Unsupervised logic-based mechanism inference for network-driven biological processes

Martina Prugger[1,2], Lukas Einkemmer[3], Samantha P. Beik[2], Leonard A. Harris[2,4], Carlos F. Lopez[2,5],

1 Department of Biochemistry, University of Innsbruck, Innsbruck, Tyrol, Austria
2 Department of Biochemistry, School of Medicine, Vanderbilt University, Nashville, TN, USA
3 Department of Mathematics, University of Innsbruck, Innsbruck, Tyrol, Austria
4 Currently at Department of Biomedical Engineering, University of Arkansas, Fayetteville, AR, USA
5 Department of Biomedical Informatics, Vanderbilt University Medical Center, Nashville, TN, USA

* to whom correspondence should be addressed: Carlos F. Lopez (c.lopez@vanderbilt.edu)

Abstract

Modern analytical techniques enable researchers to collect data about cellular states, before and after perturbations. These states can be characterized using analytical techniques, but the inference of regulatory interactions that explain and predict changes in these states remains a challenge. Here we present a generalizable unsupervised approach to generate parameter-free, logic-based mechanistic hypotheses of cellular processes, described by multiple discrete states. Our algorithm employs a Hamming-distance based approach to formulate, test, and identify, the best mechanism that links two states. Our approach comprises two steps. First, a model with no prior knowledge except for the mapping between initial and attractor states is built. Second, we employ biological constraints to improve model fidelity. Our algorithm automatically recovers the relevant dynamics for the explored models and recapitulates all aspects of the original models biochemical species concentration dynamics. We then conclude by placing our results in the context of ongoing work in the field and discuss how our approach could be used to infer mechanisms of signaling, gene-regulatory, and any other input-output processes describable by logic-based mechanisms.

Introduction

A mechanistic understanding of dynamic cellular processes is at the core of multiple areas of research including molecular cell biology, physiology, biophysics, and bioengineering [1–6]. Although analytical tools have improved the breadth and depth with which intra- or extra-cellular biochemical processes are explored [7–9], the vast majority of available data is limited to experiments that probe cue-response relationships with a specified set of inputs and outputs. Although significant efforts have been devoted to understand how biochemical interactions link these inputs and outputs, the formulation of mechanistic hypotheses remains a challenging problem which is essential to explain and predict cellular responses to perturbations [10–15].

The Boolean logic formalism, introduced by Kauffmann in 1969 [16] is a simple yet powerful approach to describe gene-regulatory networks, signaling networks, metabolic networks, and many others [17, 18]. In this representation, each node in a network corresponds to a gene or gene-product while each edge corresponds to a Boolean rule or set of rules that describes the interaction between nodes. The system can evolve for a number of discrete steps, where the state of each node (one or zero) is determined by evaluating its associated logic rules at each step. The system is typically evolved for a number of steps using a Markov-chain process until a steady state (aka attractor state) is achieved [19]. These Boolean representations of biochemical reaction networks have yielded important biochemical insights [20–25] and offer a parameter-free alternative to other formalisms where exact parameters may be difficult or impossible to acquire [26, 27].

Despite the mathematical simplicity of Boolean logic based biochemical networks, the interaction rules that dictate the dynamics cannot be directly obtained from either experimental data or curated interaction databases [28–30]. For this reason, logic rules enumeration, which comprise a specific mechanism of action, remains a central challenge in Boolean logic modeling. This problem can be found in all areas of biology as well as other areas (e.g. ecology,

control theory) where Boolean modeling is employed. Therefore, our goal is to propose a rigorous methodology to automatically generate Boolean rules, given input and output states, and generate mechanistic hypothesis to link network states within biological constraints in an unsupervised manner.

Assembling a Boolean-logic based model from experimental data is most commonly done manually, requiring inference of both the network structure and the Boolean rules. For that purpose data from various sources, including time series data, can be used [31–33]. The attractors obtained from these formalisms are therefore model predictions, given a specified mechanism rather than the inverse problem of formulating a mechanism for a set of observables [31, 33]. Although it is desirable to preserve experimentally observed attractors, there is no guarantee for such models that a given initial state necessarily evolves towards the correct steady states [34]. Enforcing such constraints manually is often a tedious and error-prone process. The choice of updating scheme chosen for model evolution can significantly affect the interpretation of model dynamics. For example, synchronous updating schemes may yield network dynamics with no clear biological interpretation [35–38]. By contrast, sequential node updating schemes, such as General Asynchronous, can provide a mechanism with better biochemical correlation [22, 39–42].

In this work, we address the problems of mechanism inference in biological processes where input states and attractors are known but the mechanism is unknown. The proposed algorithm constructs both candidate network structure and the corresponding Boolean rules in an unsupervised manner. Our method guarantees that the selected initial states reach their designated steady states, that no spurious steady states are introduced, and that the network logic inferred is compatible with the biological relevant asynchronous updating [34]. In addition, experimentally-observed probability distributions from one initial state to multiple attractor states are preserved by our algorithm – often a biologically important observation. Our algorithm can thus be used for hypothesis exploration, model identification, and mechanism exploration in silico in the context of complex experimental data.

Methods & Results

The main idea of the proposed algorithm is shown in Figure 1. As input a mapping between each initial state and the corresponding steady states is given. In an asynchronous update, as we consider here, the state of only one species can be changed per step. This means that the Hamming distance of all states that are reachable in the next step is equal to one [43]. We exploit this knowledge to construct paths from each initial value to the reachable steady states, while avoiding paths that lead to incorrect results. This allows us to generate (in general many) candidate networks that satisfy precisely the prescribed mapping. The probability distribution of the steady states can also be specified. This is then used, within a genetic optimization algorithm, to select models which show the same dynamics. At this point, expert knowledge on the network (such as on which species a given rule depends on or specific transitions that should be included) can be incorporated as well. A number of good candidate models are then selected and the corresponding Boolean rules are generated. The algorithm automatically simplifies these results using symbolic manipulation. The algorithm proposed is described in the following sections for two examples: an Enzyme-Substrate kinetics model and an established Epithelial to Mesenchymal Transitions (EMT) epithelial mouse cancer cell metastasis [44, 45].

The basic algorithm is explained in some detail for the Enzyme-Substrate kinetics reaction mechanism which is facilitated by the smaller size of that particular problem. However, all the steps in the algorithm (except for the problem specific expert guidance that can be used) have been fully automated and are part of a parallelized hybrid Python/C++ code. Thus, the creation of the Boolean rules is done fully automatically and the detailed enumeration of some of those steps for the Enzyme-Substrate kinetics reaction mechanism problem are only provided as examples to gain a better understanding of the algorithm. For both examples we show how to incorporate expert knowledge into our method. This by necessity is problem dependent and thus different approaches, that should generalize well to many other problems, are explored.

Network inference for Enzyme-Substrate dynamics

We employ an enzyme-substrate reaction system to demonstrate the details of our approach. In this representation E is the enzyme, S is the substrate, and P is the resulting product. The enzyme can bind to the substrate into the complex ES via a specific rate k_f and break up into the two species via the rate k_r or catalyze the substrate-to-product reaction, resulting in free enzyme and product according to the chemical equation

$$E + S \underset{k_r}{\overset{k_f}{\rightleftharpoons}} ES, \qquad ES \xrightarrow{k_c} E + P.$$

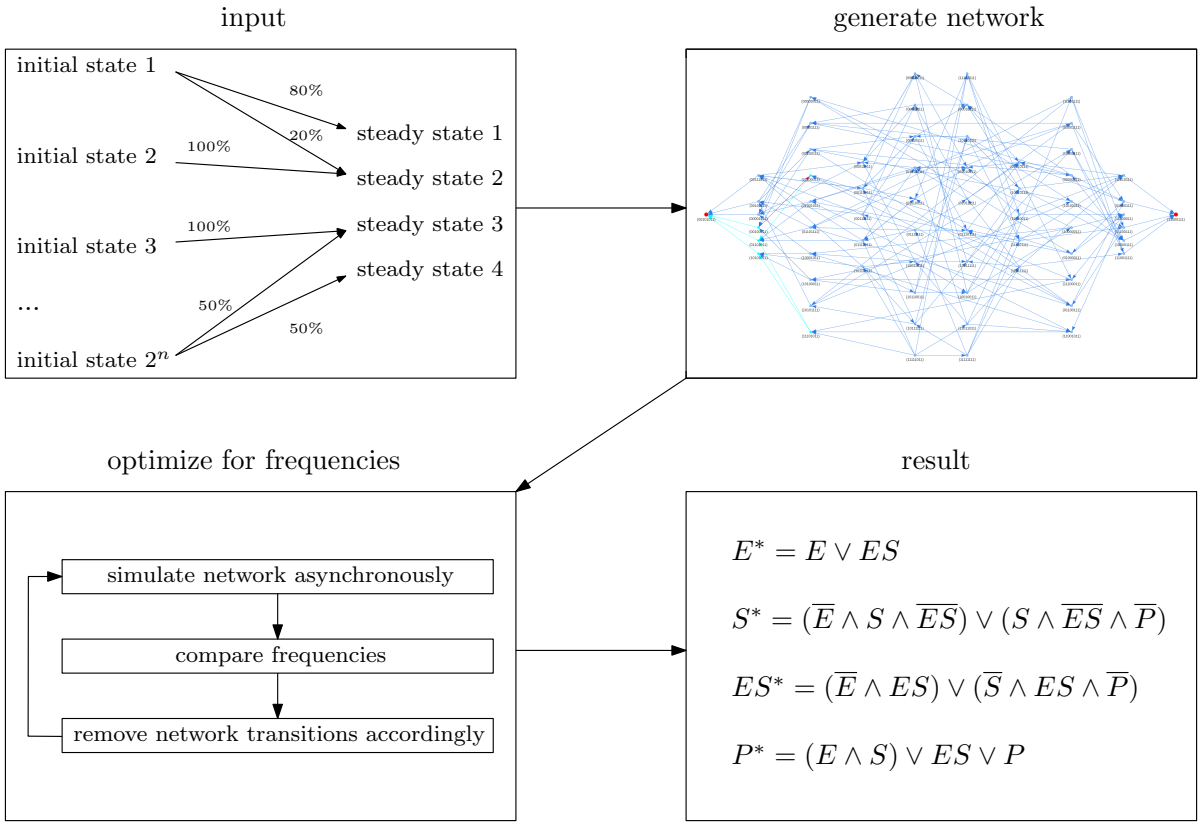

Fig 1. Schematic depiction of our workflow. Our input data is of the form initial state and corresponding steady states. If the same initial state is observed to end up in multiple steady states, a probability to reach each steady state can be prescribed as well. From the initial state to attractor relationship, a network is created, taking into account every possible connection allowed by an asynchronous updating scheme. We then simulate the network and compare the resulting probabilities to the specified measurement data. If necessary, we remove transitions from the network to achieve a better match between the probabilities of the resulting network and the the experimental data. The result is a system of Boolean rules that describes the network dynamics.

Mathematically, this results in a system of ordinary differential equations (ODEs) with species concentration E, S, ES, and P as well as the three parameters k_f, k_r, and k_c.

Our goal is to model the corresponding dynamics using a Boolean network. Boolean networks assume that the species are either present (1) or absent (0), i.e. $E, S, ES, P \in \{0, 1\}$, and that all reactions are equally likely, i.e. all rate constants are equal to 1. To match this we will also make the assumption that $k_f = k_r = k_c = 1$ in our enzyme-substrate reaction. For the concentrations we will start with either 1 or 0 for each species, but the concentration is allowed to take on fractional values as the reaction dynamics evolve. The results of such a simulation are shown in Figure 2. The initial value to steady state mapping so obtained will be used in this section to automatically construct a Boolean network using the proposed algorithm. The goal of this Boolean network is to recover the dynamics of the ODE simulation. Let us note that in the Boolean model the values of the species are necessarily either 1 or 0. We will, however, interpret the average of the stochastic asynchronous update as a (relative) concentration similar to the one found in the ODE model.

For this particular system, we have four species each of which can take on two conditions, for a total of $2^4 = 16$ possible states of the system, namely:

$$\begin{aligned}(E, S, ES, P) = \{&(0,0,0,0), (0,0,0,1), (0,0,1,0), (0,0,1,1), (0,1,0,0), (0,1,0,1),\\ &(0,1,1,0), (0,1,1,1), (1,0,0,0), (1,0,0,1), (1,0,1,0), (1,0,1,1),\\ &(1,1,0,0), (1,1,0,1), (1,1,1,0), (1,1,1,1)\}.\end{aligned}$$

For consistency, each tuple represents the species in the order as shown (i.e. the first entry is the E state, the second entry is the S state, the third is the ES state, and the fourth entry is the P state). Once all states have been

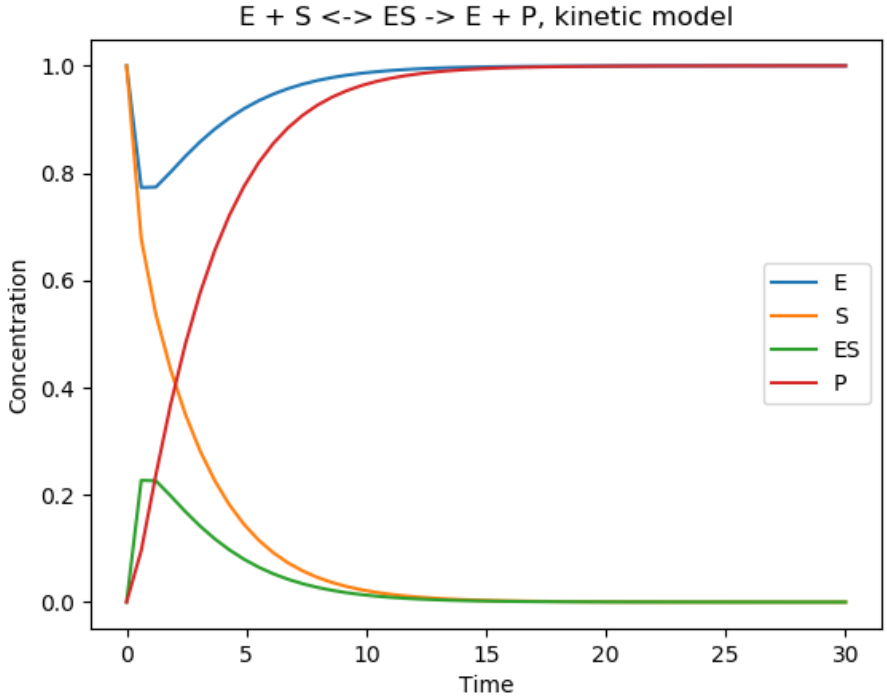

Fig 2. Reaction kinetics of an enzyme-substrate system with rate constants $k_f = k_r = k_c = 1$ and initial concentrations $E = S = 1$ and $ES = P = 0$. The simulation is the solution of the underlying ordinary differential equation, and the concentrations of the species are therefore still $\in \mathbb{R}_{[0,1]}$. The dynamics depicted in this graph is considered to be the underlying truth that our algorithm tries to recreate with an automatically generated Boolean logic network.

defined we can analyze the states for biochemical significance. For example, state $(0,0,0,0)$ signifies that no species are present in the system and therefore no chemical reactions can occur. States $(1,0,0,0)$, $(0,1,0,0)$, and $(0,0,0,1)$ similarly have only one of enzyme, substrate, or product present and similarly no chemical reactions can take place. Absence of reactions is also seen in the state $(0,1,0,1)$, since substrate and product do not interact with each other. Removing those states form further considerations leads to a network that treats them as a steady state which can not be accessed by any other state. All other states, however, converge towards the attractor state $(1,0,0,1)$. The first step of our introduced method is to match all initial states to their according steady state represented in Table 1.

Steady State	Initial States
$(0,0,0,0)$	$(0,0,0,0)$
$(0,0,0,1)$	$(0,0,0,1)$
$(0,1,0,0)$	$(0,1,0,0)$
$(0,1,0,1)$	$(0,1,0,1)$
$(1,0,0,0)$	$(1,0,0,0)$
$(1,0,0,1)$	$(0,0,1,0)$, $(0,0,1,1)$, $(0,1,1,0)$, $(0,1,1,1)$, $(1,0,0,1)$, $(1,0,1,0)$, $(1,0,1,1)$, $(1,1,0,0)$, $(1,1,0,1)$, $(1,1,1,0)$, $(1,1,1,1)$

Table 1. Mapping of the initial states to their corresponding steady state. By mapping states to themselves, they create a steady state for the network that can not be accessed by any other state. Note, that in principle, initial states can converge towards multiple different steady states. This behavior is captured easily by just adding these states to all of the corresponding steady state lists.

In the next step, for each attractor, the initial states are sorted according to their Hamming distance from the steady state. The sorting for steady state $(1,0,0,1)$ is listed in Table 2.

Based on the asynchronous updating scheme, a state with Hamming distance n will require at least n updates to reach the attractor. We then identify the transitions necessary to build a pathway for each Hamming distance level.

Hamming distance from steady state $(1,0,0,1)$	Initial States
1	$(1,0,1,1)$, $(1,1,0,1)$
2	$(0,0,1,1)$, $(1,0,1,0)$, $(1,1,0,0)$, $(1,1,1,1)$
3	$(0,0,1,0)$, $(0,1,1,1)$, $(1,1,1,0)$
4	$(0,1,1,0)$

Table 2. Sorting of the initial states according to their Hamming distance from the steady state $(1,0,0,1)$.

We use this information to create a transition map for each species that contains the necessary transformations to reach a given attractor. We achieve this by working our way backwards from each attractor. For example, for a level one (i.e. Hamming distance = 1) transition, the state $(1,0,1,1)$ needs to flip the third bit (the bit for ES) to reach the attractor $(1,0,0,1)$. Similarly the state $(1,1,0,1)$ needs to flip the second bit (the bit for S) to reach the attractor $(1,0,0,1)$. Therefore, the transition lists for S and ES will be updated with the states $(1,1,0,1)$, and $(1,0,1,1)$ respectively. We do the same for level 2 (i.e. states with Hamming distance=2), as well as for all other levels and extend the lists accordingly. The full sorting can be found in Table 3. Note, that for a system with multiple attractors, each attractor gets a similarly created list.

species	list for $d=1$	list for $d=2$	list for $d=3$	list for $d=4$
E		$(0,0,1,1)$	$(0,0,1,0)$, $(0,1,1,1)$	$(0,1,1,0)$
S	$(1,1,0,1)$	$(1,1,1,1)$	$(0,1,1,1)$, $(1,1,1,0)$	$(0,1,1,0)$
ES	$(1,0,1,1)$	$(1,1,1,1)$	$(1,1,1,0)$	
P		$(1,0,1,0)$, $(1,1,0,0)$	$(0,0,1,0)$, $(1,1,1,0)$	$(0,1,1,0)$

Table 3. List of transitions for each species that make up the network pathways sorted by their Hamming distance d to the steady state $(1,0,0,1)$. Note, that a state can reach the steady state in multiple ways. It is therefore possible to have the same initial assigned to multiple species.

A graphic representation for the corresponding pathways can be found in the transition graphs Figures S1 and S2 in the supplementary material.

This list includes all the necessary transitions for each species to reach a given attractor. In a system with multiple steady states, this algorithm has to be performed for each attractor.

With the transition list, we can then infer the Boolean rules to update each species according to its associated transition list. This is done by translating every transition state via the logical operator AND ($\wedge$) and connecting each of the transition states from Table 3 via a the logical OR ($\vee$) operator. An exclusive XOR ($\veebar$) with the active species ensures that the species activates only with the states from the given list. In our example, the inferred Boolean rules are written as:

rule for E:

$$((\overline{E} \wedge \overline{S} \wedge ES \wedge P) \vee (\overline{E} \wedge \overline{S} \wedge ES \wedge \overline{P}) \vee (\overline{E} \wedge S \wedge ES \wedge P) \vee (E \wedge S \wedge ES \wedge \overline{P})) \veebar E,$$

which simplifies to:

$$E \vee ES$$

The same procedure for the other species results in the following rules

rule for S:

$$((E \wedge S \wedge \overline{ES} \wedge P) \vee (E \wedge S \wedge ES \wedge P) \vee (\overline{E} \wedge S \wedge ES \wedge P) \vee (E \wedge S \wedge ES \wedge \overline{P}) \vee (\overline{E} \wedge S \wedge ES \wedge \overline{P})) \veebar S = (\overline{E} \wedge S \wedge \overline{ES}) \vee (S \wedge \overline{ES} \wedge P)$$

rule for ES:

$$((E \wedge \overline{S} \wedge ES \wedge P) \vee (E \wedge S \wedge ES \wedge P) \vee (E \wedge S \wedge ES \wedge \overline{P})) \veebar ES = (\overline{E} \wedge ES) \vee (\overline{S} \wedge ES \wedge \overline{P})$$

rule for P:

$$((E \wedge \overline{S} \wedge ES \wedge \overline{P}) \vee (E \wedge S \wedge \overline{ES} \wedge \overline{P}) \vee (\overline{E} \wedge \overline{S} \wedge ES \wedge \overline{P}) \vee (E \wedge S \wedge ES \wedge \overline{P}) \vee (\overline{E} \wedge S \wedge ES \wedge \overline{P})) \veebar P = (E \wedge S) \vee ES \vee P$$

Dynamic behavior of the forward-only (ES-F) network

The described way to obtain Table 3 only includes the transitions in one direction (from initial state to steady state) and therefore qualifies as "forward-only" network that does not allow for transitions that step backwards away from the steady state. The automatically created network evaluating forward-only interactions results in the following set of update rules

$$\begin{aligned} E^* &= E \vee ES \\ S^* &= (\overline{E} \wedge S \wedge \overline{ES}) \vee (S \wedge \overline{ES} \wedge \overline{P}) \\ ES^* &= (\overline{E} \wedge ES) \vee (\overline{S} \wedge ES \wedge \overline{P}) \\ P^* &= (E \wedge S) \vee ES \vee P. \end{aligned}$$

The generated network structure of the forward network is depicted in Figure 3 top left. Using the General Asynchronous updating scheme [39, 40, 42] for the initial state $(1, 1, 0, 0)$ for 100 simulations yields the result shown in Figure 3 on the top right. The depicted initial condition is the same as we have used for the kinetic simulation in Figure 2 to compare and judge the quality of our result.

The kinetic model in Figure 2 depicts concentrations of the species in the system, i.e, $E, S, ES, P \in \mathbb{R}_{\geq 0}$. Since this is not possible for a Boolean simulation, where $E, S, ES, P \in \{0, 1\}$, to capture the overall dynamics of the network, multiple simulations have to be performed. The random nature underlying the General Asynchronous updating scheme results in different pathways taken by each simulation. Looking for each species at the fraction of how many simulations are 0 or 1 at each simulation step allows us to capture dynamics similar to the kinetic description.

As we can see, the correct steady state is achieved. However, in these dynamics the enzyme never gets bound to the substrate, but substrate is directly convert into the product. The middle part of the enzyme-substrate kinetics is thus omitted. This is clearly not the desired dynamics.

A look at the graphic representation of the network as depicted in Figure 3, top left, gives us further insight into this problem. We can see that by our basic construction, we only allow the direct "forward" pathway for the network: The state $(1, 0, 0, 0)$, i.e., the state where substrate is consumed has not been included into the network and therefore, the only possibility of our initial state to change is the creation of the product into the state $(1, 1, 0, 1)$. This state has also no other path than directly go to the final state $(1, 0, 0, 1)$, i.e., consume the substrate. However, no circulation or other dynamics are allowed in this network. Michaelis Menten kinetics, however, are only a simple example that already demonstrates, that circulation within the pathways are an important biological factor of networks.

We therefore propose to extend our method to include the backward pathways into the network as well.

Backward dynamic paths to enable dynamic loops (ES-B)

The list from Table 3 only accounts for transitions in the forward direction towards the attractor. By extending the corresponding lists to include the backward transitions, we generate a new rule-set that extends to a new network including all possible backward pathways as well. Note, however, that we can only include backward pathways starting with level 1 and higher. Including a backward path for level 0, i.e., the attractor, means that the simulation can leave this state and it therefore would be no longer be a steady state.

Using the extended transition sets and the same translation into updating rules, we obtain

$$\begin{aligned} E^* &= (E \wedge \overline{ES}) \vee (\overline{E} \wedge ES) \\ S^* &= (\overline{S} \wedge ES) \vee (\overline{E} \wedge S \wedge \overline{ES}) \vee (S \wedge \overline{ES} \wedge \overline{P}) \\ ES^* &= (\overline{E} \wedge ES) \vee (E \wedge S \wedge \overline{ES}) \vee (\overline{S} \wedge ES \wedge \overline{P}) \\ P^* &= (ES \wedge \overline{P}) \vee (E \wedge S \wedge \overline{P}) \vee (\overline{E} \wedge \overline{ES} \wedge P) \vee (\overline{S} \wedge \overline{ES} \wedge P). \end{aligned}$$

The network structure and simulation results for this rule-set and the initial state $(1, 1, 0, 0)$ is depicted in the middle row of Figure 3 on the left.

The network depiction now demonstrates, that all paths in the network are now enabled. I.e., the state $(1, 1, 0, 0)$ can also transition into $(1, 1, 1, 0)$, which means that the product ES is generated. We observe, that the simulation

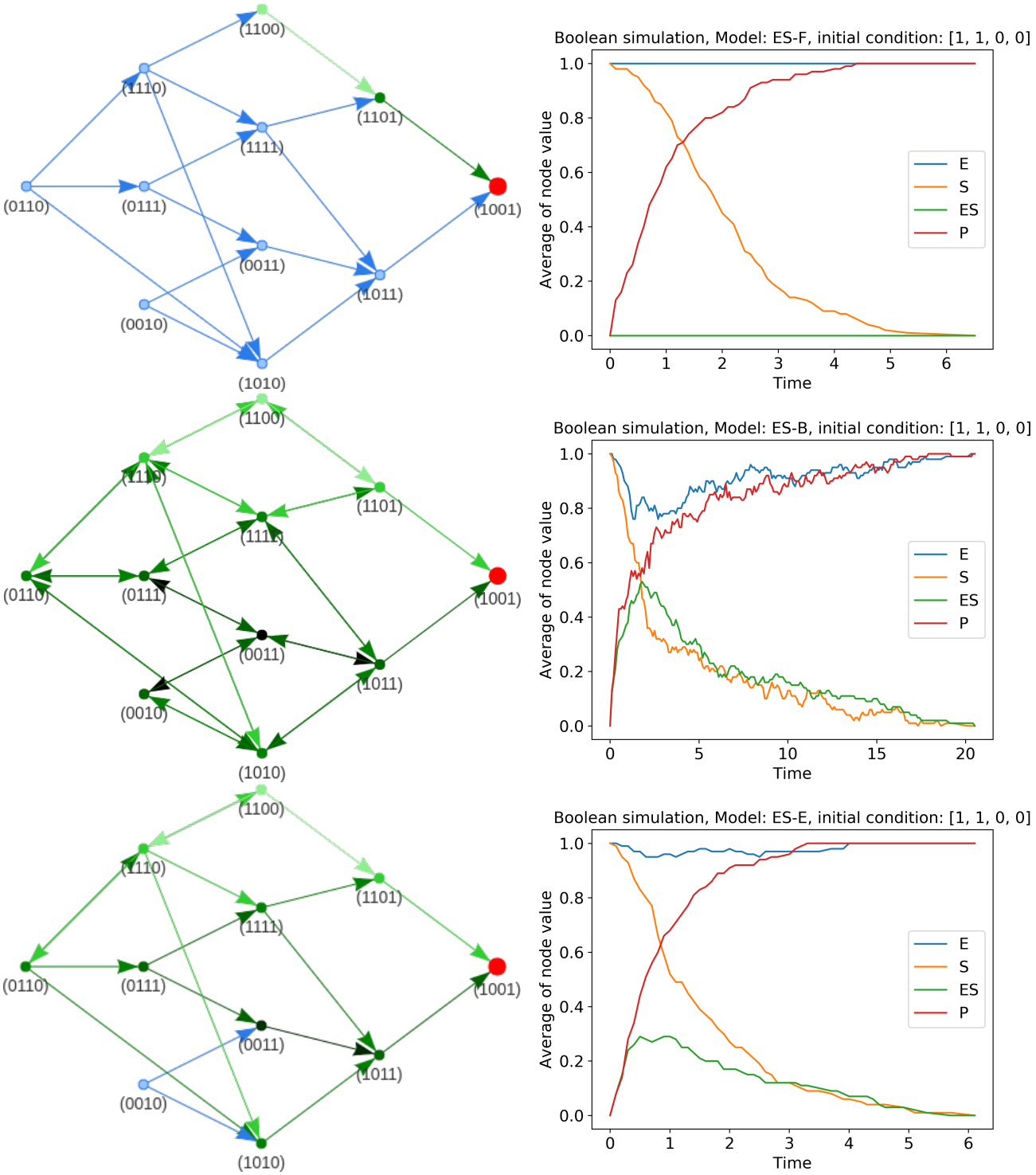

Fig 3. Boolean networks created by our proposed rules generator (left) for the enzyme-substrate mechanics and their resulting dynamics after 100 asynchronous updating simulations for the initial state $(E, S, ES, P) = (1, 1, 0, 0)$ (right). The first row depicts the resulting network from the forward-only pathways network (ES-F), the second row the network including backward paths (ES-B), and the third row the network including expert knowledge (ES-E). For the network representation, the pathway for state $(1, 1, 0, 0)$ to reach the steady state is highlighted in green. The darker the arrow, the more simulation steps are necessary to reach this particular transition in principle. For the dynamics plots, the fraction of 100 simulations of the asynchronous updating scheme that are on/off is shown on the y-axis. The x-axis represents the number of simulation steps it takes to reach the steady state.

now both consumes enzyme and creates the complex ES before the attractor is reached. The dynamics, depicted in the middle row of Figure 3 on the right, match the ground truth from Figure 2 well. We can also see, that it takes noticably longer for all simulations to reach their overall attractor. This makes sense, since backward paths also enable simulations to loiter in loops.

Reduction of logic-rule search space with experimental data (ES-E)

Both networks described above are created by automatically mapping initial states to their corresponding attractor without any additional knowledge. Due to the construction of our method, however, it is straightforward to include expert knowledge into the dynamics as well.
Let us for example look back at the construction of our first network. We have noted that in this case we omit the pathway for the creation of the complex ES. We are, however, aware that this part is a necessary step of the dynamics. In this example, we therefore propose to start with the forward-pathway network and add the transitions for $(1,1,0,0)$ to $(1,1,1,0)$, as well as the resulting consumption of E, namely the transition from $(1,1,1,0)$ to $(0,1,1,0)$ to the corresponding transition list.
The resulting ruleset is

$$\begin{aligned} E^* &= (ES \wedge P) \vee (E \wedge \overline{ES}) \vee (\overline{E} \wedge ES) \vee (\overline{S} \wedge ES) \\ S^* &= (\overline{E} \wedge S \wedge \overline{ES}) \vee (S \wedge \overline{ES} \wedge \overline{P}) \\ ES^* &= (\overline{E} \wedge ES) \vee (\overline{S} \wedge ES \wedge \overline{P}) \vee (E \wedge S \wedge \overline{ES} \wedge \overline{P}) \\ P^* &= (E \wedge S) \vee ES \vee P \end{aligned}$$

and in Figure 3 bottom, we see the resulting network (left) and the corresponding simulation for the initial state $(1,1,0,0)$ (right). Since we only added the absolute minimum necessary to create ES, most of the loops from the backward pathways model are omitted and the simulation reaches the steady state in a similar time frame as the simulation with the forward pathways only while also capturing some of the dynamics of the ES creation and E consumption.
Note, that in this case we manually added transitions to the network we judged feasible. We also provide the option to exclude transitions that the user is certain are biologically unfeasible.
Our implementation enables the user to start with either the forward-path network, or the full backward path including network from which transitions can be added or removed as seen fit. Note, however, that not all removals are valid to keep the network dynamics: adding and/or removing random transitions could result in the following problems:

1. adding a transition that leads directly away from the attractor will result in a loss of this attractor as a steady state
2. adding a transition could create a pathway to the wrong attractor
3. removing a transition could make it impossible for a state to reach its attractor

Since the full backward path network includes all possible pathways between all nodes in the network, for an unknown process, we recommend to start with the full backward path network and start strategically removing transitions from there. This way, we can be certain that the necessary network connections are present, while we only need to assure that point 3. of the list is not violated. Note, however, that due to the many loops that are created in this network, more steps are required by the asynchronous updating scheme before equilibrium is achieved.

Application to an established model: Epithelial to Mesenchymal Transitios (EMT) in cancer cell metastasis

To demonstrate our mechanism inference approach in a real-world system, we infer the Boolean logic mechanism for the EMT transition observed in [44, 45]. The ruleset for the reference EMT model is:

$$
\begin{aligned}
\text{NICD}^* &= \text{Notch} \wedge \overline{\text{TP63\_TP73}} \wedge \overline{\text{TP53}} \\
\text{Notch}^* &= \text{ECM} \wedge \overline{\text{miRNA}} \\
\text{TP53}^* &= (\text{DNAdam} \vee \text{NICD} \vee \text{miRNA}) \wedge \overline{\text{EMTreg}} \wedge \overline{\text{TP63\_TP73}} \\
\text{TP63\_TP73}^* &= \text{DNAdam} \wedge \overline{\text{miRNA}} \wedge \overline{\text{NICD}} \wedge \overline{\text{TP53}} \\
\text{miRNA}^* &= (\text{TP53} \vee \text{TP63\_TP73}) \wedge \overline{\text{EMTreg}} \\
\text{EMTreg}^* &= \text{NICD} \wedge \overline{\text{miRNA}} \\
\text{ECM}^* &= \text{ECM} \\
\text{DNAdam}^* &= \text{DNAdam}
\end{aligned}
$$

The system comprises six species NICD, Notch, TP53, TP63_TP73, miRNA, and EMTreg. ECM and DNAdam are input parameters that do not change during the simulation. For example, the model has been used in [44] to investigate the effect of Notch upregulation and TP53 deletion. The model captures the EMT dynamics triggered by TP53 deletion and Notch activation, and the interplay between multiple interactions that lead to mesenchymal behavior in epithelial mouse cells. Such mechanism are also common in many other cancers.
Let us now assume, that this set of rules is the underlying truth for our mechanism inference algorithm, and therefore refer to the model as EMT-O (original). We have generated reference data by running the asynchronous updating simulator 100 times for each of the $2^8 = 256$ states and recorded the steady state for each run. We use this data, to automatically infer a set of rules.
Table 4 summarizes the so obtained results.

(ECM,DNAdam)	SS	frequency
(0,0)	(0,0,0,0,0,0,0,0)	0.61
	(0,0,1,0,1,0,0,0)	0.39
(0,1)	(0,0,1,0,1,0,0,1)	1.0
(1,0)	(0,0,1,0,1,0,1,0)	0.52
	(1,1,0,0,0,1,1,0)	0.48
(1,1)	(0,0,1,0,1,0,1,1)	0.79
	(1,1,0,0,0,1,1,1)	0.21

Table 4. Summary of the asynchronous updating results for the EMT model from [44] after 100 asynchronous updating simulations for each of the 256 possible initial states. The order of species is (NICD, Notch, TP53, TP63_TP73, miRNA, EMTreg, ECM, DNAdam). The network splits into four mutually exclusive sub-networks depending on the two parameters ECM and DNAdam. Initial states for three of those sub-networks can run into two different attractors with varying frequency.

Since ECM and DNAdam are parameters and can not change during the simulation, the network naturally divides into four mutually exclusive sub-networks depending on the parameter inputs

$$(\text{ECM}, \text{DNAdam}) \in \{(0,0), (0,1), (1,0), (1,1)\}.$$

Each of those sub-networks contains the 64 states with the corresponding parameters fixed in each. In three of the four networks, the data suggests that initial states can run into two different steady states. A closer inspection reveals, that the system has in total three different attractors: $(\text{NICD}, \text{Notch}, \text{TP53}, \text{TP63\_TP73}, \text{miRNA}, \text{EMTreg}) = (0,0,0,0,0,0)$, $(0,0,1,0,1,0)$, or $(1,1,0,0,0,1)$. Note, however, that for our method we treat the parameters as species and therefore work with seven different attractors. A schematic depiction of the resulting four sub networks divided by the corresponding parameter set can be found in the file `states_to_remove.pdf`. The input file for our method consists of a list for every one of the 256 states and their corresponding attractors.
We can now run the Boolean rules generator described in the previous section. Due to the many species involved in this system, we do not expect our method to produce rules that are short and easily understandable (at least not without additional constraints). However, they are created automatically and we could immediately put the generated rule-set into the asynchronous simulator and analyze the inferred model.

In the supplemental Figures S3, S4, S5, and S6, we depict the four generated subnetworks determined by the four different parameter options generated by our method. The blue lines are transitions that are captured in the original model (EMT-O) [44] as well as our forward path model (EMT-FW). The orange lines are transitions that are captured by EMT-FW but not EMT-O. The green lines are transitions that are not captured by EMT-FW, but are added by the backward model (EMT-BW). We want to point out, that EMT-O introduced some states that are part of a dual-attractor network, but only reach one of the steady states (denoted with the bright blue lines). Our implementation recognizes those states and successfully treats them in the same way.
In the supplementary Figure S4, we can observe the difference between EMT-FW and EMT-BW for a network with a single steady state more clearly. Due to the large number of species, the inter-connectivity between the states results in backward transitions even in EMT-FW (thus a forward connection for one rule can act as the backward connection for another). The difference between the two networks only occurs between level 1 and level 2 of the distance to the steady state. Note, that for some states, EMT-O includes these transitions that are not captured by EMT-FW.

Comparison between data and automatically created networks

Since the construction of EMT-BW includes all possible transitions under the asynchronous updating scheme, we need to remove particular transitions to recapture EMT-O. In the supplementary file `states_to_remove.pdf`, we list the full list of transitions to be removed from EMT-BW to capture the original EMT model, and depict the resulting network graph separated by the corresponding parameter set. In Table 5, we summarize the transitions by counting how many of them to remove from EMT-BW to obtain the original model.

(ECM,DNAdam)	NICD	Notch	TP53	TP63_TP73	miRNA	EMTreg
(0,0)	32|61	32|61	32|56	32|56	28|56	24|57
(0,1)	32|63	32|63	32|63	28|63	36|63	24|63
(1,0)	32|62	32|62	32|59	32|59	28|59	24|59
(1,1)	32|60	32|62	32|57	28|57	32|60	24|57
	52.03%	51.61%	54.47%	51.06%	52.1%	40.68%

Table 5. Summary of transitions assigned to each network. Each species in each sub-network has a number of transitions. The number on the left is the number of transitions for EMT-O, the number on the right is the number of transitions for the backward pathway model generated by our tool. The last row denotes the percentage for all four sub-networks in total.

The numbers on the left denote the number of transitions for EMT-O, while the numbers on the right are the corresponding numbers of transition we obtain with the backward rule generator. These numbers confirm our suspicion, that our generated rules include about twice as many transitions as EMT-O.

Let us now look at some simulation results. In Figure 4, we depict the dynamics of two initial states (NICD,Notch,TP53,TP63_TP73,miRNA,EMTreg,ECM,DNAdam) $= (0,0,0,0,0,0,1,0)$, and $(0,1,0,0,0,1,0,0)$ for the original model (row 1), and EMT-BW (row 2). For the first initial condition, EMT-O reaches the steady state $(1,1,0,0,0,1,1,0)$ more than half the time and the steady state $(0,0,1,0,1,0,1,0)$ less than half the time. For EMT-BW this dynamics is exactly reversed. For the second initial state reaching the attractor $(0,0,1,0,1,0,0,0)$ is significantly less likely in EMT-O than in EMT-BW, but qualitatively the correct behavior is obtained.

To get a broader view, we extended Table 4 by the according frequencies for our automatically created systems in Table 6. We can see that the subnetwork $(\text{ECM}, \text{DNAdam}) = (1,0)$ experiences a trend towards steady state $(1,1,0,0,0,1,1,0)$ for the systematic approach, where for EMT-O, both steady states of the subnetwork are reached about the same number of times. For the other three subnetworks, the frequencies to enter into a given steady state generally match well with EMT-O.

Automated model selection

Our automatic rule-creation method does not take the time evolution of the system into account. It is therefore not surprising, that our results experience some qualitative differences between the input model and our created models. If, however, a model predicts the majority of a cell fate as death, when the experiment clearly states the majority as survival, the model is very restricted in its usefulness. In this section, we therefore propose an automatic model selection algorithm to find a model that agrees better with the underlying data.

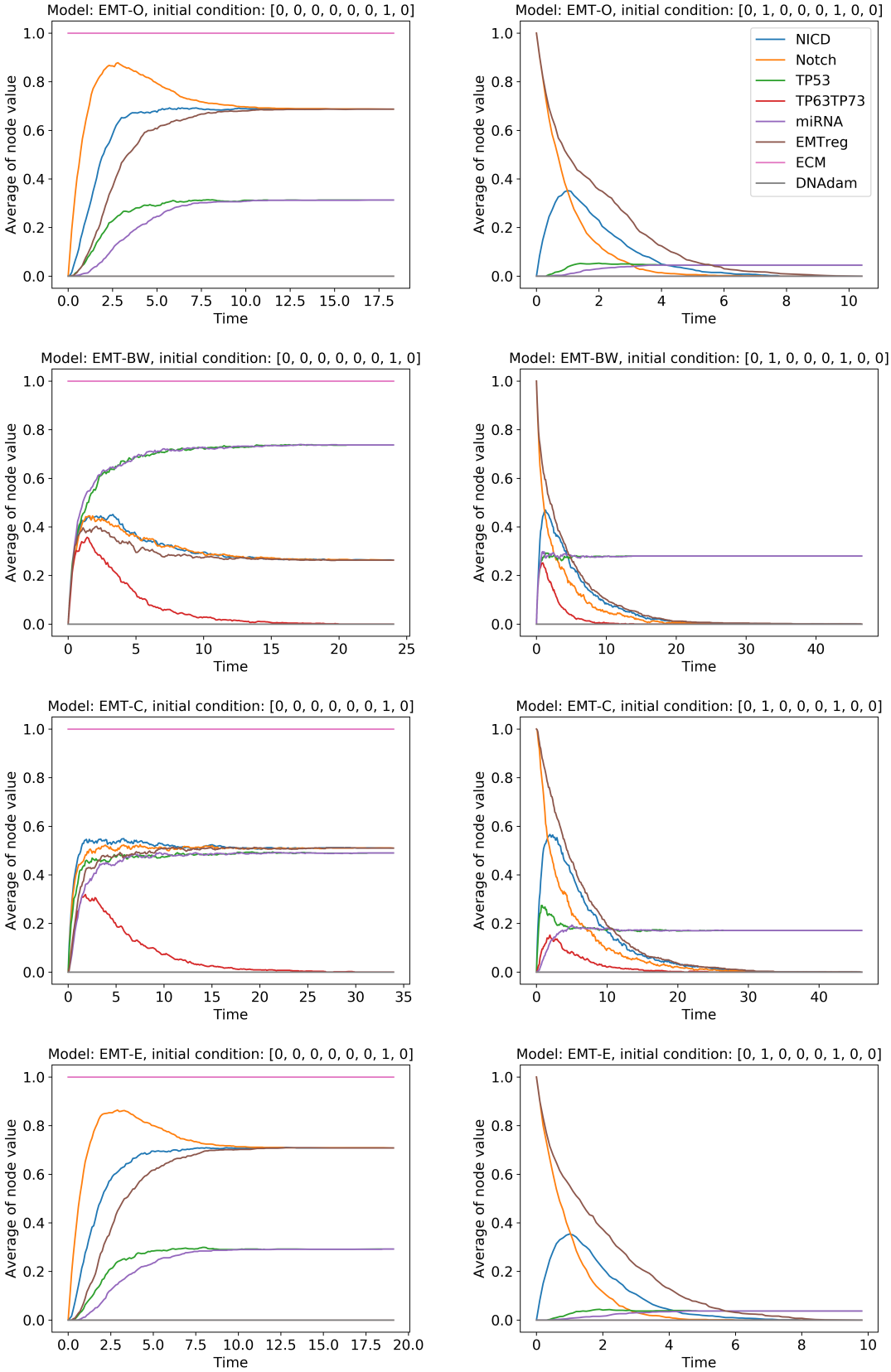

Fig 4. Comparison between EMT-O and some of our automatically created rulesets (EMT-BW, EMT-C, EMT-E). Initial state $(0,0,0,0,0,0,1,0)$ has the two attractors $(0,0,1,0,1,0,0,0)$ and $(1,1,0,0,0,1,1,0)$ (left column). The initial state $(0,1,0,0,0,1,0,0)$ converges towards the two attractors $(0,0,1,0,1,0,0,0)$ and $(0,0,0,0,0,0,0,0)$ (right column). On top, we depict the dynamics of the original model EMT-O. From row 2 to row 3 and 4, we include the more and more sophisticated automatically created network (EMT-BW, EMT-C, and EMT-E, respectively).

(ECM,DNAdam)	SS	EMT-O	EMT-FW	EMT-BW
(0,0)	(0,0,0,0,0,0,0,0)	0.61	0.51	0.52
	(0,0,1,0,1,0,0,0)	0.39	0.49	0.48
(0,1)	(0,0,1,0,1,0,0,1)	1.0	1.0	1.0
(1,0)	(0,0,1,0,1,0,1,0)	0.52	0.73	0.72
	(1,1,0,0,0,1,1,0)	0.48	0.27	0.28
(1,1)	(0,0,1,0,1,0,1,1)	0.79	0.78	0.77
	(1,1,0,0,0,1,1,1)	0.21	0.22	0.23

Table 6. Summary of the asynchronous updating results after 100 asynchronous updating simulations for each of the 256 possible initial states. The numbers on the right hand side are frequencies that the simulation reaches the corresponding steady state for the EMT model from [44], EMT-FW, and EMT-BW, respectively. The order of species is (NICD, Notch, TP53, TP63_TP73, miRNA, EMTreg, ECM, DNAdam).

If we start with the backward pathway model, we already cover all possible connections in the system. From Table 5, we know that we would need to remove about half of the connections from each network. This, however, is information taken from the underlying truth and can not be assumed as knowledge in a real experiment.
Our hypothesis is, that by removing a transition from the pathway towards an attractor, the initial state has to take a "detour" through the network to reach the attractor, thus making it less likely to reach this particular steady state, and more likely to move towards the other one instead. Due to the highly interconnected structures of our networks, however, we are aware that by removing the transition for one state to make it harder to reach an attractor, we might involuntarily also affect states that are supposed to reach the attractor more often. Therefore, a gradient based optimization will not perform well.
We chose a genetic optimization algorithm instead and extended our implementation by an option to randomly remove a number of transitions from the transition lists of each species. After gathering all the transitions created by EMT-BW and before we translate the transition lists into the ruleset, we randomly remove entries from those lists. The number of transitions to remove is the parameter that our optimizer chooses in order to improve the frequencies of the steady states.
This setup leads to a couple of difficulties.

1. By randomly removing transitions, we could violate point 3 of our problem list above and remove the pathway necessary to reach the attractor at all. Before we remove any transition from a species list, we therefore first check, if removing this transition is legal. Only if the reachability check answers TRUE, the transition will be removed from the list. We therefore ensure that the initial states will always be able to reach their attractors. These checks, however, extend the runtime for the creation of each model. Furthermore, we do not always remove the number of transitions proposed by the optimizer. If, for example, the optimizer proposes to remove 50 transitions and the algorithm can only find 40 valid transitions to remove, it will stop at the removal of 40 transitions, while still registered as a model with 50 transitions removed.

2. Due to the randomness of the removal, it is not enough to only create one model according to the suggested number of transition removals. In a genetic optimizer, every generation consists of multiple individuals that suggest their own number of transitions to remove. One of the individuals might have found the perfect number, however, the randomly selected transitions might be a bad choice and thus result in a bad fit. To avoid this, each individual of the algorithm does not only simulate one model, but multiple models with the same number of transitions removed.

A genetic algorithm consists of a number of individuals called a generation. After each individual computes its fitness, a selection process decides how to pick individuals to mate with each other and produce two new individuals according to the selected crossover. Some parts of the new individuals also get mutated according to the given mutation percentage.
For our optimization, our algorithm is based on the software package DEAP [46] using the build-in toolboxes for crossover, mutation, and selection.
We initialize 150 individuals as lists of random numbers between 0 and 1, denoting the number of transitions to be removed. Then, each individual uses its list of numbers to create 50 models that differ from each other by the randomness of the transition removals. Each of these models then runs 100 asynchronous updating simulations for each of the 256 initial states. It gathers the resulting frequencies of each state and compares it with the corresponding frequencies of the experimental data (i.e., the data taken from the model from [44]). The root mean square error

(RMS) is computed by taking the difference between each of the corresponding frequencies. The fitness for each individual is the smallest RMS from the 50 simulations.
The detailed parameter setup of DEAP for our simulation can be found in the supplementary Table S1. In the supplementary Figure S7, we see the development of the RMS over 90 generations. This simulation creates a total of $150 \times 50 \times 90 = 675000$ models.We identify the smallest RMS at the 86th generation with a value of 12.68. This will be from now on referred to as model EMT-C. A similar RMS of 12.76 can be found at generation 40 (EMT-B), which is less than half of the full simulation. As a third point of interest, we chose a relatively early model found at generation 7 with an RMS of 13.14 (EMT-A).

Model selection using expert knowledge

The above introduced model selection is the most general version with the least amount of knowledge input possible. In this section, we decrease the state space of models by adding expert knowledge to the interference process. Often the network structure (i.e. the dependence of a rule on other species) is known, or at least suspected. In our situation, the rule for NICD, e.g., displays a dependency to Notch, TP63_TP73, and TP53, but not to miRNA, EMTreg, ECM, or DNAdam. We now assume here that we know the dependencies for each node. In other words, the expression of our new rule set is no longer allowed to include the dependency of a species that it does not depend on in the original rule set. For this task, we take a closer look into how the ruleset is formulated from the transition states that have been determined by our model creator. Similar to the automated model selection from the previous section, we select our new network by legally discarding transitions. In this case, however, we do not randomly choose a set, but exploit the fact that $(x \wedge y \wedge z) \vee (\bar{x} \wedge y \wedge z) = y \wedge z$. I.e., if we want to eliminate a dependency of a variable, we need to eliminate each transition that does not include its symmetric counter part in the transition list. This method assures a non-dependency of the right hand side on the chosen species. Note, however, that our rules are formulated by $\text{species}^* = \left(\bigvee_{\text{transitions}} \text{transition state} \right) \veebar \text{species}$. Due to the XOR operation, we need to treat the self-dependency in the opposite matter. While for the elimination process of the other species we want to make sure that the symmetric counter parts are all present in the transition list, to ensure non self dependency we want to make sure that only one state of the two counter parts is present. In this step we need to be careful to ensure the symmetry from the step before. Let us, e.g,. assume that we want to create the first rule for species A for a list of five species (A, X, Y, Z, W), that only depends on the second and third species X and Y. Let us further assume, that the last two species, Z and W, have been eliminated form the formulation by only keeping the symmetric counterparts of each transition state in the list. In addition, both states $(0, X, Y, Z, W)$ and $(1, X, Y, Z, W)$ are in the transition list. We therefore need to eliminate one of those states. The algorithm needs to choose, whether to eliminate the transition, where the first species is in state 0, or in state 1. Both choices lead to a correct network. However, to guarantee the symmetry, for a fixed pattern of X and Y, the same choice has to be made. The selection process is detailed in Algorithm 1.

Due to the non-unique choice of the asymmetric elimination process taking place for the self-elimination, this algorithm leads once again to multiple possibilities of the network structure. In Table 7, we give an overview of the species to eliminate and the resulting number of possibilities for each rule.

From there we can see, that the rule for Notch and EMTreg are both unique. There are, however, still $> 8e6$ network possibilities for the rule for TP53 alone. This method still leads to a total of $16 \times 1 \times 8388608 \times 256 \times 32 \times 1 > 1e12$ possible networks to choose from. We therefore used again the DEAP algorithm to optimize for the frequencies of the steady states. For the list of the used DEAP parameters see the supplementary Table S1. In the supplementary Figure S8, we show the results for running the optimization over 25 generations using a population of 150 individuals. We already reach an RMS of 7.45 at generation 14, however, the smallest RMS achieved in this optimization is 7.44 at generation 20. This is the model that we use as expert guided model (EMT-E) in the following analysis. Note, that for this optimization, there was no additional random choice necessary. This simulation therefore produced $150 \times 25 = 3750$ models, where some of them are equivalent.

start with all possible transitions EMT-BW;
while *select species to make rule for* **do**
  while *select species that must not be part of the rule, except self reference* **do**
    if *symmetric counterpart not in transition list* **then**
      eliminate transition from list
    else
      keep both transitions in list
    end
  end
  if *both symmetric counterparts regarding the selected species are in the transition list* **then**
    choose a pattern for the non-eliminated species;
    decide the state of the species to remove (either 0 or 1);
    for every transition state of the same pattern, eliminate the transition where the species is in the chosen state;
  else
    keep the transition
  end
end

Algorithm 1: Model selection using biological insight.

rule	species to eliminate	possible number of resulting networks
NICD	NICD, miRNA, EMTreg, ECM, DNAdam	16
Notch	NICD, Notch, TP53, TP63_TP73, EMTreg, DNAdam	1
TP53	Notch, TP53, ECM	8388608
TP63_TP73	Notch, TP63_TP73, EMTreg, ECM	256
miRNA	NICD, Notch miRNA, ECM, DNAdam	32
EMTreg	Notch, TP53, TP63_TP73, EMTreg, ECM DNAdam	1

Table 7. Overview of model selection for the expert knowledge guided variant.

The resulting rule set for this choice is

$$
\begin{aligned}
\text{NICD}^* &= (\text{Notch} \wedge \overline{\text{TP53}} \wedge \overline{\text{TP63\_TP73}}) \\
&\quad \vee (\overline{\text{Notch}} \wedge \text{TP53} \wedge \text{TP63\_TP73}) \\
\text{Notch}^* &= \text{ECM} \wedge \overline{\text{miRNA}} \\
\text{TP53}^* &= (\text{DNAdam} \wedge \text{EMTreg} \wedge \overline{\text{miRNA}} \wedge \text{NICD} \wedge \text{TP63\_TP73}) \\
&\quad \vee (\text{DNAdam} \wedge \overline{\text{EMTreg}} \wedge \text{miRNA}) \\
&\quad \vee (\text{DNAdam} \wedge \overline{\text{EMTreg}} \wedge \overline{\text{NICD}}) \\
&\quad \vee (\text{DNAdam} \wedge \text{miRNA} \wedge \overline{\text{NICD}}) \\
&\quad \vee (\text{DNAdam} \wedge \overline{\text{NICD}} \wedge \overline{\text{TP63\_TP73}}) \\
&\quad \vee (\overline{\text{DNAdam}} \wedge \overline{\text{EMTreg}} \wedge \overline{\text{miRNA}} \wedge \text{NICD}) \\
&\quad \vee (\overline{\text{EMTreg}} \wedge \text{miRNA} \wedge \overline{\text{TP63\_TP73}}) \\
&\quad \vee (\overline{\text{EMTreg}} \wedge \overline{\text{miRNA}} \wedge \overline{\text{NICD}} \wedge \text{TP63\_TP73}) \\
&\quad \vee (\overline{\text{EMTreg}} \wedge \text{NICD} \wedge \overline{\text{TP63\_TP73}}) \\
\text{TP63\_TP73}^* &= \text{False} \\
\text{miRNA}^* &= \overline{\text{EMTreg}} \wedge \text{TP53} \\
\text{EMTreg}^* &= \text{NICD} \wedge \overline{\text{miRNA}} \\
\text{ECM}^* &= \text{ECM} \\
\text{DNAdam}^* &= \text{DNAdam}
\end{aligned}
$$

As we can see, the unique possibilities of the model selection for Notch and miRNA result in the correct rule expression. Setting TP63_TP73 to False is correct from a mathematical point of view (all attractors have this species at 0). This is a choice out of 256 possibilities and is valid given the system constraints imposed on the model selection. This rule could be made more biologically relevant by imposing additional expert knowledge, if desired. Due to the $> 8e6$ possible formulations for TP53, the convoluted formulation is not very surprising. The randomly

chosen models from the model optimizer above, as well as the original EMT-FW and EMT-BW models have even longer and more convoluted terms, which is why we do not represent the rulesets themselves in this paper. The rules for NICD and miRNA are very close to the ones from the original paper.

Analysis of the different models

In Figure 4, we can observe how the optimizers gradually improves the model. While EMT-BW produces some qualitatively wrong dynamics, EMT-C converges towards the correct dynamics. EMT-E captures the dynamics nearly perfectly. For a full comparison of the dynamics of all initial values, we refer the reader to the additional files `comp_SS1.pdf`, `comp_SS2.pdf`, `comp_SS3.pdf`, `comp_SS4.pdf`. To get an overall idea of how the dynamics of the states develop, we extend Table 6 by the frequencies resulting from the three models with EMT-A, EMT-B, and EMT-C, as well as the result for the expert knowledge guided optimization EMT-E, respectively in Table 8. As we have observed before, the steady state for the parameter set $(\text{ECM}, \text{DNAdam}) = (0, 0)$ for the EMT model is biased towards the first state $(\text{NICD}, \text{Notch}, \text{TP53}, \text{TP63\_TP73}, \text{miRNA}, \text{EMTreg}, \text{ECM}, \text{DNAdam}) = (0, 0, 0, 0, 0, 0, 0, 0)$ with approximately 60%. Both the EMT-FW and EMT-BW, as well as our first selected model EMT-A are very close in their behavior to 50%. With an RMS < 13 (EMT-B, EMT-C, EMT-E), the models, however, capture the bias towards the first steady state with 60% very similar to EMT-O.

(ECM,DNAdam)	SS	EMT-O	EMT-FW	EMT-BW	EMT-A	EMT-B	EMT-C	EMT-E
(0,0)	(0,0,0,0,0,0,0,0)	0.61	0.51	0.52	0.59	0.66	0.6	0.63
	(0,0,1,0,1,0,0,0)	0.39	0.49	0.48	0.42	0.34	0.4	0.37
(0,1)	(0,0,1,0,1,0,0,1)	1.0	1.0	1.0	1.0	1.0	1.0	1.0
(1,0)	(0,0,1,0,1,0,1,0)	0.52	0.73	0.72	0.5	0.5	0.5	0.5
	(1,1,0,0,0,1,1,0)	0.48	0.27	0.28	0.5	0.5	0.5	0.5
(1,1)	(0,0,1,0,1,0,1,1)	0.79	0.78	0.77	0.75	0.76	0.77	0.8
	(1,1,0,0,0,1,1,1)	0.21	0.22	0.23	0.25	0.24	0.23	0.2

Table 8. Summary of the asynchronous updating results after 90 asynchronous updating simulations for each of the 256 possible initial states. The numbers on the right hand side are frequencies that the simulation reaches the corresponding steady state for the EMT model from [44]. This is an extension to Table 6 with the inclusion of the three selected models from the genetic optimizer EMT-A, EMT-B, and EMT-C, respectively. Furthermore, we include the model found by the expert knowledge guided optimization EMT-E.

Contrary to the predictions of EMT-FW and EMT-BW, that bias the steady states of the parameter set $(\text{ECM}, \text{DNAdam}) = (1, 0)$ towards the steady state
$(\text{NICD}, \text{Notch}, \text{TP53}, \text{TP63\_TP73}, \text{miRNA}, \text{EMTreg}, \text{ECM}, \text{DNAdam}) = (0, 0, 1, 0, 1, 0, 1, 0)$ with 70%, all of our selected model keep the ratio of approximately 50% towards both steady states similar to the EMT results.
The bias observed for the parameter set $(\text{ECM}, \text{DNAdam}) = (1, 1)$ towards the steady state
$(\text{NICD}, \text{Notch}, \text{TP53}, \text{TP63\_TP73}, \text{miRNA}, \text{EMTreg}, \text{ECM}, \text{DNAdam}) = (0, 0, 1, 0, 1, 0, 1, 1)$ is observed by all the models.
Looking at the dynamics for parameter set $(\text{ECM}, \text{DNAdam}) = (1, 0)$, we immediately see the effect of the optimizer finding the trend of no bias towards a steady state compared with the non-optimized models. In Figure 5, we look into the distributions of the models according to Table 8. We only depict one of the steady states, since the other state would only be a complement to the corresponding figure.
The Kolmogorov-Smirnov test tries to evaluate, whether two data sets are drawn from the same distribution. A small p-value therefore hints towards two different distributions underlying the data sets. In Figure 5, we see for the subnetwork $(\text{ECM}, \text{DNAdam}) = (0, 0)$, that the distributions of EMT-FW and EMT-BW hint towards a different underlying distribution, than the data drawn from EMT-O. The generally selected models with EMT-A, EMT-B, and EMT-C have a relatively large p-value and therefore hint towards a similar distribution to the data for EMT-O. For the expert knowledge guided selected EMT-C model, the p-value is close to 1, and therefore we have obtained excellent agreement.
A similar behavior can be observed for the subnetwork $(\text{ECM}, \text{DNAdam}) = (1, 0)$. However, the p-values for EMT-FW and EMT-BW are even smaller, while for the optimized models, the p value is relatively large. This means that performing the optimizing is more important for this subnetwork.
For the subnetwork $(\text{ECM}, \text{DNAdam}) = (1, 1)$, we see that EMT-FW and EMT-BW give a relatively large p-value compared to the other two subnetworks - they can still be considered small, however, far from significant. These

models, therefore, might already give a decent approximation to the underlying data set. The optimization, in this case, actually gives a worse result for the models EMT-A and EMT-B. The effect of the optimization for EMT-C and EMT-E are not as strong as for the other subnetworks, both optimization results, however, can still be seen as an improvement to EMT-FW and EMT-BW.

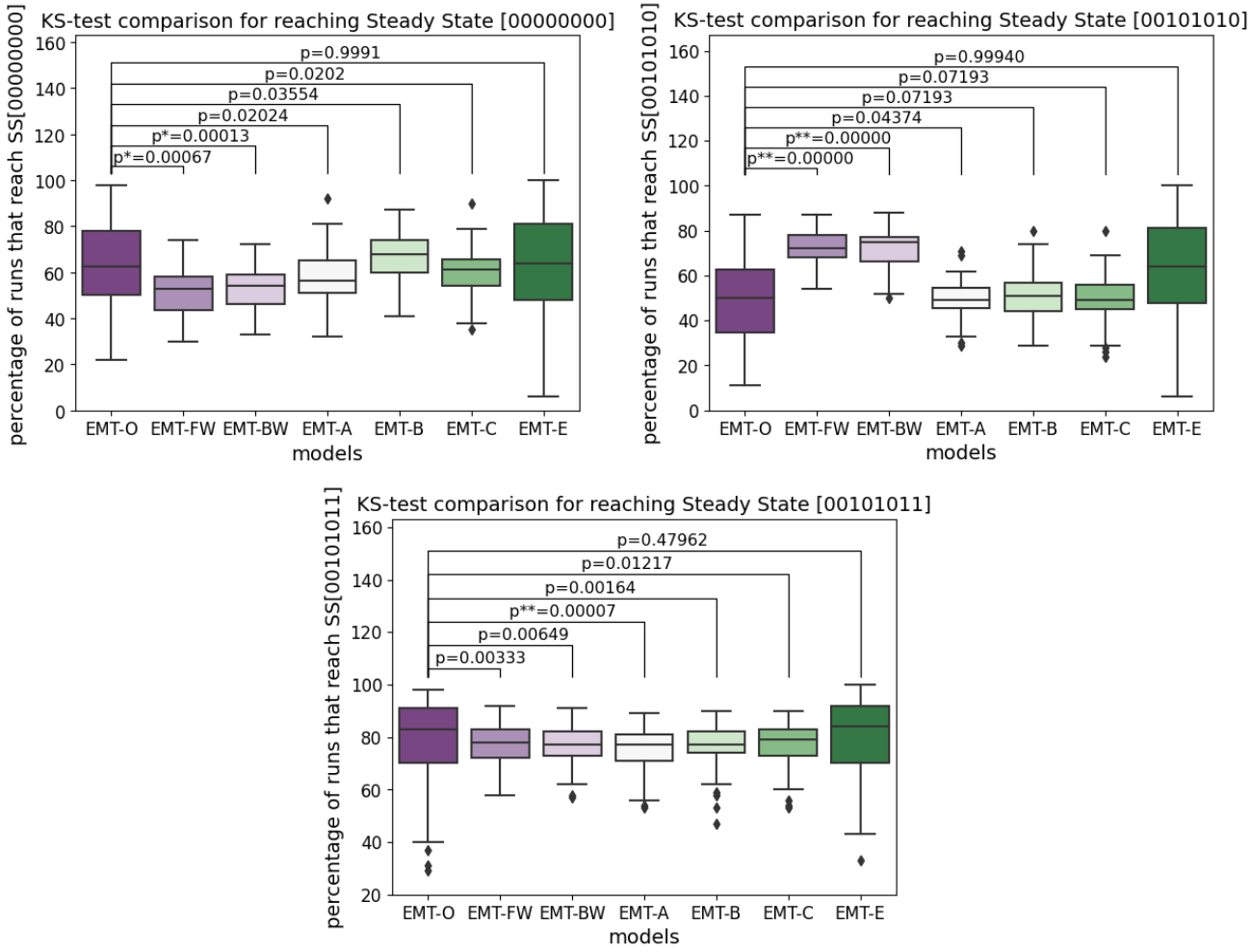

Fig 5. Frequency distribution of the states divided by the corresponding subnetworks. The second steady state for each of those subnetworks is omitted, since it is the complement of the depicted plots here. The noted p-value is taken from a Kolmogorov-Smirnov test between the distribution of frequencies between the original model EMT-O and EMT-FW, EMT-BW, EMT-A, EMT-B, EMT-C, and EMT-E, respectively. The left figure depicts the distribution of all states that can reach the attractor (00000000). The figure in the middle depicts the distribution of all states that can reach the attractor (00101010). The figure on the right depicts the distribution of all states that can reach the attractor (00101011).

The dynamics for all models discussed in this chapter for steady state 1, 2, 3, and 4 can be found in the files `comp_SS1.pdf`, `comp_SS2.pdf`, `comp_SS3.pdf`, `comp_SS4.pdf`, respectively. Note, that for steady state $(\mathrm{ECM}, \mathrm{DNAdam}) = (0, 1)$ (file `comp_SS2.pdf`), the initial states all converge towards a single attractor. Since the corresponding steady state is unique, EMT-BW, EMT-A, EMT-B, and EMT-C are equivalent (up to random fluctuations in the stochastic solver).

To summarize those findings, we look at Figure 6, where we explore in detail the differences between EMT-O and our automatically generated models. In the first category < 10, we include all the states that are within 10% of the EMT model. If a state, e.g., in EMT-O reaches a steady state 70%, and our selected model reaches the same state 65% of the time, we compute the distance using $70 - 65 = 5$. A distance of 5 is smaller than 10 and thus can be considered as a good state that represents a similar behavior than EMT-O.

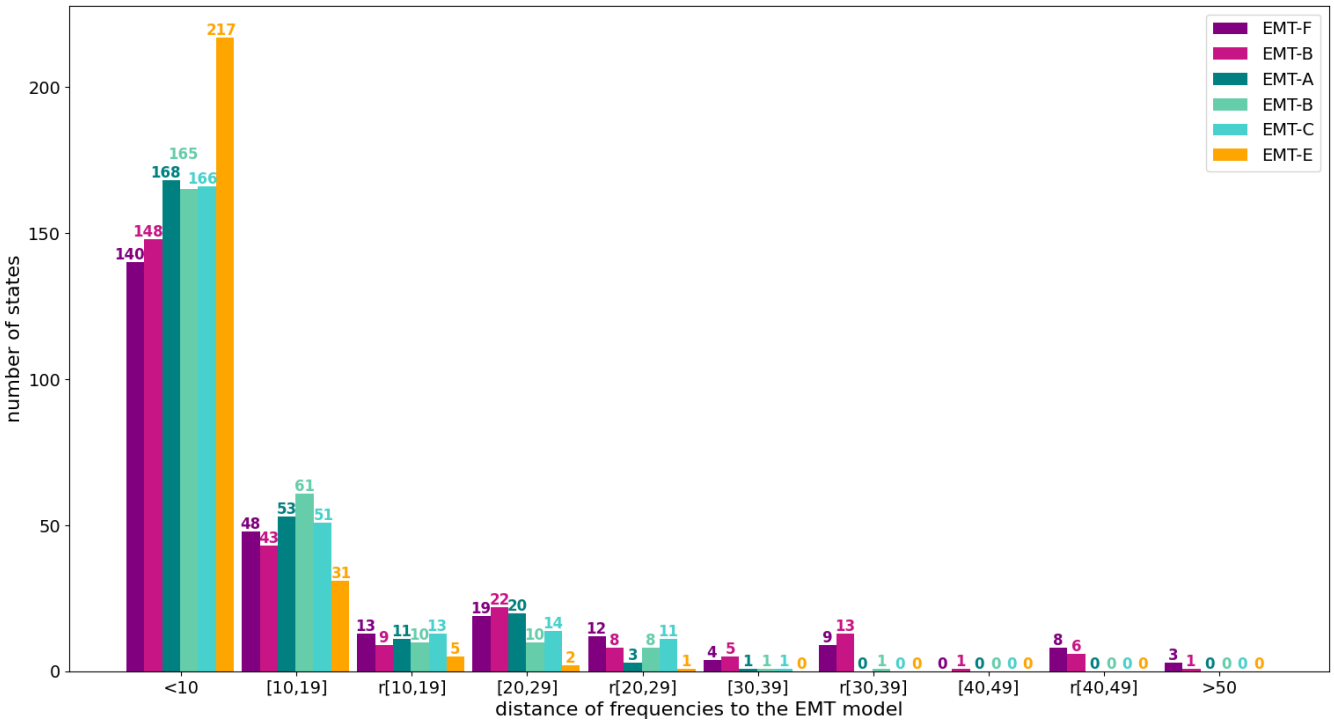

Fig 6. Accuracy of the steady state frequencies in relation to the EMT-O. The categories on the x-axis are in reference to the distance of the frequencies resulting from the EMT model over all states that converge to 2 steady states. In category < 10, all states are counted that are within a distance of 10% of the frequencies resulting from EMT-O. Category $[10, 19]$ counts the number of states that are within a distance of 20% from the results from EMT-O with the same qualitative behavior. In category r$[10, 19]$, we count the number of states that are within a distance of 20% from the results from EMT-O that also have a reversed qualitative behavior. Categories $[20, 29]$ and r$[20, 29]$ count the number of states that are within 30% of the EMT states with the same qualitative behavior, and the reversed qualitative behavior respectively, similar to categories $[30, 39]$, r$[30, 39]$, $[40, 49]$, and r$[40, 49]$. In category > 50, we count all the states that are more than 50% away from the EMT-O frequencies.

Category $[10, 19]$ counts all the states that are within 20% of the EMT results, and also experience the same qualitative behavior. If, e.g., a state in EMT-O reaches a steady state 70%, and the corresponding model state reaches this state 55%, the distance of these states is $70 - 55 = 15$. Both states have a bias towards this steady state (i.e., they are both larger than 50%), and therefore this state gets sorted into this category.
If the EMT-O state reaches a steady state 60%, and the corresponding model state reaches the same steady state 45%, the distance is still $60 - 45 = 15$, the qualitative behavior, however, is now reversed ($60 > 50$, but $45 < 50$), and this state gets sorted into category r$[10, 19]$. On first glance, the qualitative behavior of the states seems like a more important metric to keep on the states behavior. Note, however, that for a state to be within 20% of the EMT states and have a reversed behavior, both frequencies have to be relatively close to 50% and thus can still be considered approximately half.

The larger the distance, the more significant the impact of qualitatively wrong behavior gets as well. Therefore, the higher the bar on the left, i.e, the more states that are close to the EMT frequencies, the better the model. In this figure, we include now all the models discussed in this section. The purple colors depict the non-optimized models EMT-FW and EMT-BW. We can see, that both models experiences a couple of states, that are more than 50% away from the behavior of the EMT model. Neither of our optimized models has states in this category.
Our expert knowledge guided model has one state that is within 30% discrepancy that has the wrong qualitative behavior. 77% of states are within 10% of the original frequency values. This graph shows clearly, that most states are on the left side of the graph, i.e., close to the results of EMT-O, and that the optimization algorithm clearly skews the bars further to the left. The 217 orange states in the category <10 demonstrate the power of additional knowledge to guide the model selection process.

Discussion

We have proposed an algorithm to infer Boolean rules from the mapping of initial states to attractors. We have exemplified this method for two biologically relevant examples. Namely, a classic enzyme-substrate system and a model of Epithelial to Mesenchymal Transition (EMT) in cancer metastasis. In both cases the algorithm, without providing any additional information, provides candidate models that match the dynamics of the underyling system well. In particular, the steady states and their respective probability are reproduced accurately. For the enzyme-substrate system the dynamics is also well resolved. However, for the EMT model there are still an extremely large number of possible candidates and thus the dynamical behavior is not always faithfully resolved. This can be improved by incorporating additional insight into the systems. We have done this by constraining the network structure. That is, we have made assumptions on the species that each Boolean rule depends on. This results in a Boolean network which extremely accurately describes the dynamics of the underlying model. In fact, some of the inferred Boolean rules are identical to the ground truth.

The proposed algorithm could be used to infer the mechanisms of signaling, gene-regulatory, and any other input-output processes in an automatic (i.e. fully unsupervised) way. This enables us to use our methodology as part of a larger data processing, model inference, and prediction framework that can be used without human intervention. In this work we have exclusively considered data that only model the initial and final state of the system, because such experimental data are commonly available. However, with ever advancing measurement techniques more and more time series information tend to be available for such systems. We envisage the use of such time series to further improve the model selection. This will be subject of future research.

Our work takes advantage of parallel computing environments, thus reducing the amount of time required to enumerate logic rules by hand. We believe that computer-driven mechanism exploration coupled with a model selection, such as that presented in this work, could be a highly suitable tool to advance mechanism exploration and accelerate hypothesis prediction and testing *in silico*, for experimental validation, thus reducing the time and effort required to obtain mechanistic knowledge from experimental data.

Conclusion

We presented a general-purpose algorithm for mechanism exploration, hypothesis exploration, and model selection using initial and attractor state data and high-performance computing. Our approach greatly accelerates the inference of logic-based rules for complex biochemical networks and leads to dynamic networks that can be further explored in order to obtain testable hypotheses.

Materials and methods

The generic model selection consists of a genetic algorithm population of 150 individuals. Each of the individuals performs 50 simulations to account for the randomness of transition removal. We sequentially initialize the optimization with a Python interface and spawn a parallel environment using 50 nodes. On each node, one of the 50 individuals create and simulate the model according to the random process of transition removal. To speed up the simulations, each of the created rule sets was compiled into a C++ code to perform the asynchronous updating simulations. The full code can be accessed at `https://github.com/LoLab-VU/Boolean_rules_creator`.

Acknowledgements

We thank Prof. Vito Quaranta and the Quaranta Lab for their useful insights throughout the development of this work. This work was supported by National Science Foundation (NSF) award MCB 1942255 to CFL, National Institutes of Health (NIH) award 1U01CA215845 to CFL (M-PI).

Author Contributions

Conceptualization, MP, LE, SPB, LAH, and CFL; Software, MP, LE; Writing—Original Draft, MP, LE, CFL; Writing—Review Editing, MP, LE, SPB, LAH, and CFL; Supervision, CFL; Funding Acquisition, CFL.

Conflicts of Interest

The authors declare no competing interests.

References

1. Huang S. Non-genetic heterogeneity of cells in development: more than just noise. Development. 2009;136(23):3853–3862.

2. Janes KA, Lauffenburger DA. Models of signalling networks–what cell biologists can gain from them and give to them. Journal of cell science. 2013;126(9):1913–1921.

3. Karr JR, Sanghvi JC, Macklin DN, Gutschow MV, Jacobs JM, Bolival Jr B, et al. A whole-cell computational model predicts phenotype from genotype. Cell. 2012;150(2):389–401.

4. Kim E, Kim JY, Smith MA, Haura EB, Anderson AR. Cell signaling heterogeneity is modulated by both cell-intrinsic and-extrinsic mechanisms: An integrated approach to understanding targeted therapy. PLoS biology. 2018;16(3):e2002930.

5. Meyer M, Paquet A, Arguel MJ, Peyre L, Gomes-Pereira LC, Lebrigand K, et al. Profiling the Non-genetic Origins of Cancer Drug Resistance with a Single-Cell Functional Genomics Approach Using Predictive Cell Dynamics. Cell Systems. 2020;11(4):367–374.

6. Wang S, Lin JR, Sontag ED, Sorger PK. Inferring reaction network structure from single-cell, multiplex data, using toric systems theory. PLoS computational biology. 2019;15(12):e1007311.

7. Jajoo R, Jung Y, Huh D, Viana MP, Rafelski SM, Springer M, et al. Accurate concentration control of mitochondria and nucleoids. Science. 2016;351(6269):169–172.

8. Norris JL, Farrow MA, Gutierrez DB, Palmer LD, Muszynski N, Sherrod SD, et al. Integrated, high-throughput, multiomics platform enables data-driven construction of cellular responses and reveals global drug mechanisms of action. Journal of proteome research. 2017;16(3):1364–1375.

9. Wiśniewski JR, Hein MY, Cox J, Mann M. A "proteomic ruler" for protein copy number and concentration estimation without spike-in standards. Molecular & cellular proteomics. 2014;13(12):3497–3506.

10. Azizi E, Carr AJ, Plitas G, Cornish AE, Konopacki C, Prabhakaran S, et al. Single-cell map of diverse immune phenotypes in the breast tumor microenvironment. Cell. 2018;174(5):1293–1308.

11. Ireland AS, Micinski AM, Kastner DW, Guo B, Wait SJ, Spainhower KB, et al. MYC Drives Temporal Evolution of Small Cell Lung Cancer Subtypes by Reprogramming Neuroendocrine Fate. Cancer Cell. 2020;.

12. Jung S, Del Sol A. Multiomics data integration unveils core transcriptional regulatory networks governing cell-type identity. NPJ systems biology and applications. 2020;6(1):1–4.

13. Sacco F, Silvestri A, Posca D, Pirrò S, Gherardini PF, Castagnoli L, et al. Deep proteomics of breast cancer cells reveals that metformin rewires signaling networks away from a pro-growth state. Cell Systems. 2016;2(3):159–171.

14. Shockley EM, Rouzer CA, Marnett LJ, Deeds EJ, Lopez CF. Signal integration and information transfer in an allosterically regulated network. NPJ systems biology and applications. 2019;5(1):1–9.

15. Wooten DJ, Groves SM, Tyson DR, Liu Q, Lim JS, Albert R, et al. Systems-level network modeling of Small Cell Lung Cancer subtypes identifies master regulators and destabilizers. PLoS computational biology. 2019;15(10):e1007343.

16. Kauffman S. Homeostasis and differentiation in random genetic control networks. Nature. 1969;224(5215):177–178.

17. Clarke MA, Fisher J. Executable cancer models: successes and challenges. Nature Reviews Cancer. 2020; p. 1–12.

18. Niarakis A, Helikar T. A practical guide to mechanistic systems modeling in biology using a logic-based approach. Briefings in Bioinformatics. 2020;.

19. Saadatpour A, Albert I, Albert R. Attractor analysis of asynchronous Boolean models of signal transduction networks. Journal of theoretical biology. 2010;266(4):641–656.

20. Kholodenko B, Yaffe MB, Kolch W. Computational approaches for analyzing information flow in biological networks. Science signaling. 2012;5(220):re1–re1.

21. Mogilner A, Allard J, Wollman R. Cell polarity: quantitative modeling as a tool in cell biology. Science. 2012;336(6078):175–179.

22. Paulevé L, Kolčák J, Chatain T, Haar S. Reconciling qualitative, abstract, and scalable modeling of biological networks. Nature Communications. 2020;11(4256).

23. Saez-Rodriguez J, Alexopoulos LG, Zhang M, Morris MK, Lauffenburger DA, Sorger PK. Comparing signaling networks between normal and transformed hepatocytes using discrete logical models. Cancer research. 2011;71(16):5400–5411.

24. Wang RS, Saadatpour A, Albert R. Boolean modeling in systems biology: an overview of methodology and applications. Physical biology. 2012;9(5):055001.

25. Yachie-Kinoshita A, Onishi K, Ostblom J, Langley MA, Posfai E, Rossant J, et al. Modeling signaling-dependent pluripotency with Boolean logic to predict cell fate transitions. Molecular systems biology. 2018;14(1):e7952.

26. Mitra ED, Dias R, Posner RG, Hlavacek WS. Using both qualitative and quantitative data in parameter identification for systems biology models. Nature communications. 2018;9(1):1–8.

27. Shockley EM, Vrugt JA, Lopez CF. PyDREAM: high-dimensional parameter inference for biological models in python. Bioinformatics. 2018;34(4):695–697.

28. Kanehisa M. Toward understanding the origin and evolution of cellular organisms. Protein Science. 2019;28(11):1947–1951.

29. Wishart DS, Tzur D, Knox C, Eisner R, Guo AC, Young N, et al. HMDB: the human metabolome database. Nucleic acids research. 2007;35(suppl_1):D521–D526.

30. Wu G, Haw R. Functional interaction network construction and analysis for disease discovery. In: Protein Bioinformatics. Springer; 2017. p. 235–253.

31. Akutsu T, Miyano S, Kuhara S. Inferring qualitative relations in genetic networks and metabolic pathways. Bioinformatics. 2000;16(8):727–734.

32. Hamey FK, Nestorowa S, Kinston SJ, Kent DG, Wilson NK, Göttgens B. Reconstructing blood stem cell regulatory network models from single-cell molecular profiles. Proceedings of the National Academy of Sciences. 2017;114(23):5822–5829.

33. Razzaq M, Paulevé L, Siegel A, Saez-Rodriguez J, Bourdon J, Guziolowski C. Computational discovery of dynamic cell line specific Boolean networks from multiplex time-course data. PLoS computational biology. 2018;14(10):e1006538.

34. Fisher J, Köksal AS, Piterman N, Woodhouse S. Synthesising executable gene regulatory networks from single-cell gene expression data. In: International Conference on Computer Aided Verification. Springer; 2015. p. 544–560.

35. Barman S, Kwon YK. A Boolean network inference from time-series gene expression data using a genetic algorithm. Bioinformatics. 2018;34(17):i927–i933.

36. Gao S, Xiang C, Sun C, Qin K, Lee TH. Efficient Boolean Modeling of Gene Regulatory Networks via Random Forest Based Feature Selection and Best-Fit Extension. In: 2018 IEEE 14th International Conference on Control and Automation (ICCA). IEEE; 2018. p. 1076–1081.

37. Martin S, Zhang Z, Martino A, Faulon JL. Boolean dynamics of genetic regulatory networks inferred from microarray time series data. Bioinformatics. 2007;23(7):866–874.

38. Shi N, Zhu Z, Tang K, Parker D, He S. ATEN: And/Or tree ensemble for inferring accurate Boolean network topology and dynamics. Bioinformatics. 2020;36(2):578–585.

39. Tyson JJ, Chen K, Novak B. Network dynamics and cell physiology. Nature reviews Molecular cell biology. 2001;2(12):908–916.

40. Tyson JJ, Chen KC, Novak B. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. Current opinion in cell biology. 2003;15(2):221–231.

41. Wynn ML, Consul N, Merajver SD, Schnell S. Logic-based models in systems biology: a predictive and parameter-free network analysis method. Integrative biology. 2012;4(11):1323–1337.

42. Zanudo JG, Albert R. Cell fate reprogramming by control of intracellular network dynamics. PLoS Comput Biol. 2015;11(4):e1004193.

43. Hamming RW. Error detecting and error correcting codes. The Bell system technical journal. 1950;29(2):147–160.

44. Calzone L, Barillot E, Zinovyev A. Logical versus kinetic modeling of biological networks: applications in cancer research. Current Opinion in Chemical Engineering. 2018;21:22–31.

45. Chanrion M, Kuperstein I, Barrière C, El Marjou F, Cohen D, Vignjevic D, et al. Concomitant Notch activation and p53 deletion trigger epithelial-to-mesenchymal transition and metastasis in mouse gut. Nature communications. 2014;5(1):1–15.

46. Fortin FA, De Rainville FM, Gardner MA, Parizeau M, Gagné C. DEAP: Evolutionary Algorithms Made Easy. Journal of Machine Learning Research. 2012;13:2171–2175.