resp. $q_{ji}^-$) represents the frequency of regulator $x_j^+$ (resp. $x_j^-$) in the minimal wiring diagrams of $x_i$ (see Example 2.3 and Table 3 for additional details). Then we construct an adjacency matrix $W^*$ by considering the interactions with a frequency above certain threshold $\tau$. If conflicts arise (that is, when $q_{ji}^- = q_{ji}^+$ for some $j$), then we discard those interactions. Subsequently, for each row of $W^*$, say $W_i^*$, we calculate the distance with each possible wiring diagram of $x_i$ (these are represented as rows). Finally, we construct an adjacency matrix $W$ with rows corresponding to the rows with minimum distances.

**Example 2.3.** *For the network in Example 2.1, we calculated the frequency of the interactions; see Table 3. For instance, for variable $x_1$ in the first row of Table 3, $x_1^+$ appears in one out of two wiring diagrams (see Table 2), therefore $q_{11}^+ = 1/2$. Then, we computed an adjacency matrix $W^*$ by including the interactions with*

| | Frequencies of activations and inhibitions | # of WDs |
|---|---|---|
| $x_1$ | $q_{11}^+ = 1/2$, $q_{21}^- = 1/2$, $q_{31}^+ = 1/2$, $q_{41}^+ = 1/2$ | 2 |
| $x_2$ | $q_{12}^+ = 2/4$, $q_{22}^- = 2/4$, $q_{32}^- = 1/4$, $q_{42}^+ = 1/4$, $q_{42}^- = 1/4$ | 4 |
| $x_3$ | NA | 0 |
| $x_4$ | $q_{14}^+ = 2/4$, $q_{24}^- = 2/4$, $q_{34}^- = 1/4$, $q_{44}^- = 1/4$, $q_{44}^+ = 1/4$ | 4 |
| $x_5$ | $q_{15}^+ = 2/4$, $q_{25}^- = 2/4$, $q_{35}^- = 1/4$, $q_{45}^+ = 1/4$, $q_{45}^- = 1/4$, $q_{55}^+ = 4/4$ | 4 |

Table 3: Frequencies of interactions on minimal wiring diagrams. The parameter $q_{ji}^+$ (resp. $q_{ji}^-$) represents the frequency of regulator $x_j^+$ (resp. $x_j^-$) in the minimal wiring diagrams of $x_i$. For instance, for variable $x_1$ in the first row, $x_1^+$ appears in one out of two wiring diagrams, therefore $q_{11}^+ = 1/2$.

*a frequency above the threshold $\tau = 1/5$. We discarded conflicting interactions (i.e., the cases where $q_{ji}^- = q_{ji}^+$).*

$$W^* = \begin{pmatrix} 1 & -1 & 1 & 1 & 0 \\ 1 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 1 \end{pmatrix}$$

*Then, for each row of $W^*$, say $W_i^*$, we calculate the distance with each possible wiring diagram of $x_i$ (these are represented as rows). Then we construct an adjacency matrix with rows corresponding to the rows with minimum distances. Then, the matrix after the distance calculations is:*

$$W = \begin{pmatrix} 0 & -1 & 1 & 1 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 1 \end{pmatrix}$$

The reason for why we take a distance approach is because there might not be a truth table satisfying the matrix $W^*$ but there is certainly one for $W$ as shown in Example 2.2.

## 2.4 Fitting Model to Data

After one wiring diagram has been selected from the family of minimal wiring diagrams, we proceed to construct a function that fits the data. Although there are known formulas for interpolation, we are interested in *monotone* interpolation,

that is, we need to find a model that not only fits the data, but one whose signs of interaction match the wiring diagram selected.

We illustrate our approach with wiring diagram $\{x_1^+, x_2^-\}$ for variable $x_4$. Since it is guaranteed that there is a monotone function $h(x_1, x_2)$ that fits the data for variable $x_4$ [23], we consider Table 1 with only $x_1$ and $x_2$ in the first column (inputs) and only $x_4$ in the second column (output).

| $(x_1, x_2)$ | output |
|---|---|
| 01 | 0 |
| 00 | 2 |
| 20 | 2 |
| 11 | 2 |
| 01 | 0 |
| 01 | 0 |

Table 4: Partial information for variable $x_4$ with wiring diagram $\{x_1^+, x_2^-\}$.

We now rewrite this table as a truth table by ordering the inputs lexicographically ($x_1 \in \{0, 1, 2\}$, $x_2 \in \{0, 1\}$), where some entries are unknown, Table 5.

| $(x_1, x_2)$ | output | $h(x_1, x_2)$ |
|---|---|---|
| 00 | 2 | 2 |
| 01 | 0 | 0 |
| 10 | ? | 2 |
| 11 | 2 | 2 |
| 20 | 2 | 2 |
| 21 | ? | 2 |

Table 5: Incomplete truth table for variable $x_4$ with wiring diagram $\{x_1^+, x_2^-\}$ and the corresponding construction of a function for all inputs.

To fill in the table, we use the fact that the function increases with respect to $x_1$ and decreases with respect to $x_2$. For example, since $h(2, 1) \geq h(1, 1) = 2$, it follows that $h(2, 1) = 2$. Similarly, since $2 = h(0, 0) \leq h(1, 0) \leq h(2, 0) = 2$, it follows that $h(1, 0) = 2$. In this way, we obtain the value of the missing entries. This process can be done for all wiring diagrams and for all variables. Since the existence of a wiring diagram guarantees that there is at least one suitable function that fits the data, this is always possible [23]. To guarantee that the fitting is unique, we implemented the algorithmic construction from Lemma 2.4 from [23] .

## 2.5 Stochastic Framework

For the simulations we will use the stochastic framework introduced in [16] referred to as Stochastic Discrete Dynamical Systems (SDDS). This framework is a natural extension of Boolean networks and is an appropriate setup to

model the effect of intrinsic noise on network dynamics. Consider the discrete variables $x_1, \ldots, x_n$ that can take values in finite sets $S_1, \ldots, S_n$, respectively. Let $S = S_1 \times \cdots \times S_n$ be the Cartesian product. An *SDDS* in the variables $x_1, \ldots, x_n$ is a collection of $n$ triplets

$$F = \{f_i, p_i^{\uparrow}, p_i^{\downarrow}\}_{i=1}^{n}$$

where

- $f_i : S \to S_i$ is the update function for $x_i$, for all $i = 1, \ldots, n$.
- $p_i^{\uparrow} \in [0, 1]$ is the activation propensity.
- $p_i^{\downarrow} \in [0, 1]$ is the degradation propensity.

The stochasticity originates from the propensity parameters $p_i^{\uparrow}$ and $p_i^{\downarrow}$, which should be interpreted as follows: If there would be an activation of $x_k$ at the next time step, i.e., if $s_1, s_2 \in S_k$ with $s_1 < s_2$ and $x_k(t) = s_1$, and $f_k(x_1(t), \ldots, x_n(t)) = s_2$, then $x_k(t+1) = s_2$ with probability $p_i^{\uparrow}$. The degradation probability $p_i^{\downarrow}$ is defined similarly. SDDS can be represented as a Markov chain by specifying its transition matrix in the following way. For each variable $x_i$, $i = 1, \ldots, n$, the probability of changing its value is given by

$$Prob(x_i \to f_i(x)) = \begin{cases} p_i^{\uparrow}, & \text{if } x_i < f_i(x), \\ p_i^{\downarrow}, & \text{if } x_i > f_i(x), \\ 1, & \text{if } x_i = f_i(x), \end{cases}$$

and the probability of maintaining its current value is given by

$$Prob(x_i \to x_i) = \begin{cases} 1 - p_i^{\uparrow}, & \text{if } x_i < f_i(x), \\ 1 - p_i^{\downarrow}, & \text{if } x_i > f_i(x), \\ 1, & \text{if } x_i = f_i(x). \end{cases}$$

Let $x, y \in S$. The transition from $x$ to $y$ is given by

$$a_{xy} = \prod_{i=1}^{n} Prob(x_i \to y_i). \tag{1}$$

Notice that $Prob(x_i \to y_i) = 0$ for all $y_i \notin \{x_i, f_i(x)\}$.

The stochastic framework is implemented in the toolbox to give the user with simulation options including the deterministic case. By setting all propensities equal to 1, one obtains a deterministic model. Alternatively, setting all propensities equal to 0.9 gives a 90% chance of using the regulatory function for each node and a 10% chance of keeping the current state. Likewise, setting all propensities equal to 0.5 gives a 50% chance of using the regulatory function for each node and a 50% chance of keeping the current state value. Furthermore, one could use the parameter estimation techniques for computing the propensity parameters of SDDS that have been presented in [15].

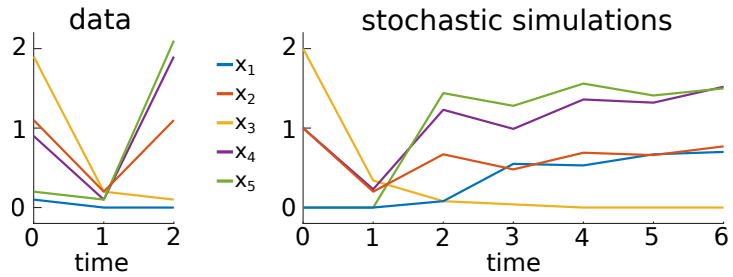



Figure 4: Comparison between data (only first time course shown) and stochastic simulations. Using the discretization of the initial condition of the data, 01210, we can use the model obtained to simulate the system for any arbitrary number of steps. Stochastic simulations are the average of 100 realizations, so they may take non discrete values between 0 and 2.

# 3 Applications: Gene expression data from experiments in axolotls

In this section we apply our method to a time-course, gene expression data that were collected during salamander (axolotls – *Ambystoma mexicanum*) tail regeneration under control and intervention conditions. Chemicals that inhibit cell-signaling activities are used as intervention agents to block tail regeneration and alter gene expression. Modeling gene interactions can provide confirmatory and novel information for developing hypotheses about the actions of cell-signaling molecules and transcription factors that orchestrate tissue regeneration.

Using our method, we generated a Boolean model for a set of 10 genes that were expressed differently during axolotl tail regeneration under control and Wnt C59 treatment, a chemical that blocks the secretion of Wnt signaling molecules from cells [20]. We note that the Wnt C59 intervention in that study inhibited tail regeneration. Seven of the genes are ligands (*Areg*, *Fgf9*, *Bmp2*, *Inhbb*, and *Wnt5a*) or negative feedback regulators (*Dusp6*, *Nradd*) of cell signaling pathways, *Sp7* is a bone-specific transcription factor, *Hapln3* is a cell adhesion molecule and *Phlda2* is an intracellular protein. We label these genes using the following variables:

$$\begin{array}{ll} x_1 = \mathit{Areg}, & x_2 = \mathit{Phlda2}, \\ x_3 = \mathit{Fgf9}, & x_4 = \mathit{Bmp2}, \\ x_5 = \mathit{Nradd}, & x_6 = \mathit{Hapln3}, \\ x_7 = \mathit{Sp7}, & x_8 = \mathit{Wnt5a}, \\ x_9 = \mathit{Inhbb}, & x_{10} = \mathit{Dusp6}. \end{array} \tag{2}$$

In Figure 5 we show the wiring diagrams obtained using our method for both conditions, control and intervention. These wiring diagrams present gene-by-gene interactions for the given gene expression data set. For the control case, *Wnt5a* represents a key node in the network; this Wnt signaling ligand

is predicted to activate ligands that function in BMP (*Bmp2*), FGF (*Fgf9*), and TGF$\beta$ (*Inhbb*) pathways, and transcriptional regulation of bone formation (Sp7) [7], consistent with Wnt signaling playing a central, integrative role in regeneration [26] (Figure 5 a). Additionally, for the control case, we note that the well-established inhibitory effect of Dusp6 on FGF signaling is captured by this wiring diagram [11], and it also implicates *Wnt5a* as an inhibitor of *Areg* during regeneration. By comparing the intervention with the control case, we see that Wnt C59 eliminated all of the *Wnt5a* activating edges and the inhibitory edge to *Areg*. A new activating edge from *Areg* to *Nradd* suggests a novel hypothesis that Wnt C59 blockade of Wnt ligand secretion indirectly (via Nradd) inhibits the transcription of BMP and FGF pathway ligands that are required for tissue regeneration [27].

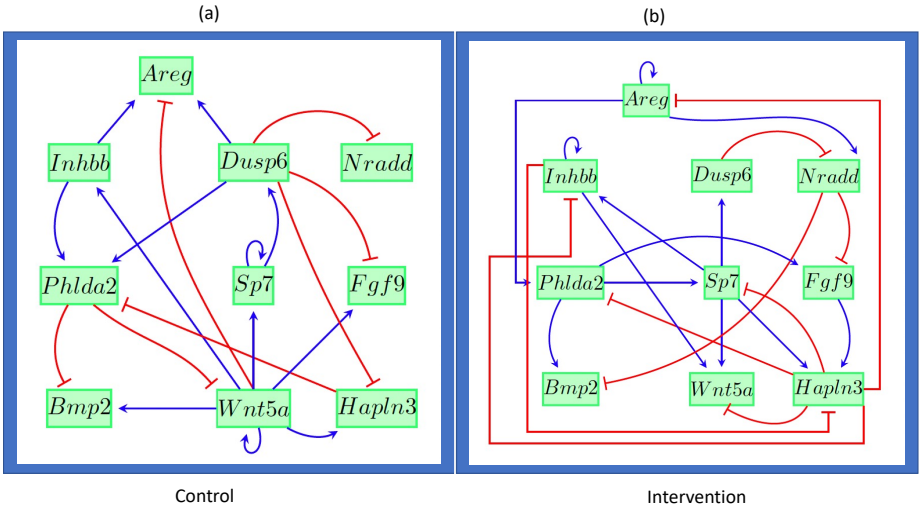



Figure 5: Wiring diagrams for the genes in Equation 2 for both conditions: (a) control and (b) intervention. Blue edges represent activation while red edges inhibition.

To further validate this model we compare the experimental data versus the simulations that are shown in Figures 6-7. These figures were obtained from 100 runs. Simulations using the framework SDDS [16] were performed initializing the system at the initial state 1100000001. This initialization represents a discretized version of the actual data at time 0. For the simulations in Figures 6-7 we used propensities equal to 0.9 for all variables. To assess the quality of the predictions, we use the Mean Squared Error (MSE) between the discretized data and simulated trajectories (see the Appendix for details of the MSE). We also generated simulation plots using propensities equal to 0.5 for all variables, see Figures 11-12. From the MSE values, it can be seen that the simulated trajectories with propensities equal to 0.9 give better fits of the discretized data. The propensity values can further be optimized using the method in [15].

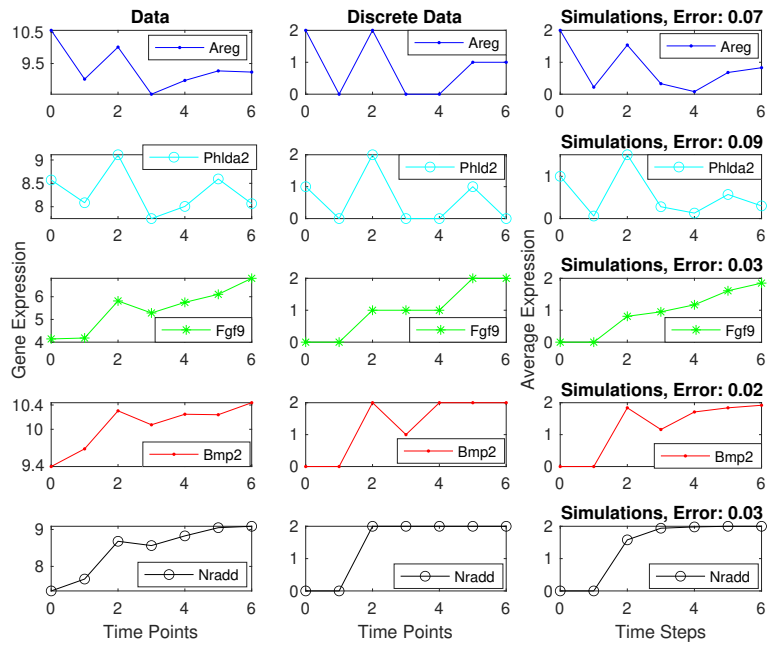



Figure 6: Gene expression data, discretized data, and simulations of the first five genes in Equation 2. The plots in the left panel are the experimental data, the ones in the middle are the discretized data, and the ones in the right panel are average expressions from simulations of 100 runs, all initialized at 1100000001. For the simulations, all the propensities are equal to 0.9 and the mean squared error is between the discretized data and simulated trajectories.

This model can further be used to attractor analysis, control, modularity, etc. However, the main result from this application is the potential novel interactions that can be experimentally tested.

# 4 Effect of data size, noise, number of levels/states, and threshold value

Since our toolbox consists in the combination of several methods/algorithms, any advantages or disadvantages of these will affect the performance of the model created. We explore some of these effects using synthetic networks so that we can compare the model constructed with the original network.

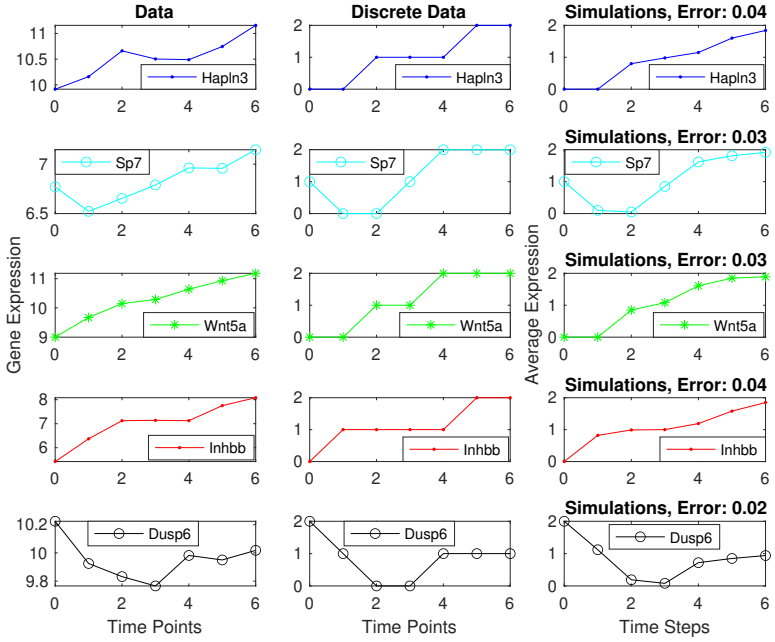



Figure 7: Gene expression data, discretized data, and stochastic simulations of the last five genes in Equation 2. The plots in the left panel are experimental data, the ones in the middle are discretized data, and the ones in the right panel are average expressions from simulations of 100 runs, all initialized at 1100000001. For the simulations, all the propensities are equal to 0.9 and the mean squared error is between the discretized data and simulated trajectories.

## 4.1 Effect of data size

Here we illustrate the effect of data size using a synthetic example given by the Boolean network $f = (f_1, \ldots, f_5)$, where

$$\begin{aligned} f_1 &= \neg x_3 \vee x_5 \\ f_2 &= x_1 \wedge x_4 \\ f_3 &= \neg x_2 \\ f_4 &= \neg x_3 \wedge x_5 \\ f_5 &= x_4 \end{aligned} \tag{3}$$

We will generate synthetic data using this Boolean network. Then, *using the data only*, we will see if the following novel trajectories can be predicted: $01111 \to 10001 \to 10110 \to 01101 \to 10000$ and $00100 \to 00100$ (that is, 00100 is a steady state). That is, we will initialize the system at 00100 and 01111 in the model predicted by our toolbox. The results are summarized in Table 6. To

illustrate the effect of the number of data points clearer, we do not consider stochasticity in this subsection (all propensities are set equal to 1).

First, we start with the synthetic time series $T_1 = \{11011 \to 11011\}$ and $T_2 = \{11010 \to 11001 \to 10010\}$. With this data, our toolbox predicts that the trajectories initialized at 01111 and 00100 are $01111 \to 11011 \to 11011 \to 11011 \to 11011$ and $00100 \to 10000$. In this case the data was not enough to recover the trajectories.

Second, we start with the synthetic time series $T_1$, $T_2$, and also $T_3 = \{11110 \to 01001 \to 10010 \to 11101\}$. With this data, our toolbox predicts that the trajectories initialized at 01111 and 00100 are $01111 \to 01011 \to 11011 \to 11011 \to 11011$ and $00100 \to 00100$. In this case we see that with more data still the trajectory for 01111 was not recovered, but the model created by the toolbox correctly predicted that 11011 is a steady state.

Third, we start with the synthetic time series $T_1$, $T_2$, $T_3$, and also $T_4 = \{01011 \to 10011 \to 11111 \to 11001\}$. With this data, our toolbox predicts that the trajectories initialized at 01111 and 00100 are $01111 \to 10001 \to 10110 \to 01101 \to 10000$ and $00100 \to 00100$. That is, with the data given, the model predicted by the toolbox was able to correctly reproduce the novel trajectories. Note that $T_1, T_2, T_3, T_4$ represent 9 out of the $2^5 = 32$ possible transitions.

| Data used | Novel trajectories predicted |
|---|---|
| $T_1, T_2$ | $01111 \to 11011 \to 11011 \to 11011 \to 11011$, $00100 \to 10000$ |
| $T_1, T_2, T_3$ | $01111 \to 01011 \to 11011 \to 11011 \to 11011$, $00100 \to 00100$ |
| $T_1, T_2, T_3, T_4$ | $01111 \to 10001 \to 10110 \to 01101 \to 10000$, $00100 \to 00100$ |

Table 6: Effect of increasing the data size. $T_1 = \{11011 \to 11011\}$, $T_2 = \{11010 \to 11001 \to 10010\}$, $T_3 = \{11110 \to 01001 \to 10010 \to 11101\}$, $T_4 = \{01011 \to 10011 \to 11111 \to 11001\}$. Transitions predicted correctly are indicated by bold arrows.

## 4.2 Effect of noise

Here we study the effect of noise in the model predicted by the toolbox. We use the Boolean network in Eq. 3 as the truth and the trajectories $T_1, T_2, T_3, T_4$ as data. To this data we will add noise following a uniform distribution centered at 0 and with standard deviation $\sigma$. With the noisy data, we use the toolbox to create a model and make a prediction of the trajectory with initial condition 01111.

The results are shown in Fig. 8. We see that for small noise the model is still able to make an accurate prediction of the trajectory. Since Boolean models focus on the qualitative features of he dynamics, it is robust to small noise levels. However, for large enough noise, the predicted trajectory does not match the true trajectory. A possible cause is that the first step of the discretization needs to distinguish between "low" and "high". If noise is large, it is possible that a

low value plus noise is larger than a high value plus noise and may be incorrectly discretized.

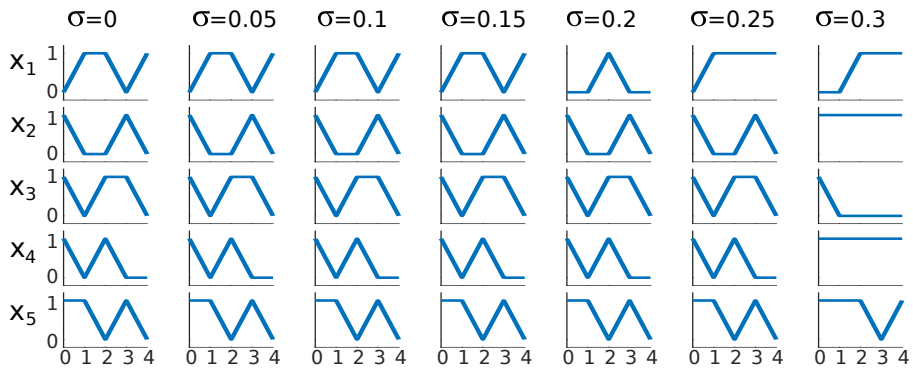



Figure 8: Effect of noise on predicted model. For small noise, the qualitative behavior is maintained, but as noise increases, the predicted model loses its predictive power.

### 4.3 Effect of number of levels

Here we use the network with 4 levels (or states) $f = (f_1, \ldots, f_5) : \{0, 1, 2, 3\}^5 \to \{0, 1, 2, 3\}^4$, where

$$\begin{aligned} f_1 &= \max(3 - x_3, x_5) \\ f_2 &= \min(x_1, x_4) \\ f_3 &= 3 - x_2 \\ f_4 &= \min(3 - x_3, x_5) \\ f_5 &= x_4 \end{aligned} \tag{4}$$

We generated 5 trajectories using this network: 32032 → 33123 → 32022 → 32122 → 22122, 23131 → 22013 → 31131 → 23213 → 31011, 22322 → 22102 → 20120 → 22302 → 20100, 03123 → 30022 → 32322 → 22102 → 20120, 03121 → 20012 → 31321 → 12202 → 20110. Using this data only, we use our toolbox to create a model and make a prediction for the novel trajectory with initial condition 03232. The true trajectory is shown in Fig. 9 (first column).

Selecting a discretization of the data with 2 levels in the toolbox (0 and 1 become 0, and 2 and 3 become 1) will create a Boolean model. In this case, the initial condition 03232 would become 01111 and the Boolean trajectory will of course not match the true trajectory exactly. However, the Boolean trajectory does have the same qualitative features of the true trajectory, Fig. 9. For instance, $x_1$ has the pattern $0 \to 2 \to 3 \to 1 \to 3$ in the true trajectory. If discretized, this would be $0 \to 1 \to 1 \to 0 \to 1$, just like the predicted Boolean trajectory.

We then considered a discretization that included the previous one. To achieve this we considered 4 levels. Based on the data, using 4 levels is a more natural discretization and indeed, the trajectory predicted by the toolbox matches the true trajectory, Fig. 9.

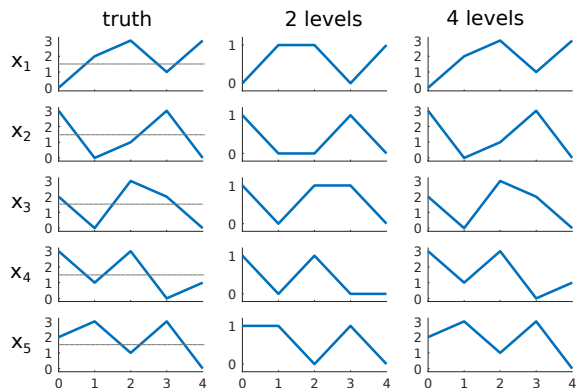



Figure 9: Effect of different levels. Using a coarser discretization, some features are lost. Using an appropriate number of levels, more features can be captured by the model.

### 4.4 Effect of threshold values

We now explore the effect of changing the threshold chosen for discretization on the predicted model. We use the network $f : \{0, 1, 2\}^3 \to \{0, 1, 2\}^3$ given by

$$\begin{aligned} f_1 &= x_2 \\ f_2 &= \min(x_1, x_3) \\ f_3 &= \max(x_1, 1) \end{aligned} \tag{5}$$

We use the data $111 \to 111$, $020 \to 201$, $002 \to 001$, $220 \to 202 \to 022$ and will attempt to predict the true trajectory $020 \to 201 \to 012 \to 101 \to 011$ using the model given by the toolbox. To make the comparison simpler, we choose 2 states only. Choosing different thresholds can potentially result in different models with different dynamical properties. The difference is shown in Fig. 10. Different states can be mapped to the same value if they are both less than or greater than the threshold. This causes the loss of certain features, but some coarse qualitative features are preserved.

## 5 Discussion

Discrete models have been successfully to model biological systems [25, 29]. Although several discrete modeling packages exist for their analysis (e.g., PlantSim-

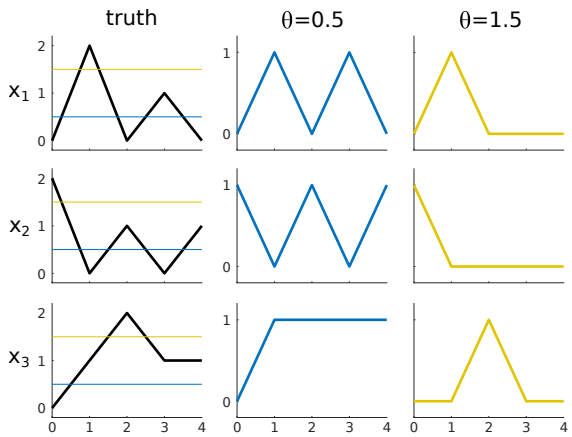



Figure 10: Effect of different thresholds. For thresholds around $\theta = 0.5$, values that should have been 1 or 2 in the true trajectory are all mapped to 1. For thresholds around $\theta = 1.5$, the values that should have been 0 or 1 are mapped to 0.

Lab [5], BoolNet [17], BNReduction [24], GinSim [18], CaSQ [1], WebMaBoSS [19]), they require an existing model or the wiring diagram to be created by the user. Few tools exist that provide an automated and easily customizable pipeline to quickly create model prototypes. Our toolbox allows the creation of model prototypes easily, which can then be used by existing modeling packages for validation, modification, or extension.

Equation learning methods in general require large amounts of data which might not be feasible in practice [2, 9]. Furthermore, those approaches require knowledge of the form of the functions (some times called a library of functions) *a priori*, which may be unfeasible for unknown interactions. Even if the form of the functions is known for continuous modeling, the model obtained can be the result of parameter estimation being stuck in a local minimum. In contrast, our method can be used even with a limited number of time points. Although this does not guarantee predictive power, our toolbox does find all minimal wiring diagrams. This is important, because it can be seen as the discrete version of finding all local minima in parameter estimation for continuous models. Furthermore, our approach does not need to know the form of the functions *a priori*. We note that the discrete model resulting from our approach can be converted into a continuous model using existing approaches such as [13, 28].

The limitations of our toolbox are those related to each component in the pipeline. Notably, if the discretization considers two instances of the same value as different due to noise (e.g. 0.7 as 0 and 0.9 as 1), this can cause overfitting. Selecting the correct number of levels of the model is also important and can cause missing some features if the number of levels is too low or overfitting if the number of levels is too large. Also, the selection of thresholds can make a

difference on which features of the true dynamics are correctly predicted with the inferred model. Another limitation is that it is not known how much data is needed to guarantee that the predicted model is "close" to the true network unless the network is known *a priori*.

For the purpose of reproducibility, we provide all the data and the code that we use in our toy example and application which can be accessed through this link: `github.com/alanavc/prototype-model`.

# References


[1] Sara Sadat Aghamiri, Vidisha Singh, Aurélien Naldi, Tomáš Helikar, Sylvain Soliman, and Anna Niarakis. Automated inference of Boolean models from molecular interaction maps using casq. *Bioinformatics*, 36(16):4473–4482, 2020.

[2] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the national academy of sciences*, 113(15):3932–3937, 2016.

[3] Elena Dimitrova, Luis David García-Puente, Franziska Hinkelmann, Abdul S Jarrah, Reinhard Laubenbacher, Brandilyn Stigler, Michael Stillman, and Paola Vera-Licona. Parameter estimation for Boolean models of biological networks. *Theoretical Computer Science*, 412(26):2816–2826, 2011.

[4] Elena S Dimitrova, M Paola Vera Licona, John McGee, and Reinhard Laubenbacher. Discretization of time series data. *Journal of Computational Biology*, 17(6):853–868, 2010.

[5] S Ha, E. Dimitrova, D. Hoops, S.and Altarawy, M. Ansariola, D. Deb, J. Glazebrook, R. Hillmer, H. Shahin, F. Katagiri, J. McDowell, M. Megraw, J. Setubal, B. M. Tyler, and R. Laubenbacher. PlantSimLab - a modeling and simulation web tool for plant biologists. *BMC Bioinformatics*, 20(1):508, 2019.

[6] Franziska Hinkelmann and Abdul Salam Jarrah. Inferring biologically relevant models: nested canalyzing functions. *International Scholarly Research Notices*, 2012, 2012.

[7] Hironori Hojo and Shinsuke Ohba. Sp7 action in the skeleton: Its mode of action, functions, and relevance to skeletal diseases. *International Journal of Molecular Sciences*, 23(10):5647, 2022.

[8] Abdul Salam Jarrah, Reinhard Laubenbacher, Brandilyn Stigler, and Michael Stillman. Reverse-engineering of polynomial dynamical systems. *Advances in Applied Mathematics*, 39(4):477–489, 2007.

[9] John H Lagergren, John T Nardini, G Michael Lavigne, Erica M Rutter, and Kevin B Flores. Learning partial differential equations for biological transport models from noisy spatio-temporal data. *Proceedings of the Royal Society A*, 476(2234):20190800, 2020.

[10] Reinhard Laubenbacher and Brandilyn Stigler. A computational algebra approach to the reverse engineering of gene regulatory networks. *J Theor Biol*, 229(4):523–37, Aug 2004.

[11] Chaoying Li, Daryl A Scott, Ekaterina Hatch, Xiaoyan Tian, and Suzanne L Mansour. Dusp6 (mkp3) is a negative feedback regulator of fgf-stimulated erk signaling during mouse development. 2007.

[12] Shoudan Liang, Stefanie Fuhrman, and Roland Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Biocomputing*, volume 3, 1998.

[13] Santosh Manicka, Kathleen Johnson, David Murrugarra, and Michael Levin. Biological regulatory networks are less nonlinear than expected by chance. *bioRxiv*, 2021.

[14] David Murrugarra and Reinhard Laubenbacher. The number of multistate nested canalyzing functions. *Physica D: Nonlinear Phenomena*, 241(10):929–938, 5 2012.

[15] David Murrugarra, Jacob Miller, and Alex N Mueller. Estimating propensity parameters using google pagerank and genetic algorithms. *Frontiers in Neuroscience*, page 513, 2016.

[16] David Murrugarra, Alan Veliz-Cuba, Boris Aguilar, Seda Arat, and Reinhard Laubenbacher. Modeling stochasticity and variability in gene regulatory networks. *EURASIP Journal on Bioinformatics and Systems Biology*, 2012(1):5, 2012.

[17] Christoph Müssel, Martin Hopfensitz, and Hans A Kestler. BoolNet - an R package for generation, reconstruction and analysis of Boolean networks. *Bioinformatics*, 26(10):1378–1380, 2010.

[18] Aurélien Naldi, Duncan Berenguier, Adrien Fauré, Fabrice Lopez, Denis Thieffry, and Claudine Chaouiya. Logical modelling of regulatory networks with GINsim 2.3. *Biosystems*, 97(2):134–139, 2009.

[19] Vincent Noël, Marco Ruscone, Gautier Stoll, Eric Viara, Andrei Zinovyev, Emmanuel Barillot, and Laurence Calzone. Webmaboss: A web interface for simulating Boolean models stochastically. *Frontiers in Molecular Biosciences*, 8, 2021.

[20] Larissa V Ponomareva, Antony Athippozhy, Jon S Thorson, and S Randal Voss. Using ambystoma mexicanum (mexican axolotl) embryos, chemical genetics, and microarray analysis to identify signaling pathways associated

with tissue regeneration. *Comparative Biochemistry and Physiology Part C: Toxicology & Pharmacology*, 178:128–135, 2015.

[21] Brandy Stigler, Abdul Jarrah, Michael Stillman, and Reinhard Laubenbacher. Reverse engineering of dynamic networks. *Annals of the New York Academy of Sciences*, 1115(1):168–177, 2007.

[22] Jie Sun, Abd AlRahman AlMomani, and Erik Bollt. Data-driven learning of Boolean networks and functions by optimal causation entropy principle (bocse). *arXiv preprint arXiv:2006.01023*, 2020.

[23] Alan Veliz-Cuba. An algebraic approach to reverse engineering finite dynamical systems arising from biology. *SIAM Journal on Applied Dynamical Systems*, 11(1):31–48, 2012.

[24] Alan Veliz-Cuba, Boris Aguilar, Franziska Hinkelmann, and Reinhard Laubenbacher. Steady state analysis of Boolean molecular network models via model reduction and computational algebra. *BMC Bioinformatics*, 15:221, 2014.

[25] Alan Veliz-Cuba and Brandilyn Stigler. Boolean models can explain bistability in the *lac* operon. *Journal of Computational Biology*, 18(6):783–794, 2011.

[26] Daniel Wehner, Wiebke Cizelsky, Mohankrishna Dalvoy Vasudevaro, Günes Özhan, Christa Haase, Birgit Kagermeier-Schenk, Alexander Röder, Richard I Dorsky, Enrico Moro, Francesco Argenton, et al. Wnt/$\beta$-catenin signaling defines organizing centers that orchestrate growth and differentiation of the regenerating zebrafish caudal fin. *Cell reports*, 6(3):467–481, 2014.

[27] Daniel Wehner and Gilbert Weidinger. Signaling networks organizing regenerative growth of the zebrafish fin. *Trends in Genetics*, 31(6):336–343, 2015.

[28] Dominik M Wittmann, Jan Krumsiek, Julio Saez-Rodriguez, Douglas A Lauffenburger, Steffen Klamt, and Fabian J Theis. Transforming Boolean models to continuous models: methodology and application to t-cell receptor signaling. *BMC Systems Biology*, 3(1):1–21, 2009.

[29] David J Wooten, Jorge Gómez Tejeda Zañudo, David Murrugarra, Austin M Perry, Anna Dongari-Bagtzoglou, Reinhard Laubenbacher, Clarissa J Nobile, and Réka Albert. Mathematical modeling of the *Candida albicans* yeast to hyphal transition reveals novel control strategies. *PLoS Computational Biology*, 17(3):e1008690, 2021.

# A Code Usage

Here we show how the Matlab code is used. All code files and examples can be found at Github `github.com/alanavc/prototype-model`. The toy example and application are included.

The input files must be in the form of trajectories (one line per state of the system). Different trajectories must be in different files and all trajectories files must have the same prefix in the name. For example, `timecourse1.txt`, `timecourse2.txt`, `timecourse3.txt`, `timecourse4.txt`, etc. For our example in Methods, the content of timecourse1.txt would be the following.

```
0.1,1.1,1.9,0.9,0.2
0.0,0.2,0.2,0.1,0.1
0.0,1.1,0.1,1.9,2.1
```

Once we are in the folder `codes`, we can discretize the data. Prior to this we need to specify the prefix for the time course files and the file where the data will be saved.

```
> clear all
> time_course_prefix='timecourse';
> num_levels=[3,2,3,3,3];
> file_for_disc_data='data0.mat';
> % discretize data
> addpath('discretize_data')
> discretize(time_course_prefix,num_levels,file_for_disc_data)
```

Then, we generate all minimal wiring diagrams using the Matlab function `generate_wiring_diagrams`. Prior to this, we need to specify the file where we will save all wiring diagrams.

```
> %create wiring diagrams
> file_for_wiring_diagrams='WD0.mat';
> addpath('create_WD')
> generate_wiring_diagrams(file_for_disc_data,file_for_wiring_diagrams)
```

To select the best wiring diagram we use the command `select_best_wiring_diagram` as follows.

```
> %select wiring diagram
> addpath('select_WD')
> W=select_best_wiring_diagram(file_for_wiring_diagrams);
```

The variable `W` is a matrix that contains the best wiring diagram that fits the data. The user can modify this to account for prior knowledge or to try any other of the wiring diagrams (found in file `WD0.mat`) for exploration.

We use the function `generate_monotone_functions` to create the model that fits the data and has the wiring diagram given. The model will be saved as a truth table for each variable.

```
> %create model
> file_for_model='model0.mat';
> addpath('create_Model')
> generate_monotone_functions(time_course_prefix,W,file_for_model)
```

To simulate the model we use the following code.

```
> %simulate model
> addpath('simulate_model')
> init_state = [0 1 2 1 0];
> num_simulations = 100;
> num_steps = 6;
> num_vars = 5;
> % Propensity matrix
> propensity_matrix = 0.8*ones(2,num_vars);
> mean_trajectories = simulate(file_for_model,propensity_matrix,
                       init_state,num_steps,num_simulations);
```

The mean trajectories are saved in `mean_trajectories` and can be plotted using standard Matlab commands.

# B Inferred Model

Here we describe the inferred model for the data in the Applications section. We used $n = 10$ genes and the number of input variables in each Boolean function is given by the following vector:

$$nv = [2, 4, 3, 2, 2, 3, 3, 2, 2, 2].$$

The maximum number of inputs is $m = 4$. In Table 7 we list the input variables for each gene. The table is a $m$-by-$n$ matrix usually referred as $varF$. Since the number of variables may vary between different functions, only the first $nv(i)$ elements are relevant in the $i^{th}$ column of $varF$. The remaining spots in each column are set to -1 so that the whole table can be encoded as a matrix.

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ |
|---|---|---|---|---|---|---|---|---|---|
| 9 | 6 | 6 | 2 | 2 | 7 | 6 | 2 | 2 | 6 |
| 10 | 7 | 7 | 9 | 9 | 9 | 7 | 9 | 9 | 7 |
| -1 | 9 | 10 | -1 | -1 | 10 | 9 | -1 | -1 | -1 |
| -1 | 10 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

Table 7: Input variables for each gene in Eq. 2.

In Table 8 we put the truth table in compact form usually referred as $F$. This table has size $2^m$-by-$n$. Since the length of the truth tables may vary between different functions, only the first $2^{nv(i)}$ bits are relevant in the $i^{th}$ column of $F$.

The remaining spots in each column are set to -1 so that the whole table can be encoded as a matrix. The entries of the columns of $F$ are ordered so that they correspond to the binary input arrays in lexicographic order. For example, the first column with relevant entries 1,0,0,0, represents the truth table that satisfies $h(0,0) = 1, h(0,1) = 0, h(1,0) = 0, h(1,1) = 0$.

The propensities that we used for the simulations in Figures 11-12 are all equal to 0.9. That is, $p_i^{\uparrow} = p_i^{\downarrow} = 0.9$ for all $i = 1, \ldots, 10$.

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ | $x_{10}$ |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| -1 | 1 | 0 | -1 | -1 | 1 | 0 | -1 | -1 | -1 |
| -1 | 0 | 0 | -1 | -1 | 0 | 0 | -1 | -1 | -1 |
| -1 | 1 | 1 | -1 | -1 | 1 | 0 | -1 | -1 | -1 |
| -1 | 0 | 0 | -1 | -1 | 0 | 1 | -1 | -1 | -1 |
| -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

Table 8: Truth tables (in compact form) for the inferred functions for the data in the Results section.

# C Simulations with Different Propensities

In the main text we used propensities equal to 0.9 for the simulations. Here we show simulations results when all the propensities are equal to 0.5. That is, $p_i^{\uparrow} = p_i^{\downarrow} = 0.5$ for all $i = 1, \ldots, 10$.

# D Mean Squared Error

Let $Y$ is the vector of discretized data and $\hat{Y}$ is a vector of average expressions from simulated trajectories. If $Y$ and $\hat{Y}$ have the same length, we calculate the Mean Squared Error (MSE) of the predictor (or the Mean Squared Prediction Error) using the following formula:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left( Y_i - \hat{Y}_i \right)^2 .$$

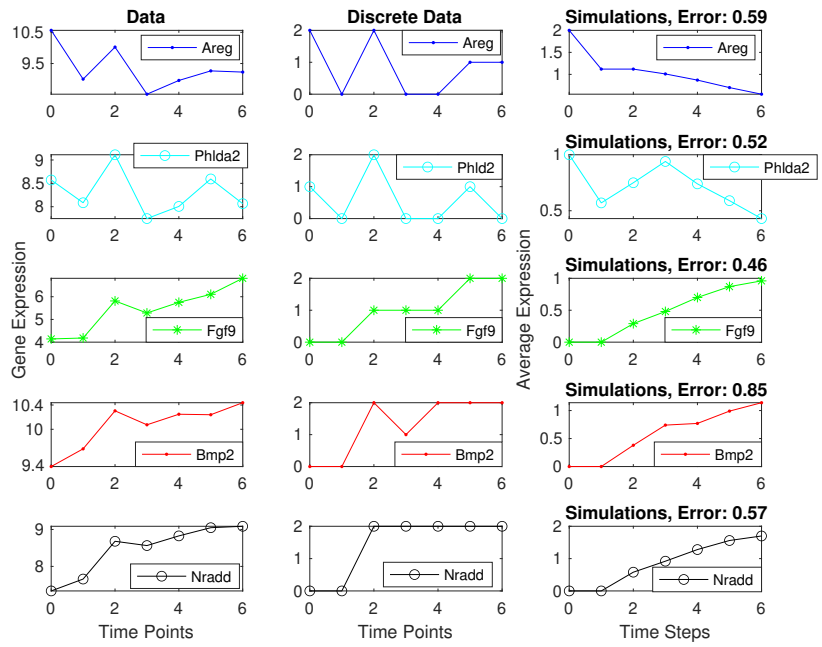



Figure 11: Gene expression data, discretized data, and simulations of the first five genes in Equation 2. The plots in the left panel are the experimental data, the ones in the middle are the discretized data, and the ones in the right panel are average expressions from simulations of 100 runs, all initialized at 1100000001. For the simulations, all the propensities are equal to 0.5 and the mean squared error is between the discretized data and simulated trajectories.

Note that the MSE is the "mean" $\left(\frac{1}{n}\sum_{i=1}^{n}\right)$ of the "squares of the errors" $\left(Y_i - \hat{Y}_i\right)^2$.

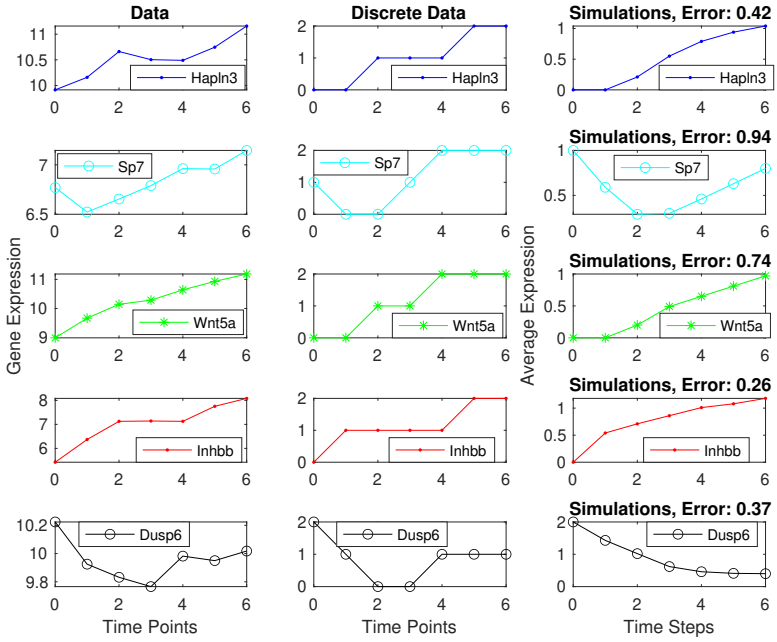



Figure 12: Gene expression data, discretized data, and stochastic simulations of the last five genes in Equation 2. The plots in the left panel are experimental data, the ones in the middle are discretized data, and the ones in the right panel are average expressions from simulations of 100 runs, all initialized at 1100000001. For the simulations, all the propensities are equal to 0.5 and the mean squared error is between the discretized data and simulated trajectories.