AN AUTOMATED METHOD TO OVERCOME THE DIFFICULTY OF BUILDING COMPUTATIONAL MODELS

By

Bryan J. Glazer

Thesis
Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

Master of Science
in

Biomedical Informatics

Graduation Month Day, Year

Nashville, Tennessee

Approved:

Jake Hughey, Ph.D.

Ivelin Georgiev, Ph.D.

Carlos Lopez, Ph.D.

Copyright © 2022 Bryan J. Glazer
All Rights Reserved

This thesis is dedicated to my wife Sara and my son Levi. I love you!

ACKNOWLEDGMENTS

This work was supported by the National Library of Medicine Training Grant T15 LM007450

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

The Necessity of Computational Models for Studying Biological Systems

Interaction between biomolecules is the primary mode through which information and energy flows in life. The fine details of even the most fundamental interactions have significant complexity (Westheimer, 1987; Kamerlin et al., 2013), which is compounded as sequences of interactions connect to form networks. At this higher scale, novel phenomena emerge that are products of the structure and dynamics of networked interactions. These effects, several of which I will discuss in Section 1.1, can be counter-intuitive and difficult to predict from a static diagrammatic or textual description of a system. However, these complex behaviors are often critically important in the response of biological systems to perturbations such as drugs or novel conditions. Consequently, I describe how mathematical modeling can account for these complex, emergent phenomena.

Despite their utility in understanding biological systems, accurate computational models can be difficult to construct. Using a simple framework, Boolean logic models, I show in Chapter 2 that even the most skilled model builders face an overwhelming task in finding the interactions that will make their model behave in the ways they want. Using empirical analysis of a collection of eleven million models, I demonstrate that Boolean models, despite their simplicity, have complex, unpredictable properties that make building them exceptionally difficult.

However, this is not a totally hopeless situation, as I will show in Chapter 3. By exploiting an efficient, modern search technique borrowed from research on game playing algorithms, I show that we can automatically generate diverse sets of Boolean models with a wide variety of desired behaviors.

In brief, I hope to make the following argument. Computational models are crucial for understand biology, but they are difficult to build. Therefore, we need automated methods to build them, as I show with the method that I developed.

1.1 Biological systems exhibit multiple complex and unintuitive behaviors

In this section, I will describe the importance of computational biological models in the context of some difficult, emergent properties of biology. When applied to systems with noise or cross-talk, computational models can reveal mechanisms and behaviors that are hidden from view when analyzed through static, non-computational representations.

1.1.1 Cross-talk and feedback

The traditional view of biochemical pathways portrays distinct, sequential chains of biochemical reactions. Pathways begin with an input such as a signal or precursor, proceed through intermediary steps, and end with production of a metabolic product or ”effector” molecule (Ross, 2021). This model of signal transduction makes two important assumptions. First, pathways are distinct from one another, each independently communicating a signal without affecting the function of other pathways. Second, pathways are strictly linear, with well defined steps that follow each other sequentially. These two assumptions are appealing, as they allow simple intuitions to guide understanding of pathway dynamics. One can imagine pathways acting deterministically and independently, similar to human-designed mechanical systems. However, as I will show below, this model of cellular signal transduction is inaccurate and inadequate for understanding many of the observed behaviors of cells.

First, I will examine interactions between pathways, a phenomena termed cross-talk, showing that it is prevalent across cellular systems and that it has important effects on the responses of signaling pathways. Efforts to catalog and classify instances of cross talk include XTalkDB (Sam et al., 2017), a manually curated database of pathways and interactions between them. Investigation of XTalkDB reveals that cross talk is indeed very common, with 345 pairs of pathways having literature supported evidence of crosstalk. Further, 14 of XTalkDB’s pathways are highly prolific, regulating at least half of the other pathways in the database. Several of the pathways, such as Hedgehog, Wnt, and Notch signaling are both prolific cross-talkers and highly evolutionarily conserved, indicating that cross talk is a key component of signaling in development and homeostasis. These interactions include 24 tissues across 17 species.

Notably, the authors of XTalkDB used a very conservative criteria for inclusion of a cross-pathway interaction in the database, selecting only well characterized interactions with manually verified literature references. Given the evidence that literature sources are biased towards the study of a few well known genes and pathways (Haynes et al., 2018), this strict curation raises the possibility that many pathways may have many more cross talk interactions than those in XTalkDB. Further study and use of high throughput methods may reveal that many pathways have a high degree of interaction with other parts of cellular regulatory systems, perhaps to the degree that individual pathways are no longer distinguishable.

Thus, the curated interactions of XTalkDB could be considered a lower bound on the extent of cross-talk across signaling pathways. What might be the upper bound on cross-talk? What is the sum of all evidence for interactions across pathways? Here, I will describe analysis that I performed to determine an hypothetical upper bound on the degree of cross-talk across the human proteome, using a database of protein-protein interactions.

The STRING database provides a list of protein-protein interactions, integrated from a variety of high throughput sources such as co-expression, literature mining, and direct experimental evidence (Szklarczyk et al., 2021). These high throughput methods promise to generate more comprehensive and unbiased lists of protein-protein interactions, at the cost of less detailed knowledge of the nature of the interactions and consequently higher false positive rates.

I analyzed a version of the STRING database that contains only experimental evidence of physical interaction, derived from the iMEX consortium and BioGRID database (Oughtred et al., 2021; Orchard et al., 2012). This database contains 18318 proteins and 2.4 million interactions between them. This forms a network comprising a single connected component, meaning that every protein has a path to interact indirectly with every other protein in the network. I analyzed the distance of paths between proteins by calculating the eccentricity. The eccentricity metric computes a shortest path between all pairs of nodes in the network, and the eccentricity of a node is the length of the longest of all the shortest paths to other nodes. The STRING physical interactions network had a median eccentricity of 5 and a maximum eccentricity of 7. Intuitively, this means that every protein is separated by at most 7 interactions, with most proteins separated by fewer than 5 steps.

This indicates extremely prevalent cross-talk between all pathways in human cells. Admittedly, many of the interactions in STRING are likely due to false-positive experimental results or are specific to certain tissues or contexts. Nonetheless, the actual level of interaction between pathways probably lies somewhere between XTalkDB and STRING. Clearly, biological evidence does not support the traditional notion of pathways as distinct, linear chains. The significance of this observation is twofold. First, understanding of a signaling pathway must incorporate at least the neighboring pathways, i.e. those that can directly influence its behavior. This breaks the conventional notion of distinct pathways and often leads to systems that have dozens, if not hundreds, of interactions. Second, feedback loops, both within and between pathways, lead to complex, non-linear behavior that can be difficult to predict from examining the static structure of a pathway.

As an example of the complexity introduced by cross-talk, I will examine a well-known pathway, specifically RAS mediated MAPK signaling, which transmits extracellular signals to nuclear transcription factors. MAPK signaling regulates the activity of a multitude of cellular processes, from cellular proliferation in cancer (Waters and Der, 2018) to chemotaxis in amoebas (Bracco and Pergolizzi, 2014). The early canonical view of RAS signaling depicted a linear chain from extracellular receptors such as EGFR to RAS, followed by sequential activation of RAF, MEK, and then ERK, which is considered the ”effector” protein of the signaling pathway. More recent studies have shown that this view of the pathway is incomplete. For example, in their review Mendoza et al. (2011) detail multiple mechanisms by which the PI3K pathway interacts with RAS signaling. Of particular therapeutic interest is cross-inhibition, in which activity of one pathway

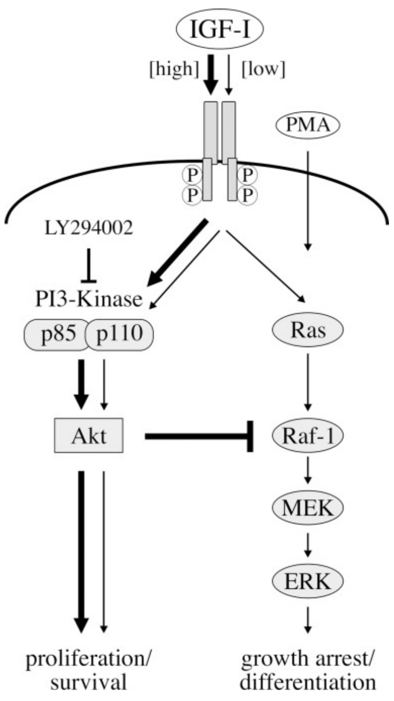


Figure 1.1: IGF signaling and cross-talk between the AKT and MAPK pathways, as depicted by Moelling et al. (2002). Used under Creative Commons Attribution (CC BY 4.0)

reduces the activity of the other. As shown in Figure 1.1, Moelling et al. (2002) shows that activation of the PI3K pathway reduces activity of the RAS signaling pathway by inhibiting RAF activity. Conversely, they also show that activation of RAS signaling does not inhibit PI3K signaling, indicating that cross-talk is not symmetric between the pathways. While the authors do provide a schematic model of the proposed cross-inhibition interactions (shown in Figure 1.1), this simple representation does not allow them to further investigate temporal responses of the pathways or the effect of perturbations on cross-inhibition from PI3K pathway. Further, the author's diagrammatic representation of cross-pathway activity does not allow quantitative comparisons to alternative models. For example, the authors cannot answer questions such as how negative feedback from ERK to RAF (proposed by Dougherty et al. (2005)) would change the dynamics of this system.

While the experimental characterization of Moelling et al. provided useful qualitative descriptions of cross-talk, work by D'Alessandro et al. illustrates the unique insights that can be gained when experimental evidence is combined with computational models (D'Alessandro et al., 2015). In this work the authors investigated mechanisms of cross-talk between the PI3K and MAPK signaling pathways. They measured temporal changes in phosphorylation of several proteins after activation by HGF under normal conditions and with inhibitors of both pathways. HGF is a signaling protein that activates both pathways. They observed that phosphorylated MEK initially decreased in concentration under a MEK inhibitor but then rebounded to a high level, indicating cross pathway compensation or perhaps a release of feedback inhibition.

Neither of these phenomena were captured in or could easily be tested by examining the canonical pathway diagrams. Therefore, the authors applied a computational approach to understanding cross-talk between the pathways. The authors automatically generated models of the pathway cross-talk configurations that were consistent with the observed data. The authors then simulated the effect of single and double inhibition of pathway components. Strikingly, they observed that simulated single inhibition of the PI3K pathway led to upregulation of MAPK signaling, consistent with their experimental observations of MEK. They also observed that inhibiting certain members of both pathways simultaneously was synergistic, meaning that double inhibition had an effect that was greater than the summed effects of single pathway inhibition.

This work benefits from several advantages of computational modeling over qualitative, schematic models of signaling. First, the authors automatically generated sixteen variations of their model, each proposing unique mechanisms of pathway interaction. While generating this number of models might be accessible for an individual investigator, it would certainly be difficult and labor-intensive. Second, the authors use simulations to predict the temporal dynamics of each of the sixteen models under a variety of conditions. This produced several unintuitive and unexpected results described above, including synergy and cross-inhibition. The authors did not hypothesize that these phenomena would occur based solely on the graphical structure of their models, yet follow up experiments validated the models predictions.

1.1.2 Noise

As demonstrated above, the cellular signaling architecture is immensely structurally complex. Signaling pathways are highly interconnected and have feedback and cross-talk interactions that lead to complex phenomena that are difficult to predict or understand. Despite this complexity, computational modeling allows researchers to manipulate and understand signaling networks in ways that are not possible with simpler tools.

However, this structural complexity is further compounded by the inherent noisiness of the cellular environment. This stochasticity arises from several sources. First, molecules are constantly buffeted by collisions in the crowded cytoplasm, exhibiting complex, non-Brownian stochastic movement (Di Rienzo et al., 2014). Further, many biologically important molecules exist at extremely low (nanomolar) concentrations, with individual cells containing as few as ten copies of a molecule (McAdams and Arkin, 1999). At such low concentrations, random fluctuations dictate that there will necessarily be high variability between cells.

In one of the first quantitative studies of noise in gene expression, Elowitz et al. (2002) inserted two fluorescent proteins with identical promoters into an E. coli genome. They then quantified differences between expression of the two fluorescent reporter proteins. This led them to define two forms of noise: extrinsic and intrinsic. In individual cells, extrinsic noise is characterized by correlated fluctuations in the levels of the two markers, with extrinsic factors affecting expression of both in a similar manner. Intrinsic noise is not correlated between markers in individual cells, with internal fluctuations affecting each reporter individually.

In their study Elowitz et al. observed substantial intrinsic and extrinsic noise. However, life depends upon extremely precise regulation of countless processes. This raises several questions. What is the source of intrinsic noise in cells and how is it regulated? How does precise organization emerge from noisy processes? Below, I will show how computational modeling has been crucial in answering these questions.

Raser and O'Shea (2004) made significant early progress in understanding the source and control of noise in gene expression. Building on prior biological knowledge they proposed three mechanisms for how genes are activated to allow transcription: chromatin remodeling, nucleosome sliding, and transcription factor binding. To test these hypotheses they used a similar dual reporter assay to Elowitz, but they also integrated their data into a computational model. For each of the three potential mechanisms they proposed different parameters for their model, corresponding to the dynamics of each biological process. They found that their model best supported nucleosome sliding as the most important mechanism for gene activation. This illustrates the power of even a simple computational model for providing insight into unintuitive biochemical processes. Importantly, their fluorescence data was not high resolution enough to directly visualize or distinguish the individual nucleosomes or sections of chromatin. However, when interpreted through the lens of a mathematical model, their data provided evidence for a process that could not be directly observed.

The work by Raser et al. and subsequent studies made significant progress elucidating transcriptional noise in single cells. However, this left the question of how this noise is controlled and integrated across whole tissues to generate the precise body plans that are ubiquitous in life. In more recent work, Zoller et al. (2018) made advances in this question by combining deep investigation of a computational model with advances in high resolution fluorescent imaging. They investigated the expression of key developmental ”gap” genes in Drosophila embryos. These genes have non-overlapping expression with sharp borders in between segments. These borders are critical for Drosophila development, determining the basic patterning of the mature fly's body plan. Thus, the authors raised the question of how such precise patterns of gap gene expression form, despite the noise observed in individual cells. By integrating single cell measurements of fluorescently labeled transcripts with a mathematical model of transcription, they found that most parameters of the model are constant across cells. Instead, the level of transcription was uniquely determined by the rates at which a promoter switches from an inactive to active state. Their mathematical modeling was able to exclude the possibility that cells modulate the rate of RNA polymerase binding and transcription initiation, instead relying only on the rate at which promoters are activated, similar to observations from Raser and O'Shea (2004).

This suggests that signals between cells somehow modify the key parameters of transcription, generating the differences in gene expression that maintain the precise patterns of key developmental genes. However, a reasonable assumption would be that extra-cellular signals are just as noisy as the intra-cellular processes observed above. Again, this raises the question of how tissues maintain precise patterning despite noisy signals. Suderman et al. (2017) used computational modeling to provide an unintuitive answer to this question through their investigation of noisy information transfer in an apoptosis pathway. They measured how different concentrations of an apoptosis inducing compound (TRAIL) affect an apoptosis marker (cPARP). They used computational methods to estimate the channel capacity, which is a theoretical upper bound on the amount of information that a signaling pathway can transmit. They found that in individual cells the channel is very low capacity, converting less than one bit of information about TRAIL concentrations to the activity of cPARP. However, this raised the question of how organisms maintain their integrity when this important apoptosis signaling pathway is so noisy. To answer this question, they developed a computational model of noisy information transfer. Simulating this model showed that while individual cells responded noisily to TRAIL signals, the *overall fraction* of cells that died was finely tuned to the concentration of TRAIL. Thus, the channel capacity of TRAIL signaling at a tissue level was quite high. Further, this high channel capacity at the population level actually required noisiness at the individual level. This counter-intuitive relationship was only evident through computational modeling and was borne out by experiments on multiple cell lines.

1.1.3 Conclusion

In multiple cases in the research that I reviewed, initial descriptions of a biological system revealed significant complexity. In the case of growth factor signaling in Section 1.1.1, early researchers found that supposedly distinct, linear pathways were in fact more complex, with cross-interactions and feedback that resulted in unexpected responses to perturbations. Adept use of computational modeling showed the power of in-silico experiments to probe mechanisms, rather than relying on intuition from static diagrams. Further, by automatically generating multiple of variations of the model, researchers could quantitatively compare a large number of hypotheses, which is practically impossible using non-computational representations of a system. Note that automatically generating models, demonstrated on a limited scale above, is the focus of Chapter 3 of this thesis.

Efforts to catalogue cross-talk show that interactions between pathways are ubiquitous, meaning that these tools are broadly applicable in studying multiple types of signaling. If one believes the most aggressive estimates, the concept of an individual pathway may be useless. Instead cellular signals are inextricably tied by millions of links into a single connected network.

Further complicating matters is the observation of significant variability in gene expression between cells. Even within a single cell, gene expression sometimes fluctuates in dramatic bursts. Yet, organisms maintain exquisitely precise regulation of their development and physiology. Again, computational models help make sense of this paradox. In fact, in each of the works presented in Section 1.1.2, I have highlighted how the authors gain insights into biological processes that are invisible without the power of computational modeling.

Having demonstrated the utility of models, I will next review frameworks for constructing computational biological models, with a focus on a simple but powerful paradigm: Boolean modeling.

1.2 Benefits and disadvantages different modeling frameworks

As I have shown above, computational modeling is a powerful tool for understanding biological systems. However, there are several different common frameworks that one can use to build models. Each allows the creation of models using a different set of mathematical tools and assumptions. Here, I will provide a brief review of the concepts and assumptions that underlie several of the most popular modeling frameworks and describe the benefits and disadvantages of each approach. I will focus on Boolean modeling, as it is the paradigm that I will pursue further in Chapter 2 and 3.

1.2.1 Boolean Models

In historical terms, the Boolean network formalism is relatively new, introduced to biology by Stuart Kauffman's 1969 explorations of homeostasis in randomly constructed networks (Kauffman, 1969a,b). More recent advances in experimental techniques allowed higher throughput biochemical measurements, spurring both the need and ability to construct Boolean models of larger systems. A notable early example is Albert and Othmer (2003), which developed a multi-cellular Boolean model that recapitulated intercellular signaling and pattern formation in Drosophila embryos. Albert and Othmer's mathematical analysis of the network also revealed that it was remarkably robust to perturbations, highlighting the utility of Boolean models for understanding higher-level, emergent properties of biochemical networks. Here, I will provide a brief overview of the fundamentals of Boolean networks. In later sections, I will compare their benefits and disadvantages compared to other modeling frameworks.

Perhaps the simplest modeling framework in common use, Boolean models represent the state of biomolecules as either one or zero (i.e. active or inactive). Interactions between nodes in Boolean models are encoded as logic statements, i.e. formulas with the AND, OR, and NOT operators acting on the binary states of the species in the model.

More specifically, the state of each node in the network at the next simulation step $(t+1)$ is determined by a logic formula that depends on the states of the other nodes at the current time (t):

$$x_n^{t+1} = f(x_1^t, x_2^t, ...)$$

Here, f could be any Boolean logic formula combining the AND, OR, NOT operators. However, attributing biological significance to an arbitrary Boolean formula can be difficult. Instead, a common strategy is to restrict the update formulas to specific forms that admit easier interpretations. For example, when modeling transcription of genes, a common form for the update formulas is dominant inhibition, in which interactions can be activating or inhibiting. ”Dominant” inhibition signifies that inhibitory interactions negate the effect of activating interactions. Mathematically this takes the form:

$$x_n^{t+1} = (x_1^t \text{ OR } x_2^t) \text{ AND NOT } (x_3^t \text{ OR } x_4^t \text{ OR...})$$

Biologically, one could interpret this relation as steric hindrance; the presence of any inhibitory factor bound to the DNA prevents the transcription activating machinery from binding. Other forms could include a threshold model for activation, or AND clauses indicating complex formation.

In addition to choosing the functional form of the Boolean update rules, one must also decide the algorithm for applying the updates during simulation. The simplest choice is to update every node in the model at every step. This is called synchronous updating and it is deterministic, meaning that repeated simulations will yield identical results. Further, with deterministic synchronous updating, the simulation is guaranteed to reach attractor states. An attractor is a steady state or fixed point of the simulation. Attractors have two types: stable and cyclic. A stable attractor is a single state, and when the update rules are applied to it they yield the same state again. Thus, the simulation is trapped at that state. Cyclic attractors are similar but comprise multiple states. Once a simulation reaches any state in the cyclic attractor, it is guaranteed to loop through the states in the cycle indefinitely. These attractor states are biologically important because a common assumption is that experimental measurements represent steady states of the underlying biological regulatory network. Thus, one could assume that a model’s attractors correspond to the steady states of the biological system.

Another method, asynchronous updating, also has useful properties. In asynchronous updating, one node is selected randomly and updated at each simulation step. This has several effects. First, attractors are no longer deterministic: repeated simulations from a single starting point can end up in different attractor states. This allows a modeler to capture stochastic phenomena such as cell fate decisions. Repeated simulations of the same initial states with asynchronous updating will predict which proportion of cells will end up in each attractor or cell fate.

Second, if one records the state of each node at each simulation step, then averages over many simulations, one can approximate continuous dynamics with the discrete Boolean model. While this does not have the direct physical interpretation like mass action kinetics (discussed below), it does allow a modeler to capture interesting behavior such as oscillatory or switch-like dynamics.

Compared to differential equations based models, Boolean models are simple to construct and simulate, having no parameters and simple dynamics. Further, their simple functional form and explicit specification of regulatory structure allows easier qualitative and mathematical analysis, when compared to machine learning models. Both of these features will be discussed in more detail below in the context of differential equations and machine learning models, respectively.

1.2.2 Ordinary Differential Equations

Modeling biochemical processes with ordinary differential equations (ODE's) has a long history, dating back (at least) to Alfred Lotka's 1910 description of periodic behavior in an autocatalytic chemical reaction - which itself drew on earlier mathematical work in chemical physics (Lotka, 1910). The success of these models inspired a century of mathematical biologists to describe biochemical processes with differential equations, creating a body of literature that is too vast to effectively summarize here. Instead, I will provide a brief description of the principles that underlie these ODE models and the relative benefits and disadvantages they provide.

A common simplification of biochemical behavior is to assume mass action kinetics, which states that the rate of change in the concentration of a product depends only on the concentration of the reactants multiplied by a rate constant. A model with mass action kinetics is illustrated here with a simple reversible reaction between three species:

$$A + B \Longleftrightarrow C$$

$$\frac{d[C]}{dt} = k_f[A][B] - k_r[C]$$

$$\frac{d[A]}{dt} = k_r[C] - k_f[A][B]$$

$$\frac{d[B]}{dt} = k_r[C] - k_f[A][B]$$

Here, $[C]$ is the concentration of the product, formed in a reversible reaction of A and B. The constants k_f and k_r describe the rate at which the forward and reverse reactions proceed, and consequently the equilibrium concentrations of the reactants and product. This framework has been successfully applied in modeling systems such as MAPK signaling (as reviewed above in D'Alessandro et al. (2015)) and apoptosis (Chen et al., 2007; Howells et al., 2011; Albeck et al., 2008).

As noted above, every reaction in a mass-action ODE model must have a forward rate constant and, if the reaction is reversible, a backwards rate. How to choose the rate constants is a difficult question. Given some data on the concentrations of species in the system at various time points, one may define a cost function that compares simulations of the model to the data. Then one can apply optimization algorithms, such as simulated annealing, to estimate the rate constants which minimize the difference between the simulation and the data. One may also employ Bayesian algorithms to give a probability distribution for each rate constant, conditioned on the data (Shockley et al., 2018).

However, estimation of these rate constants with optimization algorithms requires a large number of simulations of the model with different settings of the rate constants. This is problematic because ODE models can be computationally costly to simulate, requiring complex algorithms to linearize the dynamics of the model at a sufficiently small time resolution to avoid errors or divergence (Petzold, 1983). This is especially true in the case of models that are "stiff". While this term does not have a precise mathematical definition, Moler gives the following qualitative description, "A problem is stiff if the solution being sought varies slowly, but there are nearby solutions that vary rapidly, so the numerical method must take small steps to obtain satisfactory results" (Moler, 2004). Städter et al. (2021), show that many models of systems of biochemical reactions are stiff and can only be solved by specialized integrators.

In comparison, Boolean models are very easy to simulate, with each update to the model's state determined by simple logical equations. This ease of simulation comes at the cost of the reduced expressivity of Boolean models compared to the range and dynamics of ODE's. In addition, the mass action kinetics form of ODE's has a direct interpretation in terms of chemical interactions and is well grounded in statistical physics, while the logical update equations of Boolean models do not easily admit direct physical interpretations. However, as noted above, the each mass action reaction has at least one parameter which must be estimated while Boolean models do not have any parameters, which eases both their construction and simulation.

1.2.3 Machine Learning

Machine learning is an imprecisely defined term, but could generally be described as a family of approaches for fitting nonlinear models with a large number of parameters to data. While Boolean and ODE models attempt to explicitly describe the nature of interactions between biochemical species in a system, machine learning eschews this almost entirely, instead optimizing a generic nonlinear function to interpolate between data points or to separate data into distinct classes. While some classes of functions, such as decision trees, do allow some degree of interpretation by learning simple decision rules (Breiman et al., 2017), many popular methods, such as neural networks are effectively black boxes. Much attention has been given to this problem in recent years, producing a profusion of methods that generate explanations for the predictions of machine learning models. Methods such as LIME (Ribeiro et al., 2016) attempt to explain individual predictions by fitting easily interpretable models (like a linear model) to subsets of the data in the neighborhood of the prediction. Thus, one can understand the local structure of the full black-box model using a less complex surrogate model. Counterfactual reasoning, as applied to machine learning models by Wachter et al. (2017), finds the minimal perturbations to inputs that would generate a desired output. Another more complex method called SHAP (Lundberg and Lee, 2017) uses principles from game theory to assign credit for predictions to features used by the model.

All of these methods provide an explanation in terms of ”importance”, i.e. how much a given feature contributes to a prediction. For LIME, the weights of the surrogate linear model admit a simple interpretation of importance. In counterfactual reasoning and SHAP, importance can be understood as the marginal amount that a feature contributes to a prediction. While these definitions of importance do provide some transparency into how a prediction was made, the explanations of LIME, SHAP, and other interpretability methods lack properties that are important for biological understanding.

In ODE and Boolean models, interactions between components of the model are explicitly specified. By structuring the model as a graph, one can trace the signaling chains that channel a perturbation in one component into a change in another (Palsson, 2006). In comparison to the unit-less SHAP values, interpretations of ODE models are grounded in physical quantities such as rate constants and concentrations, as in Wrede and Hellander (2019), or chemical fluxes through pathways in PyDyno (Ortega et al., 2021).

In comparison, links between interventions and specific outcomes in machine learning models are not grounded in physical interactions or explicit chemical reactions. Thus, methods such as SHAP and LIME are only useful when one does not need to understand the mechanistic processes that generate a specific outcome. This hampers both translational applications and further basic research by obscuring underlying biological mechanisms.

For example, consider an investigation of how cells initiate apoptosis in response to DNA damage. A machine learning model could be trained on markers of DNA damage to predict whether a cell will die. Investigation of counterfactuals or SHAP values could reveal how changes in DNA damage markers contribute to changes in the probability of cell death.

However, a researcher may assume that the markers of DNA damage are not directly modifiable, but are caused by inaccessible extrinsic factors. Instead, they may ask which proteins that respond to DNA damage could be therapeutically modified to increase or decrease cell death. A machine learning model, at best, could only reveal this information indirectly, while an ODE or Boolean model would admit much more direct investigation of this question.

Further, machine learning models may learn to rely on spurious correlations between variables that are present in the training data. If a confounding factor that induces the spurious correlation is not present when applying the algorithm to new data, then the model’s performance will suffer. In contrast, ODE and Boolean are constrained to rely only on interactions that the model builder considers to be reasonable and biologically plausible. These constraints can allow ODE and Boolean models to perform well in novel settings that would stymie a machine learning model.

1.3 Conclusion

In the first part of this thesis (Section 1.1), I described two pervasive, unintuitive properties of biological systems. Cross-talk is one manifestation of the extensive connectivity of cellular signaling pathways. The propagation of signals outside their canonical pathways causes complex phenomena. Examples of this include synergistic effects and cross-inhibition in the MAPK and PI3K pathways.

I also described noise and stochasticity, and showed that they are fundamental features of cellular processes. This is a challenging notion, given the apparent precision of biological systems, especially in development. I showed in Section 1.1.2 how computational modeling revealed insights into the mechanisms that generate and control cellular noise.

Both of these features, noise and cross-talk, are fundamental properties of cellular biology, active across a wide variety of processes and cell types. Again, computational modeling is a key tool for understanding the counter-intuitive phenomena that arise due to the stochasticity and connectivity of biological reaction networks.

However, computational modeling is not a single tool, instead a researcher must choose from a varied set of modeling frameworks. In Section 1.2, I described common frameworks for modeling biochemical systems: Boolean models, differential equations, and machine learning. As the remainder of this thesis will focus on Boolean modeling, I specifically focused on the comparing the benefits and disadvantages of each framework in relation to Boolean modeling. To summarize, differential equations are expressive, able to model the full dynamic response of a system to changing conditions. However, this expressivity comes at the price of having to estimate parameters for each interaction in the model. Parameter estimation is made more difficult due to the high computational cost of simulating ODE models, which must be repeated many times during parameter estimation. In contrast, Boolean models have no parameters and their simulation algorithms are simple and easily parallelized. Machine learning optimizes the parameters of a generic function to make predictions based on input data. While several methods have been developed to understand the structure and relationships between input variables in optimized ML models, these methods do not allow insight into the biochemical or physical mechanisms that underlie the system's behavior. By explicitly specifying these relationships, ODE and Boolean models allow researchers to make predictions about the effect of perturbations of intermediate steps in the system.

Altogether, the conceptual simplicity and easy simulation of Boolean models makes them an attractive tool for representing biological systems. Unfortunately, the next chapter will show that building accurate Boolean models is an extremely difficult task. However, automated methods, which I will discuss in Chapter 3, promise to ease this task by efficiently synthesizing Boolean models.

CHAPTER 2

Manual model construction is difficult

In the first chapter, I reviewed several properties of biological systems that make them exceptionally difficult to understand without the aid of computational models. High connectivity, feedback, and noise all defy simple intuitions and the traditional linear pathway models that allow researchers to make direct assertions about the effect of perturbations on a system. However, computational models allow researchers to build and manipulate representations of a biological system that capture complex and unintuitive behaviors. These computational models can take several mathematical forms, which I briefly reviewed with a focus on Boolean models. The Boolean logic framework allows researchers to model a simple coarse-grained view of the system, while also being fast to simulate. However, In this chapter, I will make an argument that creating Boolean models manually is insufficient for the task of capturing an accurate representation of the underlying biological phenomena. I will advance three primary reasons for this, which I will support with computational and mathematical evidence. First, the space of possible models is so vast that a finding a model configuration with precisely the right behavior is extremely unlikely, even for simple, small models. Second, even if one does find a single model with the right behavior, it is very likely that there are hundreds more models with the same behavior, many of which are quite dissimilar from each other. Thus, the first model one finds may not be the best; one must repeat the process to find many more models to have any chance of finding the most accurate one. Finally, I assume that a typical researcher builds models through trial and error, adding and removing interactions from a model one at a time. However, this is a fraught process, as I show that addition or removal of a single interaction can drastically alter the behavior of a model in unexpected ways. In summary, I propose that a model builder must search through a vast, rugged landscape of model space to find many tiny disparate islands of ”good” behavior.

2.1 Experimental Setup

In order to make the arguments that I outline above, I generated a large number of Boolean models and examined the properties of this collection of models. More specifically, I generated all models with four nodes and six to ten total interactions. I exclusively focused on models with dominant inhibition regulation, which I describe in detail in Section 3.2.2. This includes all possible combinations of activating and inhibiting interactions between the four nodes. This generated 11.1 million models. I then simulated every model using synchronous updating with all sixteen possible initial conditions and I stored the resulting attractor states and their frequencies.

2.2 Finding *any* models with specific attractors

Using the collection of Boolean models and attractors described above, I will examine the claim that the space of possible models is so large that a modeler is unlikely to find a model with any specific desired behavior. If we assume that the modeler has a set of attractors that they would like to recreate, their task is then to find the model (or set of models) that has this behavior. I was interested in how difficult this problem would be in general, without assuming that our hypothetical modeler is interested in any one particular set of attractors. So, I grouped models into sets that have the same attractors, which I term attractor-equivalent sets. Then, I calculated the size of these attractor-equivalent sets.

An immediate observation was that about 10% of models (1M out of 11M) have attractors that are either all zeros or all ones, which I term "collapsed" attractors. This represents a potential pitfall, in that a modeler has to avoid a large part of model space that produces uninteresting, biologically implausible behavior. Further, as shown in Figure 2.1, the types of interactions in models with collapsed attractors are not markedly different from the models with more diverse attractors. Most models with diverse, "mixed" attractors have approximately the same proportion of activating and inhibiting interactions as models with collapsed attractors. Thus, a modeler can not use the proportion of interaction types as a simple heuristic to avoid models with collapsed attractors. Instead, to generate models with complex behavior a modeler has to rely on more complex and non-obvious features of the models.

However, the question remains of how difficult it is to find a model with any specific set of attractors. In Figure 2.2 I show the cumulative distribution of the number of models in attractor-equivalent sets of increasing size. Stated differently, this shows how many models are in attractor-equivalent sets of at most a given size. As indicated by the orange vertical line, half of all models are part of sets of size 187 or less. Thus, given any desired attractors, there is a 50% chance that a modeler would have to find one of (at most) 187 out of 11 million possible models. Further, 130 thousand attractors have exactly one corresponding model. This clearly illustrates that finding a Boolean model with a specific set of attractors is an extremely difficult search problem, with most attractors corresponding to a tiny fraction of the overall space of possible models.

2.3 Finding *all* models with specific attractors

On the other contrary, the top right hand side of Figure 2.2 shows that many models are part of large sets of models with equivalent attractors. The largest set of attractor-equivalent models in the collection has 9906 models. While finding a single model from this set may be comparatively easier, a modeler faces another question: is this the most accurate representation of the underlying biology? While two (or more) models may have equivalent attractors, they might achieve this through very different mechanisms. These different mechanisms would respond in different ways if one perturbs them. Applying the same perturbation to the

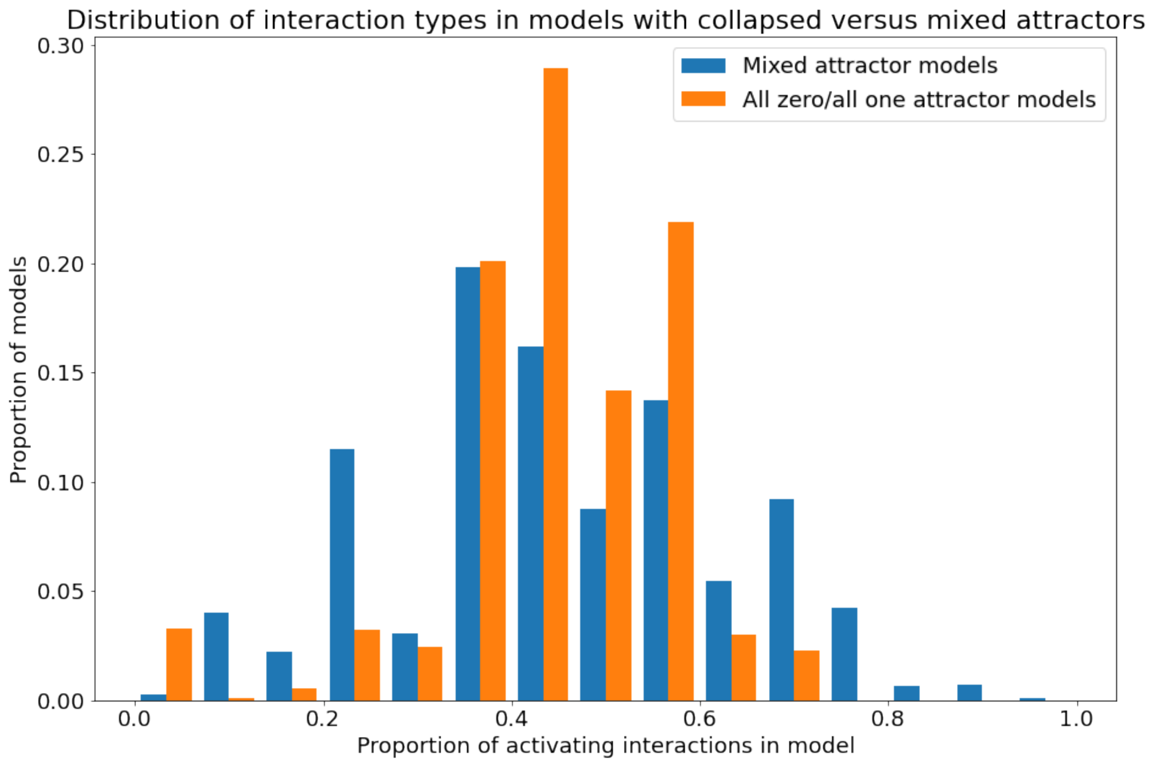


Figure 2.1: Distribution of interaction types in models with collapsed versus mixed attractors. This compares the proportion of types of interactions in models with attractors that are either all zero or all one (collapsed) to models with more diverse, mixed attractors. Mixed attractor model statistics are shown in blue and collapsed attractor models in orange.

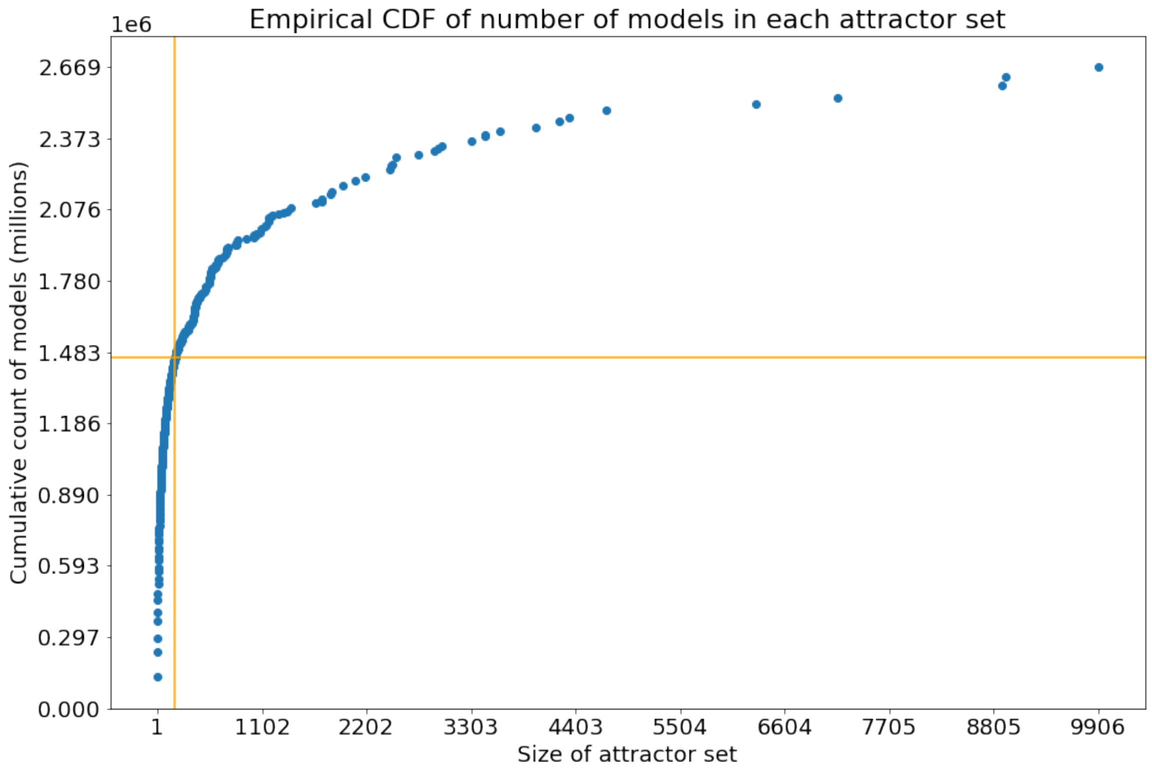


Figure 2.2: Cumulative distribution of the number of models that are in sets of a given size. Each point represents a set of models with same attractors. Sets increase in size from left to right. This shows the cumulative count of how many models are part of sets of *at most* a given size. The orange line represents the median of 187, i.e. 50% of models share their attractors with fewer than 187 other models.

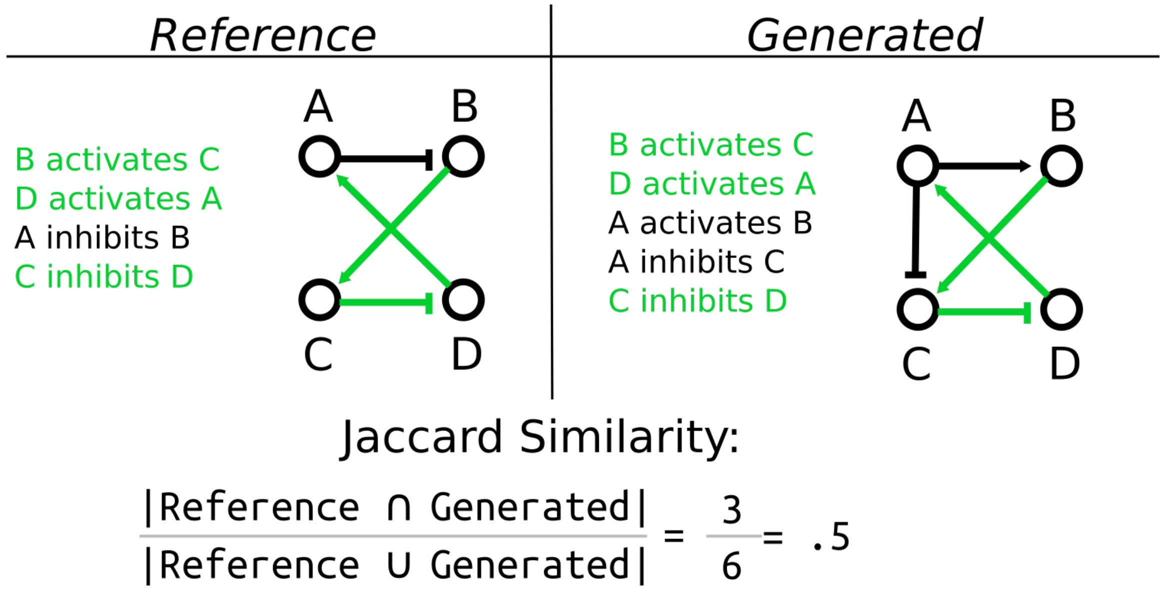


Figure 2.3: Example calculation of Jaccard similarity. We compare the structural similarity between two Boolean models by computing the Jaccard similarity between their sets of interactions. Here, shared interactions between the two models are highlighted in green, while interactions that are unique to each model are in black. In this example, the two models share three interactions in common, but have three more that are unique to each model. Thus they have a Jaccard similarity of $3/(3+3) = 3/6 = 0.5$

biological system would then show that one model's response is more accurate than others. Thus, to find the one model that responds most accurately in all conditions, one has to find all the models that are consistent with the original conditions. Based on the collection of models analyzed here, this could mean generating nearly ten thousand models by hand, which would be a monumental task. However, one might assume that all the models in attractor-equivalent sets are nearly identical, perhaps differing by only a few interactions. This would, presumably, ease the task of generating the full set of attractor-equivalent models, with each variation requiring only a few edits from a known set of interactions.

To test this assumption, I inspected the sets of attractor-equivalent models more deeply. For each model set, I compared the similarity between their interactions using the Jaccard similarity metric. First, I converted each model's update rules into sets of interactions, with each term in the update rule represented as a source node, destination node, and interaction type (activating or inhibiting). Figure 2.3 shows an example of the Jaccard similarity applied to models.

Using the model similarity metric, I then attempted to answer the question of "how diverse are models with the same attractors?". I was particularly interested in the maximum differences, i.e. which models are most different while still generating the same attractors. Figure 2.4 shows the distribution of the smallest similarities between two models in the same attractor set, i.e. the difference between the most dissimilar models with the same attractors. The median Jaccard similarity of 0.5 (indicated with an orange line) means

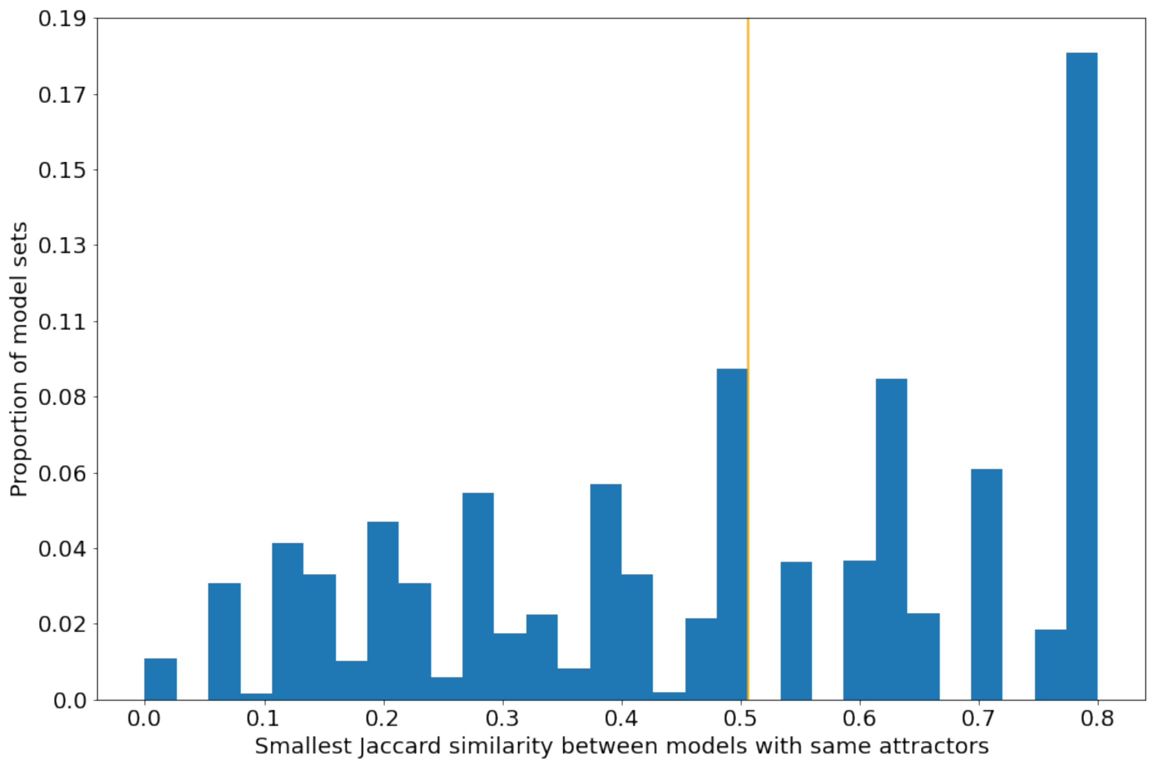
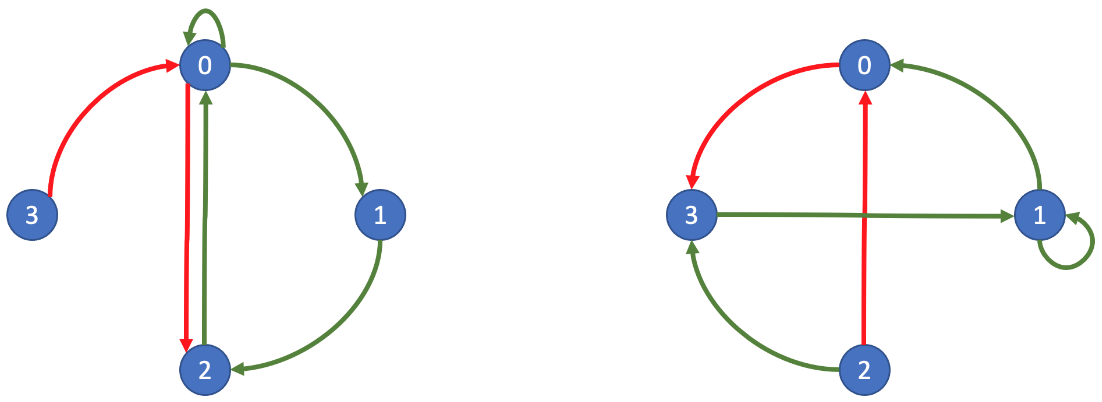


Figure 2.4: Distribution of smallest Jaccard similarities between models with the same attractors. Grouped models with the same attractors into sets, then computed the Jaccard similarity (see Fig. 2.3
) between the attractor-equivalent models. This shows how different two models can be and still have the same attractors. The orange line indicates the median of the distribution.

that in half of all attractor-equivalent model sets there are two models that are more different than they are alike. Strikingly, in 1099 of the attractor-equivalent sets there are two models that have zero similarity, i.e. they do not share any interactions in common. An example of two models that have the same attractors but completely different interactions is shown in Figure 2.5.

This indicates that a modeler could face an immense challenge in constructing all the models that are consistent with a desired set of attractors. Even for a simple system with four interacting species, they may have to construct hundreds or thousands of models. Additionally, these models may be very diverse, having unique mechanisms that don't share any interactions in common. Thus, a modeler can not simply find a single instance of a model with the desired attractors and then make small edits to arrive at the full set of consistent models. Instead, a modeler faces a difficult task, finding (potentially) hundreds of unique and dissimilar models in a space of millions of possibilities.



```
n0 := (n0 or n2) and not (n3)        n0 := (n1) and not (n2)
n1 := (n0)                           n1 := (n1 or n3)
n2 := (n1) and not (n0)              n2 := False
n3 := False                          n3 := (n2) and not (n0)
```

Figure 2.5: Two models with the same attractors but no shared interactions. These two models have the same attractors, but they share no interactions in common (i.e. Jaccard similarity of zero).

2.4 Discontinuities in model space

In the previous two sections (2.2 and 2.3), I showed that it is difficult to find a single model with the desired behavior and subsequently difficult to find all the other models with same behavior. However, we could reasonably assume that a modeler might not generate models by purely random selection of interactions. Instead, they might start with a model that has attractors that are similar to the desired attractors, then iteratively refine the model to arrive at the correct behavior. This assumes that each small change to the model produces small changes in the attractors, which the modeler could predict to substantially reduce the size of the search space.

Here I will show how I tested the hypothesis that adding or removing one interaction changes most models' attractors by a small amount. I randomly sampled one thousand models from the collection used in the previous two sections then generated all variations of each model with one interaction added or removed. I then simulated all the new variant models and compared their attractors to their corresponding original (unedited) model's attractors using a novel distance metric. I will defer description of the details of this edit distance metric to Section 3.2.3, where I describe it as part of my approach to automated model synthesis. Briefly, it calculates how many states are different between two sets of attractors. All models that I analyze here have 4 nodes and were simulated from 16 initial states. This gives each model a total of 16 attractor states. The largest possible edit distance of 64 would correspond to all 16 attractor states having different values from the comparison attractor states. This distance metric allowed me to quantify the change in attractors induced by a single change to a model.

Figure 2.6, shows the overall distribution of changes in attractors caused by a single alteration to a model's

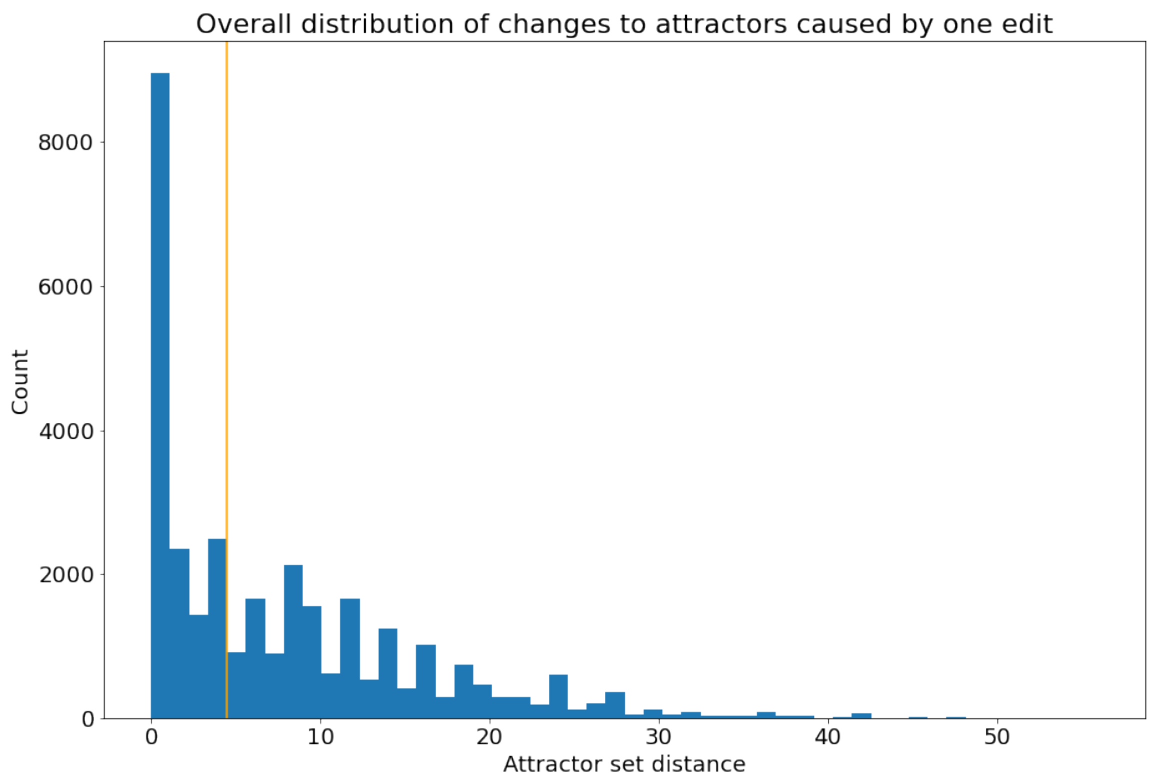


Figure 2.6: Distribution of edit distances between attractors of models after one change and original model's attractors. I randomly sampled 1000 models. Then I made all possible additions or deletions of individual interactions and simulated each variation of the model. I then compared the original attractors to the attractors of the edited model. The edit distance is detailed in Figure 3.2. The orange line indicates the median of the distribution (edit distance of 5).

rules. The median edit distance of 5 represents a relatively modest alteration to the behavior of a model. This indicates that trial and error could be a viable strategy for finding small changes that push a model towards desired attractors. Nonetheless, Figure 2.6 also shows a longer tail of changes that significantly change the attractors. This is supported by Figure 2.7, which shows the distribution of the largest attractor changes for each model that were induced by a single change to the model. This shows that half of models have one change that results in an distance of at least 24 between the original and new attractors. This corresponds to at least 6 out of the 16 total attractor states being completely different. Again, this represents a hurdle for a modeler: they may expect that adding an interaction to a model may change only a few attractor states, but instead in most models one change can significantly alter their behavior. A dramatic example of this phenomena is shown in Figure 2.8, which shows that adding a single inhibitory interaction to the model causes a 56 edit change in attractors. This means that nearly every attractor (14 out of 16) was completely different after adding a single interaction to the model.

While an experienced Boolean modeler may see this change to the model's update rules and easily rec-

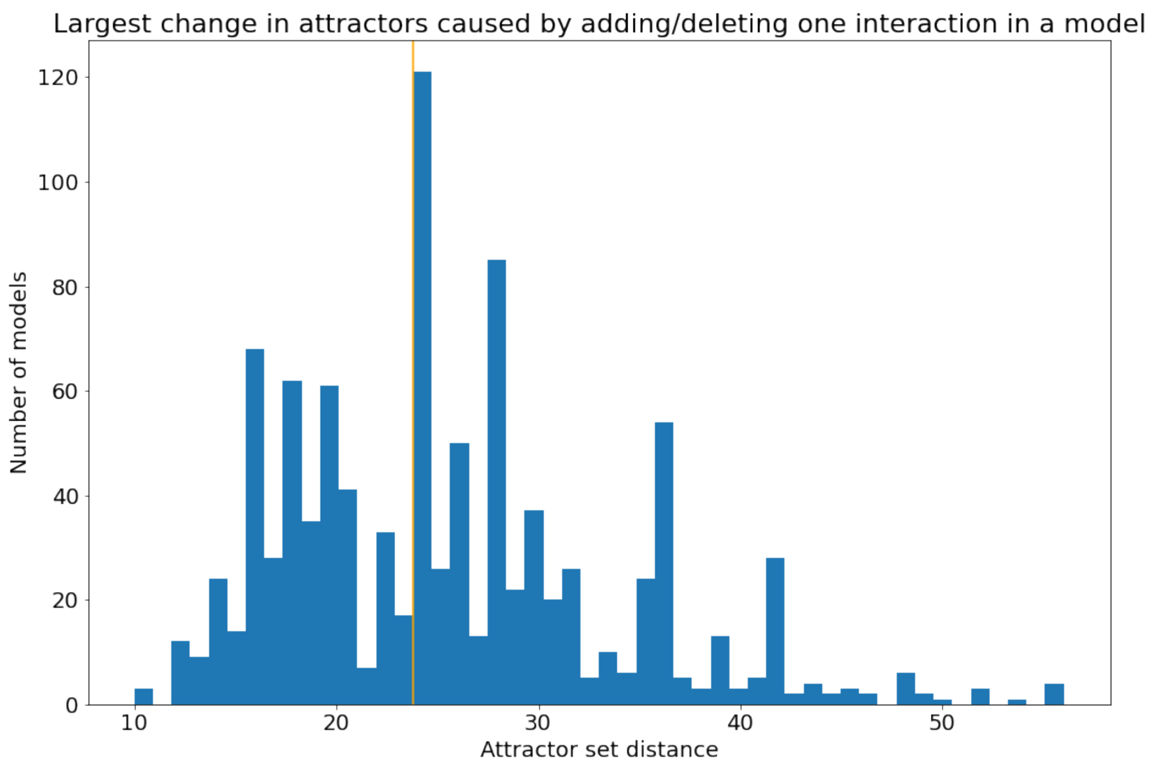
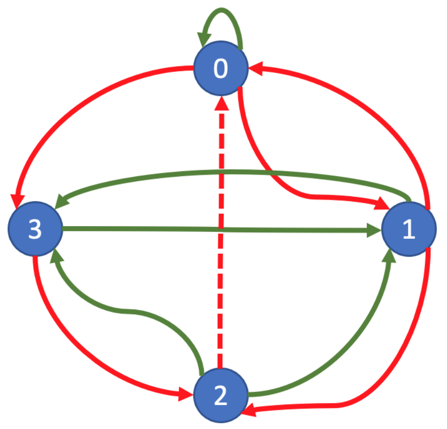


Figure 2.7: Distribution of the largest changes in attractors induced by a single change to a model. Using the same process described in Fig. 2.6, I calculated the largest changes to each model's attractors that could be caused by adding or removing one interaction. The orange line indicates the median of the distribution (edit distance of 24)



```
n0 := not (n1 or n2)
n1 := (n2 or n3) and not (n0)
n2 := not (n1 or n3)
n3 := (n1 or n2) and not (n0)
```

Figure 2.8: Model with large change in attractors caused by one new interaction. Green lines indicate activating interactions and red indicate inhibiting. The corresponding rules are shown below in text form. The dashed red line (annotated in orange in the rules) was added to the model. This one new interaction almost completely changed the attractors, resulting in an edit distance of 56 out of a maximum of 64.

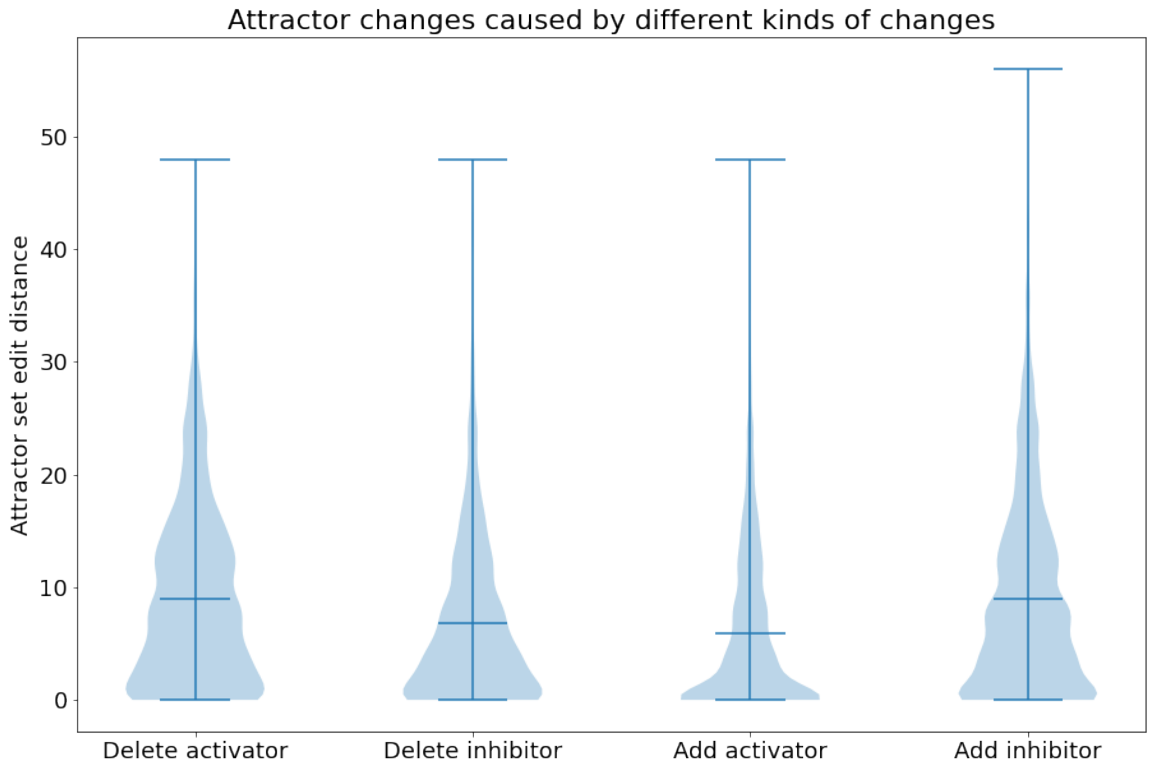


Figure 2.9: Distributions of edit distances across all possible changes to models, separated by type of change. Each violin plot shows the edit distances caused by the type of change.

ognize its significance, is not immediately clear what features separate this change from more benign interactions. When we look at the edit distances across all the tested changes, there are not drastic differences in the degree of attractor changes caused by any particular type of change. As shown in Figure 2.9, the largest changes are caused adding inhibitors and deleting activators, but the median changes are broadly similar across all change types. This argues against the utility of a simple heuristic for understanding the types of changes to a model that are likely to drastically affect its behavior. This aligns with the failure of the simple heuristic for selecting models with collapsed attractors that was based on the proportion of interaction types shown in Figure 2.1 and described in Section 2.2.

To summarize, many changes to models will only slightly change their behavior. Nonetheless most models have at least one change to their interactions that will drastically change its attractors and this is not easily predicted with simple features of the change. These unpredictable discontinuities in model behavior complicate the usual strategy of building models iteratively.

2.5 Conclusion

Taken together, these empirical observations of a large collection of models show that constructing Boolean models by hand is extremely difficult. As I show in Section 2.2, finding a model with precisely the desired attractors is likely to be time-consuming, as one must search through an enormous space of possibilities. Further, the typical strategy for searching (trial and error) is made more difficult by discontinuities in the space of model behavior, which could complicate iterative model building. As I show in Section 2.4 a single change to a model can often drastically and unpredictably change its behavior. Finally, when a researcher finds a model with desired behavior, this may only be one of thousands of models that have the same attractors, raising the possibility that another completely different model may have the same behavior but a more accurate mechanism. Thus, a researcher would have to repeat the search process an untenable number of times.

A skilled model builder may be able to recognize patterns in data and associate these with specific structural motifs, which would aid them in quickly constructing data-consistent models. An algorithm which could quickly find and exploit these same favorable structural motifs could generate models with data consistent behavior much more quickly and in much higher quantities than a person working by hand. In the following sections I describe my work in building and testing the computational tools required to allow a algorithm to generate Boolean models.

CHAPTER 3

Automated computational search methods can help overcome the difficulties of constructing models

As described in the Chapter 1, computational models are key tools for understanding the complex connections and dynamics present in cellular systems. However, Chapter 2 showed that these models are inherently difficult and time consuming to create. A modeler must search through a vast, discontinuous space of possibilities to find a relatively small set of models with the desired behavior. This set of models with ”good” behavior may still be quite large, spanning thousands of variations, many of which may share vary few interactions in common. This motivates the use of automated techniques that can find sets of interactions that lead to the desired behavior. In this chapter I will describe a method I developed for automatically constructing Boolean models. However, I am not the first to pursue this line of research, and in the section below I will first describe some previous efforts to automatically construct Boolean models.

3.1 Previous approaches to Boolean model synthesis

The first attempt to synthesize Boolean models with specific steady state behavior was proposed in by Pal et al. This work proposed two algorithms. In the first algorithm, a random search generated update rules while the second algorithm randomly generated a full state transition graph from which corresponding rules were deduced. In both cases, the resulting models were checked for consistency with the desired attractors and inconsistent models were discarded. However, these algorithms are limited by scalability of the naive random search method and would struggle to produce data-consistent models with more than a few interacting species.

More recent approaches make use of more efficient algorithms, and allow generation of larger models than in the approaches proposed by Pal et al. These techniques for synthesizing Boolean models can be divided into two categories: constraint solving and optimization.

3.1.1 Constraint solver based methods

Constraint solving based methods pose the problem as a series of abstract logical constraints, e.g. that each species can have only two states and that the update functions must be consistent with steady states described in the data. These constraints are encoded as Boolean logic statements or in a more abstract formalism such as answer set programming (ASP) or satisfiability modulo theories (SMT) problems. Specialized solvers then find a set of models which satisfy all the constraints specified by the data and the modeling assumptions.

SMT solvers and other logic programming techniques are specialist tools, designed primarily for use by practitioners of non-biological fields such as operations research or computer security. To ease applications to biology, Yordanov et al. (2016) introduced RE:IN, a domain specific language to describe experimental observations, prior knowledge, and initial conditions. Programs written in RE:IN's language are converted into constraints in the SMT formalism and solved to generate a set of models satisfying the constraints.

Fisher et al. (2015) also employ SMT solving but introduce a unique modeling assumption for single cell data. Single cell RNA expression measurements are assumed to represent individual states which are connected to other similar, observed states, forming a data-derived state transition graph. The sequence of state transitions between an assumed starting and ending state in the graph are encoded as SMT constraints. Solving these constraints generates a set of update rules which are consistent with the given state transition graph.

While Fisher et al. assume that scRNA-seq data encodes a full continuum of initial, intermediate, and final cellular states, most previous work assumes that the data encodes only the steady states of the system. Chevalier et al. (2019) make this assumption but allow an incrementally more complex constaint: including explicit specification of the reachability (or not) of steady states from each initial condition. Further, this work utilizes a novel updating scheme, termed ”most permissive”, which is a generalization of asynchronous updating. These constraints are encoded in the Answer Set Programming formalism; the constraints are then solved to produce consistent models.

3.1.2 Stochastic optimization based methods

In general, optimization based methods generate a novel set of update rules, simulate them to get the corresponding steady states, compute goodness of fit scores between the simulated and desired attractors, then use heuristics to suggest a new variation to the model, repeating the process until convergence.

Saez-Rodriguez and colleagues have developed several related methods that fit the topology and regulatory interactions of Boolean networks to measurements from perturbation assays. This work was first introduced in Saez-Rodriguez et al. (2009). Their method, named CellNOptr, converts a prior knowledge network (e.g. a subset of an interaction database) into a Boolean model. They pre-process the network to remove unidentifiable nodes, then apply a genetic algorithm to select a topology and update rules that best fit experimental data.

Similar to CellNOptr, PRUNET applies a evolutionary algorithm to select a model with a subset of interactions from a prior knowledge network, simulates the model, then compares the steady states to observed data (Rodriguez et al., 2015). BTR applies a similar procedure to select a model topology, with a key difference being the asynchronous updating scheme (Lim et al., 2016). This allows non-determinism and multiple possible steady states from a single initial condition. To account for this non-determinism, Lim et al. introduce a scoring function (termed Boolean state space scoring) that can incorporate this additional complexity.

Similar to BTR, Dorier et al. (2016) uses a genetic algorithm to synthesize an asynchronously updating Boolean network. In contrast to BTR, this method assumes that each measurement comprises two steady states connected by a perturbation. Further, this method selects which species to include in the model as well as interactions between them.

Many of the previous methods are primarily concerned with better understanding or manipulating the regulatory structure of biological networks. Crespo et al. (2013) instead use Boolean models to impute the expression of unmeasured species. Their method first expands a core model to include unmeasured species that interact with species in the core model. Then, they employ an optimization algorithm to select a subset of the expanded network with steady states that match the measured species. The imputed value of the unmeasured species is their steady state value from the simulation of the optimized model.

3.1.3 Hybrid Approaches

Prugger et al. (2020) can be be seen as a bridge between optimization and logic solver based methods. The authors assume that attractors are probabilistic, with each initial state having a probability of reaching each attractor. They use a novel logic based approach to generate a stochastic state transition graph that has the desired attractors. They infer the update rules that are consistent with the state transition graph and asynchronously simulate the inferred rules to determine attractor state probabilities. To improve consistency between the attractor probabilities and the data, they use a genetic algorithm to alter the state transition graph so that it more closely matches the desired attractor probabilities. This mix of optimization and logic solving combines the benefits of both approaches, and allows them to model non-deterministic cell fate decisions like epithelial to mesenchymal state transitions in cancer.

Aghamiri and Delaplace's Taboon algorithm (Aghamiri and Delaplace, 2020) could be viewed as an inverse to Prugger et al . Prugger et al. optimizes a state transition graph and infers the corresponding rules. Conversely, Taboon directly generates rules then simulates them to determine whether the corresponding state transition graph matches the desired attractor distribution. To generate the rules, they first solve a satisfiability problem to find a set of possible update rules that are consistent with the steady states. They then apply an optimization algorithm called Tabu search to find the subset of consistent formulas that generate steady states that match experimental observations.

3.1.4 Comparison between approaches

While both optimization and constraint solvers solve the same problem, their differing computational approaches offer several advantages and disadvantages when compared to each other. In terms of scalability and computational resources, constraint solvers are more efficient than optimization algorithms. One reason for this is that each round of optimization requires simulating many new variations of models. In contrast, constraint solvers exploit mathematical structures such as convexity, lower/upper bounds (Land and Doig, 1960), or local constraint consistency (Dechter, 2003) to efficiently find the space of models that satisfy all constraints and therefore generate the desired behavior by construction, requiring no simulation of the models.

However, choosing a set of constraints that enforce the desired attractors is far from trivial. For example, in Chevalier et al. (2019) the Answer Set Programming approach requires formulating at least forty constraints in highly abstract terms that indirectly specify the desired behavior. Specifying a novel behavior or perturbation, such as a transient knockout due to drug treatment, would require significant and unintuitive changes to the constraints. In contrast, optimization based approaches are conceptually much simpler, requiring only a search algorithm, specification of a model's rules, and a Boolean model simulator. Arbitrary alterations to the simulation, like the aforementioned temporary knockout, or addition of complex model structures, such as multi-cellularity, are implemented as direct changes to the model or simulator and are easy to test and verify.

As a final note on the limitations of constraint solvers, they only generate models that perfectly satisfy all constraints. However, this assumes that models are only valid if their behavior perfectly recreates the input data. Given the noisiness of cellular data and the imperfect process of binarizing continuous measurements, one may want to find and analyze models that generate only partially consistent behavior. In the worst case, the constraint solver may find zero models that perfectly satisfy the constraints. In contrast, optimization approaches generate models with a spectrum of behaviors. Even when no models are exactly consistent with a desired set of attractors, optimization approaches can generate models that are as close as possible to the correct behavior. The user is then free to choose a set of models with the most biologically valid behavior based on their own intuition or supporting evidence. As mentioned above, the flexibility of optimization approaches comes with the penalty of high computational cost associated with simulating each proposed model. This can be mitigated by parallelizing the search and simulation procedures, which I will explore and discuss for our method in Section 3.2.2.

3.2 Applying Monte Carlo Tree Search (MCTS) to model synthesis

Given the flexibility and conceptual simplicity of optimization approaches, I decided to explore their application to Boolean model synthesis. Most previous approaches (see Section 3.1.2) use genetic algorithms for generating and optimizing the Boolean models. While genetic algorithms have a long history, dating back to 1970's (Holland, 1975), more recent research on game playing algorithms introduced a very effective search technique: Monte Carlo Tree Search (MCTS). Briefly, MCTS is a discrete optimization algorithm that iteratively explores search space, choosing possible combinations of actions from a tree structure that is augmented with statistical bounds.

Monte Carlo Tree Search was introduced in Kocsis and Szepesvári (2006) as an algorithm for selecting the best moves in games like Go or chess. The effectiveness of this family of algorithms in game playing was quickly proven by Coulom (2006), who applied a close variant of MCTS to win a tournament of Go playing algorithms. Then, in 2016 a variant of MCTS that employed neural networks defeated the best human players of Go and later proved extremely powerful in playing other games such as chess (Silver et al., 2018). These games feature extremely large spaces of possible moves, comparable in size to the space of possible Boolean models. The success of MCTS in finding good subsets of Go moves suggested that it may be an efficient algorithm for finding good sets of interactions in Boolean models. To my knowledge, MCTS has not been applied to Boolean model synthesis.

I will explore the MCTS algorithm in greater depth in Section 3.2.1 below, explaining how it functions and show in Section 3.2.2 how to adapt it to the task of Boolean model synthesis.

3.2.1 Overview of Monte Carlo Tree Search (MCTS)

The most basic form of MCTS only requires two inputs from the user: a list of possible actions and an evaluation function. In the case of Boolean model synthesis, the list of actions is the set of all possible interactions that can be added to a model. As I will describe in more detail below, each iteraction of MCTS chooses one interaction from this list. Combinations of interactions then form branches of a search tree, which MCTS explores or prunes based on the scores from an evaluation function. The evaluation function is simply any function that assigns a score to a combination of actions. For Boolean models I developed an evaluation function that measures the similarity between the model's attractors and the desired attractors using a metric that I describe in Section 3.2.3.

Using these two inputs, MCTS then repeatedly iterates through four phases: selection, expansion, rollout, and backpropagation.

During selection, MCTS descends the search tree, choosing the child branch using a statistical estimate of the quality or value of the actions on that branch. Upon reaching a leaf node, MCTS expands the leaf by creating new child nodes for all possible subsequent actions.

After selecting and then expanding a leaf node, MCTS performs a random rollout. This consists of randomly selecting actions, stopping when a special ”stop” action is chosen. After stopping, the evaluation function is calculated on the selected branch's actions plus actions from the random rollout. The purpose of the random rollout is to provide a stochastic estimate of the best score that can be achieved by a given branch of the search tree.

In the case of Boolean model synthesis, the model corresponding to the branch's interactions is simulated and the evaluation function is the edit distance (Fig. 3.2) between the model's simulated attractors and the desired attractors specified by the user.

This estimate is then improved through the backpropagation process. Each node in the search tree maintains two statistics: number of visits (N_v) and best similarity (D^*). N_v is the number of times the selection step has chosen a branch containing the given node. After simulating and scoring a rollout, MCTS ascends the search tree from the leaf towards the root. At each node in this path, it increments the number of visits, N_v. If the similarity of the rollout $\mathscr{D}$ is greater than D^* then it sets $D^* := \mathscr{D}$. These statistics are used to calculate the upper confidence bound (UCB), which guides the selection process, as described above.

$$\text{UCB} = \mathscr{D}^* + c\sqrt{\frac{\ln N_v(n_i)}{N_v(n_j)}}$$

where $N(n_i)$ is the number of visits of the current node n_i and its child node n_j. c is an exploration constant. The exploration constant is a hyperparameter that balances exploration vs exploitation in the search. I use an exploration constant that decays with each iteration, favoring exploitation of high quality branches at later iterations.

Figure 3.1 shows an overview of the MCTS algorithm applied to Boolean model synthesis. The search tree is shown on the right, as well as examples of the visit statistics and scores for each branch. The branch labeled in gray is represented on the left side as the sequence of models corresponding to each labeled node in the search tree. This also illustrates the use of the UCB to select branches for exploration. Branches with a low number of visits but high evaluation scores will have a high upper bound, leading to higher exploration, demonstrated by the left-most branch. Other branches that have been extensively explored (right-most branch) have a lower upper bound and will have a lower probability of further exploration, as will branches with low evaluation scores.

One cycle of selection, expansion, rollout and backpropagation constitutes an iteration of the algorithm. The user can choose a fixed number of iterations or a time limit at which to halt the search. Once this limit is reached, the selection procedure is run from the root node of the search tree, choosing the branch that yielded the best models. The search is then restarted after adding the selected interaction to a base model. All subsequent search steps will then include the selected interaction. Restarting the search with a partially constructed model allows MCTS to efficiently probe deeper into the search space, following branches that consistently yield good results.

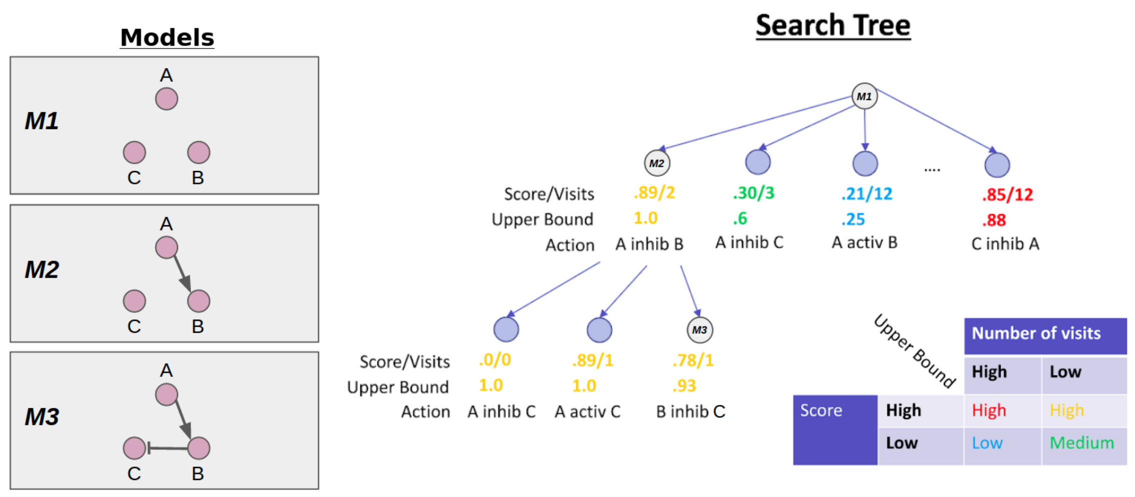


Figure 3.1: Overview of MCTS. On the left are the Boolean models corresponding to the branch of the search tree shown on the right, denoted M1, M2, M3. At each node in the tree, we also show the average score of models on the branch and the number of times the MCTS algorithm has visited the branch. These statistics are used to calculate the upper bound. In the bottom right, we show a conceptual overview of the functional form of the upper bound. In short, MCTS will aggressively explore branches with high scores but low number of visits. More exploration (i.e. a higher visit count) will progressively lower the upper bound until MCTS chooses another branch to explore.

3.2.2 Model Construction and Simulation

To apply Monte Carlo Tree Search to Boolean model synthesis I developed methods for constructing and simulating Boolean models which I will describe in Section 3.2.2 as well as a method for scoring the similarity of their attractors to experimental data, described in Section 3.2.3

As I describe above, MCTS iteratively selects and evaluates actions. For Boolean model synthesis, each action corresponds to adding an interaction to a Boolean model. These interactions can be either activating or inhibiting. Each species can have many activating or inhibiting regulators, which determine its state at the next update step. We use the dominant inhibition form of update rules. This means that if any inhibiting regulator is active then its target is inactivated. If no inhibitors are active and at least one activator is active, then the target is activated. Mathematically, this takes the following form:

$$x^{t+1} = (act_0^t \text{ or } act_1^t \text{ or ... } act_n^t) \text{ and not } (inh_0^t \text{ or } inh_1^t \text{ or ... } inh_m^t)$$

Here, x is the species in the model that will be updated, $act_{1...n}^t$ and $inh_{1...m}^t$ are the states (at time t) of other species in the network that regulate the target node. Both act_i and inh_i can be a single species or composites of of two or more species connected by an *and* clause, e.g. (a and b).

During the search, the update rules in the Boolean models are maintained in a parse tree representation. This eases the manipulation of the model's rules, with each new interaction added as new leaf on the parse tree. During simulation, this tree is converted to a logical expression in Python code, which is then evaluated by the Python interpreter. In future work this representation could also allow this method to convert the tree representation to other forms that allow more complex or efficient simulators, such as numpy (Harris et al., 2020) or PySB (Lopez et al., 2013).

After adding a new interaction to a model, we remove the corresponding inhibiting and activating interactions from the action list. This prevents MCTS from adding both activating and inhibiting interactions from a single source to a single target species. Users can apply more complex manipulations to the list of possible interactions to inject their prior knowledge of the system. For example, one could restrict possible interactions to only those in a database of transcription factor and gene pairs or from results of a tandem affinity purification experiment. Further, one could limit the number of interactions that a given species participates in by removing all interactions that include that species once the limit is reached during the search.

Each action has a user-defined prior probability of being selected. By default the actions are given an identical uniform prior probability. However, these priors can be manually specified based on preferences of the user to bias the search towards interactions that have more experimental evidence or a higher degree of certainty associated with them. The "stop" action has a manually chosen prior probability. Increasing or decreasing the stop action's prior probability corresponds to increasing or decreasing the average number of interactions that are added in random rollouts.

For simulation, my method uses synchronous updating, meaning that at each step the simulator applies the corresponding update equations to every species in the model. My simulator detects both stable and cyclic attractors by tracking previous states and halting simulation when the state matches a previously simulated state.

3.2.3 Attractor Similarity Scoring

I developed an edit distance metric that compares the user's desired attractors to a model's simulated attractors. An example of the execution of the algorithm that computes the metric is shown in Figure 3.2. This distance is used as the evaluation function, guiding MCTS towards models that generate steady states that are similar to the data.

At each step of the distance calculation, we identify every attractor state s_i in the simulation set where the occurrence count of the attractor is different between simulation and data ($c_i \neq c_j$). We then calculate "edits", which is changing one attractor state to another. The cost C of an edit is the Manhattan distance between the bit vectors representing the state of the individual species in each attractor. We then find the edit with minimum cost that would maximally reduce $c_i - c_j$. We apply these edits by changing the occurrence count of the edited simulation state, then repeat the process until all occurrence counts are equal between simulation and data. By accumulating edit costs at each step we obtain a total edit distance between simulated and

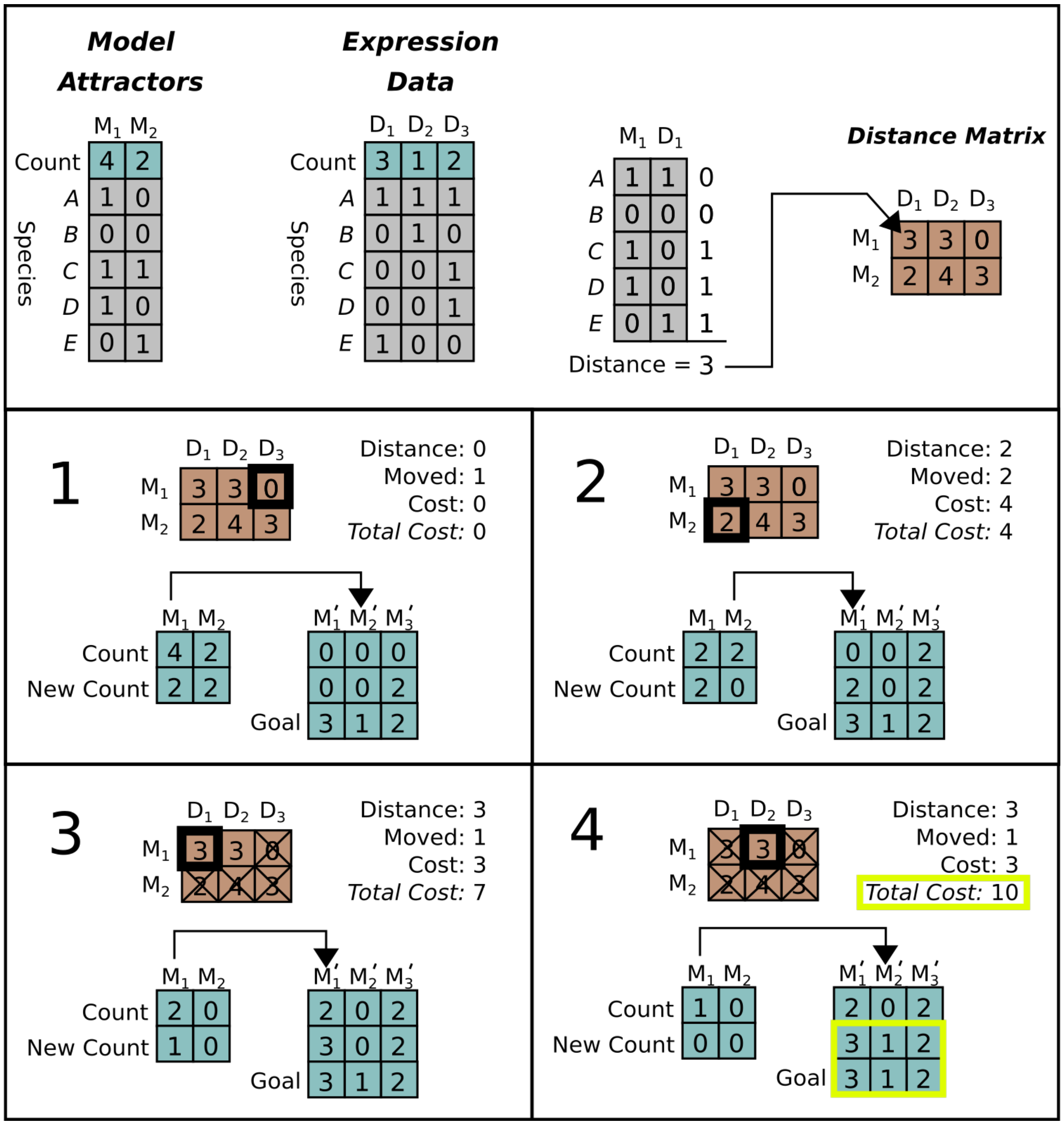


Figure 3.2: Example distance calculation for a system with five genes. *Top row:* Example attractors (left) are generated by simulating a model. The expression data (middle left) against which simulated attractors will be compared is also shown. Note that the data has three unique attractor states denoted D_i while the simulation only has two, denoted M_i. To calculate the first entry in the distance matrix (right) attractor states M_1 and reference states D_1 are compared. Differences are assigned a ”1” while matches are assigned a ”0”. As shown, the distance between states M_1 and D_1 is ”3” because they differ at three genes (C,D,E). *Bottom boxes:* Sequence of edits required to calculate the distance between simulation attractors (M) and data attractors (D). In the first step (1), we choose an edit by selecting the smallest valid distance from the distance matrix. This edit changes one of the M_1 attractors to D_2, but these are already identical, so the cost is zero. In step two (2) we select the next smallest distance (M_2 to D_1, with distance two) and change two attractors for a total cost of four. In step three (3) and four (4) we continue the same process. Note that in step three we remove multiple edits involving M_2 from consideration, as all of the available M_2 states have been edited already. In step four, the new state exactly equals D, so we halt the process with a final edit distance of ten.

measured attractor sets. This is normalized between $(0,1)$ by dividing by the maximum possible edit distance $|s_i| \cdot N_c$, where $|s_i|$ is the number species in the model and N_c is the sum of occurrence counts.

$$\mathscr{D}_{edit}(A^{sim}, A^{obs}) = \frac{\sum_{k=1}^{N_e} C_k}{|s_i| N_c}$$

where N_e is the number of edits required, and C_k is the cost of the edit at step k.

3.2.4 Extensions to MCTS

Reviewing the literature on applications of the MCTS algorithm revealed that several changes to the algorithm have been proposed to improve its performance. Specifically, I was interested in several modifications that adapt MCTS to playing ”single player” games, essentially solving puzzles, which are directly analogous to synthesizing data-consistent Boolean models. Below I will briefly explain three modifications that I applied to the MCTS algorithm.

RAVE (Gelly and Silver, 2011) - In standard MCTS, the estimated best value of a given action (i.e. adding an interaction to the model) is computed by backpropagating similarity scores from all random rollouts of children of that node in the search tree. Thus, the action's value estimate is limited to rollouts from its branch of the tree. However, other branches of the search tree may include the same action, but in a different context. Thus, rollouts from other branches could provide some information about the average value of the action. This is the motivating insight of RAVE. RAVE maintains a list of actions with their number of visits and best values, accumulated across all branches. During selection, a node's value is calculated as a weighted average of the RAVE aggregated value and the standard (branch specific) value.

$$\hat{\mathscr{D}} = (1 - \beta(n_i, k))\mathscr{D}^* + \beta(n_i, k)\mathscr{D}^{rave}$$

Here, $\mathscr{D}^*$ is the branch-specific value, $\mathscr{D}^{rave}$ is the accumulated RAVE value, and $\hat{\mathscr{D}}$ is their weighted average. The weighting factor $\beta(n_i, k)$ is a heuristically determined function of the number of visits to search tree node n_i and a parameter k that determines the number of visits when $\mathscr{D}^*$ and $\mathscr{D}^{rave}$ are weighted equally:

$$\beta(n_i, k) = \sqrt{\frac{k}{3N_v(n_i) + k}}$$

The weighted average is then substituted into the UCB calculation during selection:

$$\text{UCB} = \hat{\mathscr{D}} + c\sqrt{\frac{\ln N_v(n_i)}{N_v(n_j)}}$$

The practical effect of RAVE is to use the global, accumulated value estimate when a specific branch has not been visited very often. As the branch is visited more often, RAVE down-weights the global estimate, preferring the branch specific value estimate. This allows strong estimates of the value of an action early in the search process by leveraging the value of the action on other branches.

Nested MCTS (Rosin, 2011) - The basic implementation of MCTS discards the sequence of actions in each rollouts, only accumulating their evaluation function scores into the statistics that form the UCB. However, the best rollouts achieve high evaluation scores, which strongly indicates specific sequences of actions that should be explored further. Nested MCTS retains the sequence of actions taken by the rollouts with the highest rewards. At each search step, nested MCTS generates a set of random rollouts and stores the sequence of action in the best rollout. In the subsequent search step, more rollouts are generated, but if they fail to find a rollout with a higher evaluation score, then nested MCTS takes the action from the previous best sequence. This allows MCTS to retain very high scoring sequences instead of trying to find high scoring rollouts at each search step. Given the discontinuities in Boolean model search space discussed in Section 2.4, this allows nested MCTS to retain information about the relatively rare actions that can drastically shift the models actions towards the desired behavior.

Branch Retention - As explained above, nested search allows MCTS to retain the best rollouts from previous search steps. However, the statistics stored at each node of the tree are still lost when the search is restarted. I modified MCTS to retain the search statistics accumulated during the search step, an option I refer to as ”branch retention”. Similar to nested MCTS, this allows the algorithm to re-use experiences from previous search steps, rather than restarting the exploration process after taking each action.

Parallelization - We parallelize the search by simply running multiple searches independently with no communication. We implemented our computational pipeline in Python, which has limited facilities for efficient shared memory parallelism. This limited our ability to maintain a single shared search tree that could be independently updated by parallel processes. While this prevents information sharing between processes, this does encourage diversity of the results between independent processes, as they each operate on unique search trees.

3.3 Validation of MCTS for Boolean model synthesis

Here I describe validation experiments, showing that MCTS can find models with a wide variety of desired behaviors and structures. We tested this by randomly generating models, simulating them, and then asking MCTS to generate models with the same behavior and structure as the randomly generated models. Sections 3.3.1 and 3.3.2 describe the process of generating the test models and verifying that their behavior is representative and realistic. Sections 3.3.3 and 3.3.4 describe how we tested MCTS's ability to generate models with the desired behavior and rules.

3.3.1 Random Model Generation

In this section I will describe the method for creating the Boolean models that I will subsequently use to validate MCTS's ability to synthesize consistent models.

I generated action lists corresponding to all possible activating and inhibitory interactions between all nodes. I then uniformly sampled a random number from the interval $[.01, .06]$. This became the threshold value t_a. Then for each action in the action list, I sampled a random number r from $[0, 1]$. If $r < t_a$ I added the action to the model, and discarded it otherwise.

After generating a model, I then generated a set of initial states. First, I sampled a uniform random variable p_{on} from the $(.2, .8)$ interval. I then generated initial states by sampling a Bernoulli random variable n times with $p = p_{on}$ for each species in the model.

I simulated each random model from these associated initial states and randomly selected one of the following checks to apply to its corresponding set of attractors:

$$|\text{stable}| > 1$$
$$|\text{stable}| > 2$$
$$|\text{stable}| > 3$$
$$|\text{stable}| > 1 \text{ and } |\text{cyclic}| < 10$$
$$|\text{stable}| > 2 \text{ and } |\text{cyclic}| < 10$$
$$|\text{stable}| > 3 \text{ and } |\text{cyclic}| < 10$$

Here, $|\text{stable}|$ and $|\text{cyclic}|$ indicate the number of stable and cyclic attractors, respectively. These checks were applied after observing that large number of randomly generated models collapsed to a single trivial attractor of all active or inactive species, consistent with observations from Section 2.2.

I repeated this process 40 times for models with both 8 and 16 nodes, generating a total of 80 models.

Having randomly generated these models, I wanted to validate they had interesting, biologically realistic behaviors. In the following section, I will further examine the characteristics of the initial states and attractors generated by the random models, ensuring that they have diverse and complex behavior, similar to real biological data. This ensures that our validation experiments in Sections 3.3.3 and 3.3.4 are realistic, not asking MCTS to recreate trivial or uninteresting attractors.

3.3.2 Characteristics of Random Models

First, I show properties of of the attractors of the randomly generated models. Table 3.1 shows the mean and median number of stable and cyclic attractors. I also include the mean number of states in the cyclic attractors. This shows that there are, on average, more cyclic attractors than stable attractors in our randomly generated models. However, our data set does include models that have only stable attractors. This tests the ability of the MCTS algorithm to generate models that have both stable and cyclic attractors. Further, the tests applied in the random generation process ensure that the models have multiple distinct attractors. Again, this tests the ability of MCTS to find models with complex, realistic behavior. Typically, biological measurements comprise multiple distinct cellular states and a modeler would like to generate a model that can account for all the cellular states observed in the data. Thus, we would like for MCTS to also generate models that have multiple distinct steady states. Table 3.1 shows that the randomly generated models have multiple attractors, as desired.

Table 3.1: Number of attractors in randomly generated models

	Stable		Cyclic		
	Mean±Std	Median	Mean±Std	Median	Cycle length
8 species	2.6±1.11	3.00	2.90±1.75	3.00	2.65±0.91
16 species	2.4±1.30	2.00	3.59±2.47	3.00	3.37±1.91

In addition, Table 3.2 shows the proportion of active and inactive species in the attractors of the randomly generated models. This shows that the attractors are not in the trivial “collapsed” state.

Table 3.2: Active/Inactive ratio of attractors from sampled models

	Stable		Cyclic	
	Mean±Std	Median	Mean±Std	Median
8 species	0.3±0.27	0.33	0.33±0.20	0.30
16 species	0.2±0.21	0.19	0.28±0.17	0.30

Finally, Table 3.3 show the proportion of active and inactive states in the initial states used in our simulations of the randomly generated models. Again, this shows that the models generate realistic attractors (described above) using initial states that have a diverse composition, testing my method’s ability to generate good models across a variety of initial conditions.

Table 3.3: Active/Inactive ratio of initial states

	Mean±Std	Median
8 species	0.5±0.10	0.47
16 species	0.4±0.16	0.43

3.3.3 Steady State Behavioral Similarity

Having validated that our models will provide a realistic and diverse set of tests for MCTS, I applied MCTS to attempt to find models with the same attractors as the random models. Figure 3.3a shows that the models generated by MCTS at the beginning of the search process poorly matched the behavior of the ground truth models. This is expected, as the MCTS algorithm is effectively a random search process during the initial steps. However, by the end of the search, MCTS reliably found models that had steady states with high similarity to the ground truth models. Across all model sizes, MCTS was able to find several exact behavioral matches, with a majority having $> 95\%$ similarity, as shown in Figure 3.3b.

3.3.4 Rule Set Similarity

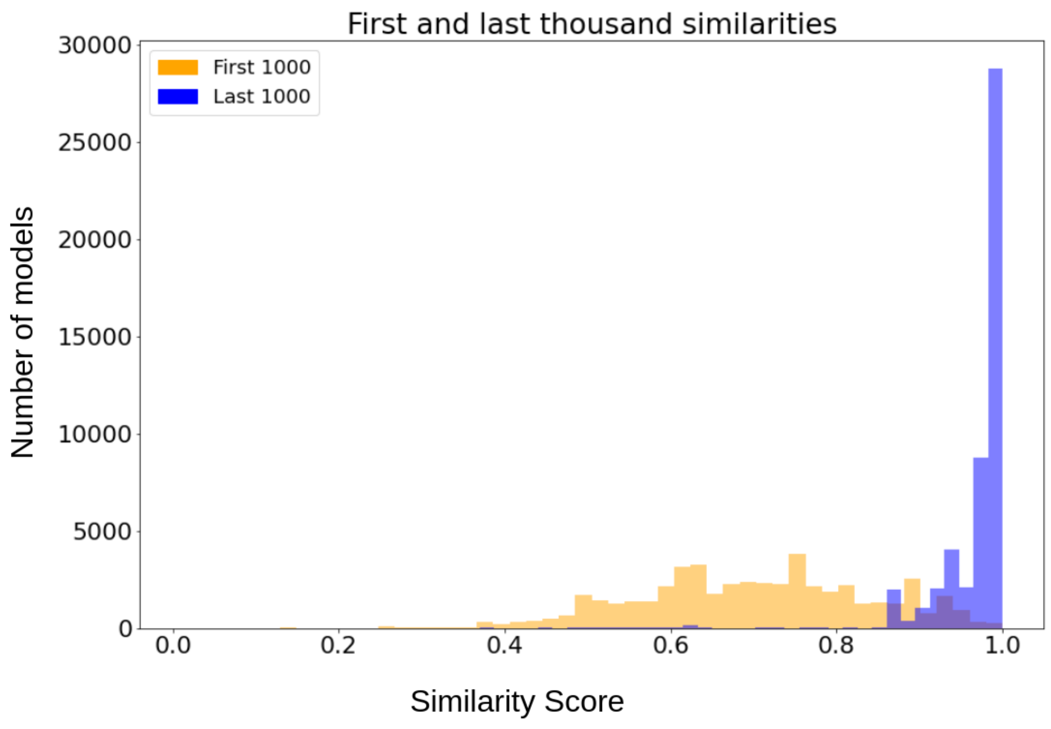
In addition to the steady state behavior of the models, I was also concerned with the content of the update rules generated by MCTS. As shown in Section 2.2 many possible rule sets can have the same steady state behavior. However, Section 2.3 showed that many of these rule sets may be significantly different from each other and, most importantly, different from the underlying biological system. Under novel perturbations or conditions, these models may behave in radically different ways. Thus, I would like MCTS to find models that match the behavior using rules that capture an accurate representation of the underlying system.

To validate MCTS in this regard, I tested its ability to generate models with interactions that are similar to the reference models. In our tests, I quantified similarity by converting the update rules to sets of interactions for both the reference (randomly generated) model and the model generated by MCTS. I then find the Jaccard index between the two interaction sets. This process is illustrated in Figure 2.3. Higher Jaccard indexes indicate that the MCTS model matches the reference interactions well.

With no restriction on the interactions selected by the model search process, MCTS was able to find models with behavior that exactly matched the steady states of the reference models, but using rule sets that differed by as much as 80%. This corresponds to the left-most column of Figure 3.4, with zero reduction in search space, indicating that MCTS was generating models using all possible interactions and no bias towards the true reference interactions.

I next investigated the effect of utilizing "prior knowledge" on MCTS's ability to recover correct rules. As noted above, model inference is an underconstrained problem with many possible models having data-consistent behavior, and so ruling out infeasible interactions can reduce the number of spurious models. I simulated varying levels of prior knowledge by randomly removing incorrect interactions from MCTS's action list, while retaining all of the correct interactions. I repeated the search five times, removing 10%, 25%, 50%, 75%, and then 90% of incorrect interactions from a set of 80 models. The aggregated Jaccard similarities for each percentage are shown in Figure 3.4. For models with both 8 and 16 species, increasing prior knowledge increased the Jaccard similarity to the reference data, as expected.

If we assume that a modeler uses a public database of biological knowledge (e.g. protein interactions) to rule out infeasible or unrealistic interactions, then one should note that most protein-protein interaction databases are much sparser than our highest level of prior knowledge tested above (90% sparsity). For a)



b)

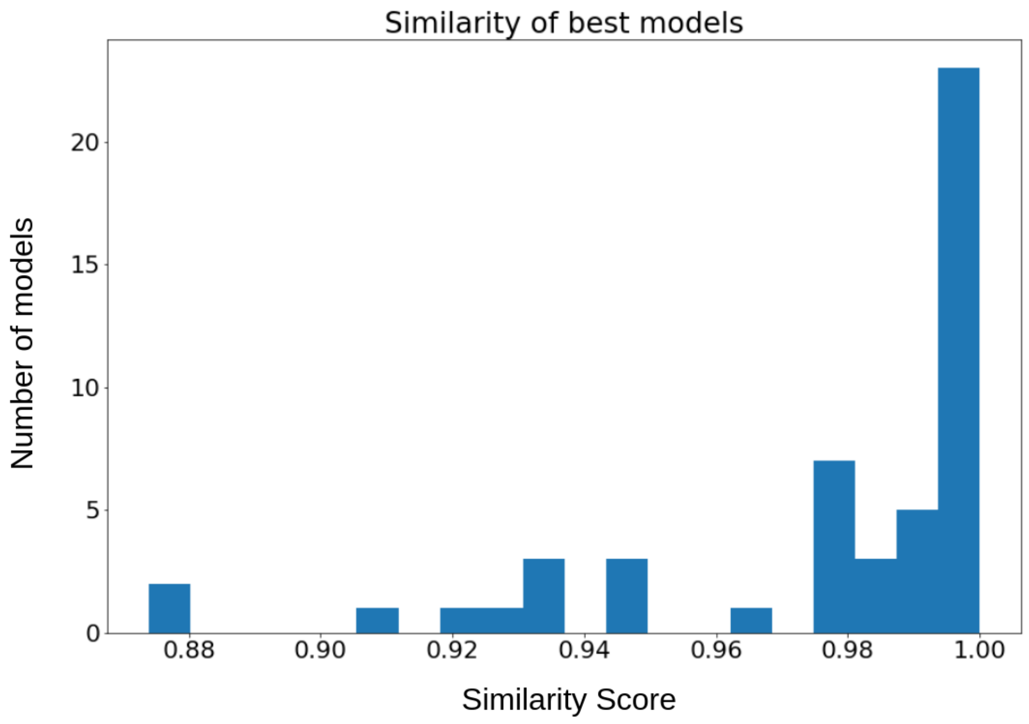


Figure 3.3: a) Orange histogram depicts distribution of similarities from the first one thousand models sampled during the MCTS search. Blue histogram is the distribution of similarities from the last thousand models. The blue distribution is significantly shifted towards higher rewards, indicating that MCTS was systematically sampling good models. b) Distribution of highest reward obtained by each independent search process. Most searches found models with $> 90\%$ similarity.

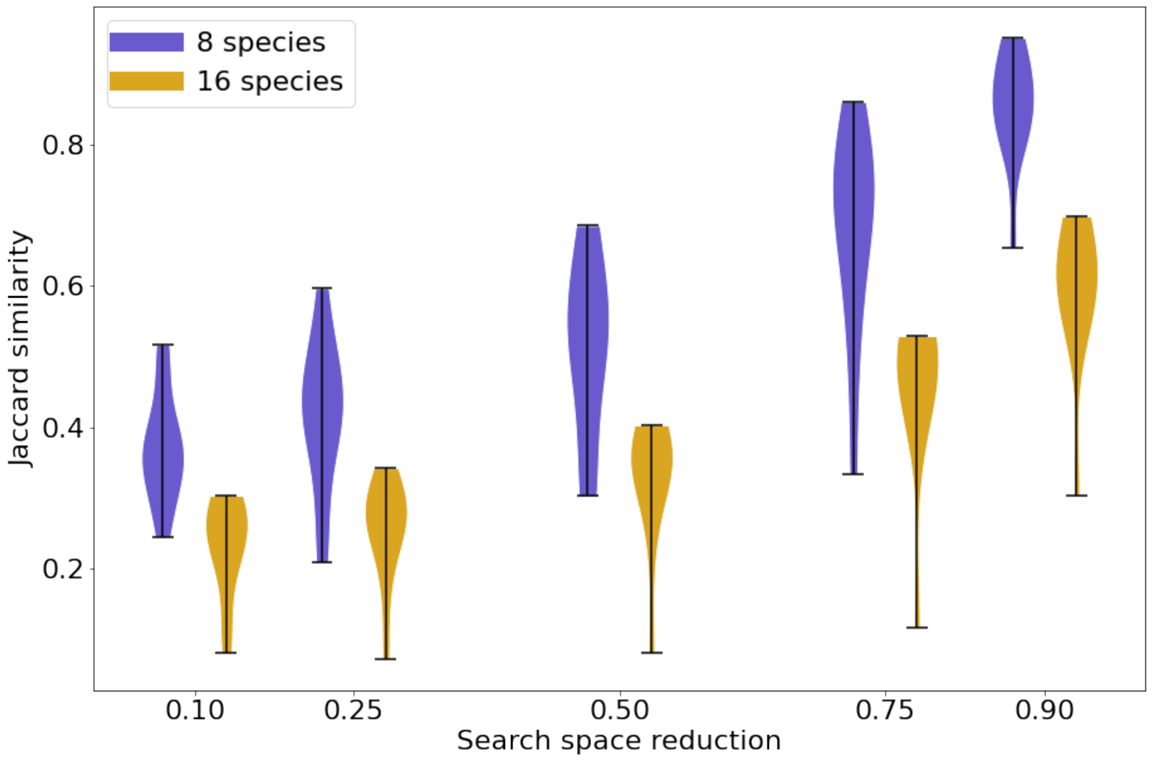


Figure 3.4: Jaccard similarity between synthetic reference and generated models with varying levels of prior knowledge. The violin plots show the distribution of Jaccard similarities achieved by MCTS for synthetic models. The horizontal axis shows varying proportions of incorrect interactions randomly removed from the list of actions that MCTS can choose when generating models. Removal of incorrect edges simulates the effect of prior knowledge, for example using only interactions from a database of validated biochemical interactions. As expected, higher levels of prior knowledge lead to higher Jaccard similarities, as MCTS has a higher probability of choosing correct interactions from a smaller list.

example, BioGRID (version 4.4.2021) has 26k genes and 806k interactions, which corresponds to a 99.9% reduction from all possible interactions (Oughtred et al., 2021). Thus, these tests simulate a very difficult scenario, relying on much less prior knowledge than is available in biochemical interaction databases.

3.4 Discussion and future directions

Through this thesis, I have attempted to push forward three main points. First computational models are crucial for understanding biology, and Boolean models are particularly useful. Second, Boolean models are hard to build. Finally, automated methods, particularly Monte Carlo Tree Search, are capable of automatically generating diverse, accurate Boolean models.

In Chapter 1, I explored how complex, networked phenomena are both crucial to life but also impenetrably difficult to understand without the aid of a computational model. However, building an accurate, comprehensive model is no trivial task. As I show through empirical evidence in Chapter 2, finding a model with the right behavior is fraught with challenges, primarily due to the enormous space of possibilities and the discontinuous, non-linear behavior of Boolean models.

The difficulty of this problem strongly suggests that automated methods could more easily generate sets of Boolean models whose behavior matches experimental observations. In this chapter I explored a novel method for synthesizing data-consistent Boolean models. However, I am not the first to make this conjecture and I briefly review existing methods for this task in Section 3.1. Many of these previous methods rely on traditional stochastic optimization techniques such as genetic optimization algorithms. Inspired by recent successes in game playing algorithms, I show that the Monte Carlo Tree Search (MCTS) algorithm (described in Section 3.2.1) is an effective method for constructing Boolean models. This required developing several novel techniques and adapting MCTS to Boolean model synthesis. Specifically, I describe a method for evaluating similarity between simulated and desired attractors and data in Section 3.2.3 and adaptations and extensions of MCTS in Sections 3.2.2 and 3.2.4.

Having described the framework I developed around MCTS, I then show that it works well and generalizes to a variety of tasks and input data. I demonstrate this through several experiments on synthetic, randomly generated models. However, even generating random data that has realistic characteristics is not totally straightforward, as I show in Section 3.3.1. Using the validated collection of random data, I ask two questions. First, can MCTS generate models with the correct behavior? Second, can MCTS recreate the correct behavior *using the same rules* as the original models? In Section 3.3.3, I show that MCTS is very effective at finding models that have the desired behavior. However, as I previously observed in Section 2.2, many models with the desired behavior had largely different mechanisms from the reference model. This led me to check in Section 3.3.4 whether MCTS generates any models with the correct behavior using the same interactions as the reference model. My experiments highlighted the importance of utilizing prior knowledge to bias and restrict the search space to generate models with the correct mechanisms.

To conclude, automated methods are a practical requirement for finding accurate Boolean models, given the immense difficulty of the problem. I develop a novel approach to automatically constructing Boolean models that utilizes Monte Carlo Tree Search and I show that it is effective at generating Boolean models that have the correct behavior and structure.

For future work, an obvious first step is to apply MCTS to experimental data to generate models of a biological system. Applying MCTS to a well-known model system could validate whether the algorithm can automatically generate the same rules and interactions as a model that was crafted by hand. In that regard, Albert and Othmer's model of the Drosophila segment polarity network is an appealing test bed. Their model of this Drosophila development gene circuit is multi-cellular with complex constraints, comprising gene expression and cross-membrane protein interactions. This would demonstrate MCTS's ability in a complicated, realistic context. Further, their model was hand-crafted based on extensive literature review. If MCTS can generate models with the same rules as Albert and Othmer's model, then this shows the power of automated model synthesis to discover the same interactions proposed by skilled model builders. In addition, if MCTS can discover new interactions not proposed by Albert and Othmer, this could shed light on novel regulatory mechanisms in this well-studied system.

However, the true test of an automated model building system would be to generate models of a system that does not have a well known reference. In this regard, the gene regulatory networks underlying cancer are attractive potential targets. This is especially true for cancer types that do not have common genetic drivers in known pathways. Small cell lung cancer (SCLC) is a particularly deadly form of cancer that doesn't have well defined genetic subtypes. While virtually all SCLC has mutations in two common cancer driver genes, some subsets of SCLC cells seem be more aggressive, especially in tumors that emerge after treatment. These subtypes of SCLC don't have specific genetic mutations that distinguish them (George et al., 2015). Instead, their phenotypes are driven by differing profiles of gene expression (Gay et al., 2021). This naturally leads to the question of how these expression profiles emerge. Another way of framing this question is "what is the structure of the gene regulatory network that has these expression profiles as steady states?". An automated model synthesis tool could provide insight into this question by generating models with attractors that are consistent with subtype expression profiles. As I described in Chapter 1, models of this system would allow researchers to ask a variety of questions. Which perturbations change the system's behavior? Can aggressive subtypes be targeted specifically? Is the gene regulatory network "rewired" compared to normal lung tissue?

More generally, I hope to make these capabilities more widely available by releasing this software under an open source license. Wider use of automated methods for model synthesis could have several effects.

First, as noted in my discussion of cross-talk in Section 1.1.1, biomedical literature is biased towards study of a few well-known genes and pathways. While this has yielded deep knowledge and several successful therapies targeting these genes and pathways, there still remains a long tail of genes, proteins, pathways, and interactions that remain profoundly mysterious. Automated methods do not necessarily have biases towards well-known mechanisms; as I showed in Section 3.3.4, my method can generate models with consistent behavior but a wide variety of mechanisms. This suggests that more widespread use of automated model synthesis methods could spur research into novel or under-studied interactions or mechanisms if they are suggested by unbiased algorithms. Additionally, automated methods could allow a democratization and proliferation of quantitative, computational modeling. As I showed in Chapter 2, building models is fraught with difficulty and, as such, largely remains the province of specialists. I hope that the method I describe here can become an accessible tool for researchers who are not skilled in model building, allowing them to quickly generate models based on their data. Further, I hope skilled modelers can use automated methods to suggest novel mechanisms or quickly expand a manually constructed model. Finally, if automated methods fulfil the lofty promise of democratizing computational modeling, this would generate a profusion of models from researchers studying different biological systems and contexts. Careful examination, curation, and integration of these models could ultimately produce comprehensive ”whole cell” models that more accurately represent broad phenotypic responses involving interactions between all cellular systems. Such an undertaking would be enormously complex and remains far in the future. However, the complexity of cellular systems and the difficulty of modeling them suggests that automated model construction methods, such as the one I developed, would be a key part of fully describing cellular behavior.

References

Aghamiri, S. S. and Delaplace, F. (2020). TaBooN – Boolean Network Synthesis Based on Tabu Search. *arXiv:2009.03587 [cs, q-bio]*. arXiv: 2009.03587.

Albeck, J. G., Burke, J. M., Aldridge, B. B., Zhang, M., Lauffenburger, D. A., and Sorger, P. K. (2008). Quantitative Analysis of Pathways Controlling Extrinsic Apoptosis in Single Cells. *Molecular Cell*, 30(1):11–25.

Albert, R. and Othmer, H. G. (2003). The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster. *Journal of Theoretical Biology*, 223(1):1–18.

Bracco, E. and Pergolizzi, B. (2014). Ras proteins signaling in the early metazoan Dictyostelium discoideum. *Methods in Molecular Biology (Clifton, N.J.)*, 1120:407–420.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (2017). *Classification And Regression Trees*. Routledge, New York.

Chen, C., Cui, J., Lu, H., Wang, R., Zhang, S., and Shen, P. (2007). Modeling of the Role of a Bax-Activation Switch in the Mitochondrial Apoptosis Decision. *Biophysical Journal*, 92(12):4304–4315.

Chevalier, S., Froidevaux, C., Paulevé, L., and Zinovyev, A. (2019). Synthesis of Boolean Networks from Biological Dynamical Constraints using Answer-Set Programming. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 34–41. ISSN: 2375-0197.

Coulom, R. (2006). Efficient selectivity and backup operators in monte-carlo tree search. In *In: Proceedings Computers and Games 2006*. Springer-Verlag.

Crespo, I., Krishna, A., Le Béchec, A., and del Sol, A. (2013). Predicting missing expression values in gene regulatory networks using a discrete logic modeling optimization guided by network stable states. *Nucleic Acids Research*, 41(1):e8–e8.

D'Alessandro, L., Samaga, R., Maiwald, T., Rho, S., Bonefas, S., Raue, A., Iwamoto, N., Kienast, A., Waldow, K., Meyer, R., Schilling, M., Timmer, J., Klamt, S., and Klingmüller, U. (2015). Disentangling the Complexity of HGF Signaling by Combining Qualitative and Quantitative Modeling. *PLoS Comput. Biol.*

Dechter, R. (2003). Consistency-enforcing and constraint propagation. In Dechter, R., editor, *Constraint Processing*, The Morgan Kaufmann Series in Artificial Intelligence, pages 51–83. Morgan Kaufmann, San Francisco.

Di Rienzo, C., Piazza, V., Gratton, E., Beltram, F., and Cardarelli, F. (2014). Probing short-range protein Brownian motion in the cytoplasm of living cells. *Nature Communications*, 5(1):5891. Number: 1 Publisher: Nature Publishing Group.

Dorier, J., Crespo, I., Niknejad, A., Liechti, R., Ebeling, M., and Xenarios, I. (2016). Boolean regulatory network reconstruction using literature based knowledge with a genetic algorithm optimization method. *BMC Bioinformatics*, 17.

Dougherty, M. K., Müller, J., Ritt, D. A., Zhou, M., Zhou, X. Z., Copeland, T. D., Conrads, T. P., Veenstra, T. D., Lu, K. P., and Morrison, D. K. (2005). Regulation of Raf-1 by Direct Feedback Phosphorylation. *Molecular Cell*, 17(2):215–224.

Elowitz, M. B., Levine, A. J., Siggia, E. D., and Swain, P. S. (2002). Stochastic Gene Expression in a Single Cell. *Science*, 297(5584):1183–1186. Publisher: American Association for the Advancement of Science.

Fisher, J., Köksal, A. S., Piterman, N., and Woodhouse, S. (2015). Synthesising Executable Gene Regulatory Networks from Single-Cell Gene Expression Data. In Kroening, D. and Păsăreanu, C. S., editors, *Computer Aided Verification*, Lecture Notes in Computer Science, pages 544–560. Springer International Publishing.

Gay, C., Stewart, C., Park, E., Diao, L., Groves, S. M., Heeke, S., Nabet, B., Fujimoto, J., Solis, L., Lu, W., Xi, Y., Cardnell, R., Wang, Q., Fabbri, G., Cargill, K., Vokes, N., Ramkumar, K., Zhang, B., Corte, C. D. D., Robson, P., Swisher, S., Roth, J., Glisson, B., Shames, D., Wistuba, I., Wang, J., Quaranta, V., Minna, J., Heymach, J., and Byers, L. (2021). Patterns of transcription factor programs and immune pathway activation define four major subtypes of SCLC with distinct therapeutic vulnerabilities. *Cancer cell*.

Gelly, S. and Silver, D. (2011). Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence*, 175(11):1856–1875.

George, J., Lim, J. S., Jang, S. J., Cun, Y., Ozretić, L., Kong, G., Leenders, F., Lu, X., Fernández-Cuesta, L., Bosco, G., Müller, C., Dahmen, I., Jahchan, N. S., Park, K.-S., Yang, D., Karnezis, A. N., Vaka, D., Torres, A., Wang, M. S., Korbel, J. O., Menon, R., Chun, S.-M., Kim, D., Wilkerson, M., Hayes, N., Engelmann, D., Pützer, B., Bos, M., Michels, S., Vlasic, I., Seidel, D., Pinther, B., Schaub, P., Becker, C., Altmüller, J., Yokota, J., Kohno, T., Iwakawa, R., Tsuta, K., Noguchi, M., Muley, T., Hoffmann, H., Schnabel, P. A., Petersen, I., Chen, Y., Soltermann, A., Tischler, V., Choi, C.-m., Kim, Y.-H., Massion, P. P., Zou, Y., Jovanovic, D., Kontic, M., Wright, G. M., Russell, P. A., Solomon, B., Koch, I., Lindner, M., Muscarella, L. A., la Torre, A., Field, J. K., Jakopovic, M., Knezevic, J., Castaños-Vélez, E., Roz, L., Pastorino, U., Brustugun, O.-T., Lund-Iversen, M., Thunnissen, E., Köhler, J., Schuler, M., Botling, J., Sandelin, M., Sanchez-Cespedes, M., Salvesen, H. B., Achter, V., Lang, U., Bogus, M., Schneider, P. M., Zander, T., Ansén, S., Hallek, M., Wolf, J., Vingron, M., Yatabe, Y., Travis, W. D., Nürnberg, P., Reinhardt, C., Perner, S., Heukamp, L., Büttner, R., Haas, S. A., Brambilla, E., Peifer, M., Sage, J., and Thomas, R. K. (2015). Comprehensive genomic profiles of small cell lung cancer. *Nature*, 524(7563):47–53.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Haynes, W. A., Tomczak, A., and Khatri, P. (2018). Gene annotation bias impedes biomedical research. *Scientific Reports*, 8:1362.

Holland, J. H. (1975). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press, Oxford, England. Pages: viii, 183.

Howells, C. C., Baumann, W. T., Samuels, D. C., and Finkielstein, C. V. (2011). The Bcl-2-associated death promoter (BAD) lowers the threshold at which the Bcl-2-interacting domain death agonist (BID) triggers mitochondria disintegration. *Journal of Theoretical Biology*, 271(1):114–123.

Kamerlin, S. C. L., Sharma, P. K., Prasad, R. B., and Warshel, A. (2013). Why nature really chose phosphate. *Quarterly Reviews of Biophysics*, 46(1):1–132. Publisher: Cambridge University Press.

Kauffman, S. (1969a). Homeostasis and Differentiation in Random Genetic Control Networks. *Nature*, 224(5215):177–178. Number: 5215 Publisher: Nature Publishing Group.

Kauffman, S. A. (1969b). Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of Theoretical Biology*, 22(3):437–467.

Kocsis, L. and Szepesvári, C. (2006). Bandit Based Monte-Carlo Planning. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M., editors, *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, pages 282–293, Berlin, Heidelberg. Springer.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520.

Lim, C. Y., Wang, H., Woodhouse, S., Piterman, N., Wernisch, L., Fisher, J., and Göttgens, B. (2016). BTR: training asynchronous Boolean models using single-cell expression data. *BMC Bioinformatics*, 17(1):355.

Lopez, C. F., Muhlich, J. L., Bachman, J. A., and Sorger, P. K. (2013). Programming biological models in Python using PySB. *Molecular Systems Biology*, 9(1):646.

Lotka, A. J. (1910). Contribution to the Theory of Periodic Reactions. *The Journal of Physical Chemistry*, 14(3):271–274. Publisher: American Chemical Society.

Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

McAdams, H. H. and Arkin, A. (1999). It's a noisy business! Genetic regulation at the nanomolar scale. *Trends in Genetics*, 15(2):65–69.

Mendoza, M. C., Er, E. E., and Blenis, J. (2011). The Ras-ERK and PI3K-mTOR pathways: cross-talk and compensation. *Trends in Biochemical Sciences*, 36(6):320–328.

Moelling, K., Schad, K., Bosse, M., Zimmermann, S., and Schweneker, M. (2002). Regulation of Raf-Akt Cross-talk. *The Journal of Biological Chemistry*, 277(34):31099–31106.

Moler, C. B. (2004). *Numerical computing with MATLAB*. SIAM.

Orchard, S., Kerrien, S., Abbani, S., Aranda, B., Bhate, J., Bidwell, S., Bridge, A., Briganti, L., Brinkman, F. S. L., Cesareni, G., Chatr-aryamontri, A., Chautard, E., Chen, C., Dumousseau, M., Goll, J., Hancock, R. E. W., Hannick, L. I., Jurisica, I., Khadake, J., Lynn, D. J., Mahadevan, U., Perfetto, L., Raghunath, A., Ricard-Blum, S., Roechert, B., Salwinski, L., Stümpflen, V., Tyers, M., Uetz, P., Xenarios, I., and Hermjakob, H. (2012). Protein interaction data curation: the International Molecular Exchange (IMEx) consortium. *Nature Methods*, 9(4):345–350. Number: 4 Publisher: Nature Publishing Group.

Ortega, O. O., Wilson, B. A., Pino, J. C., Irvin, M. W., Ildefonso, G. V., Garbett, S. P., and Lopez, C. F. (2021). Probability-based mechanisms in biological networks with parameter uncertainty. preprint, Systems Biology.

Oughtred, R., Rust, J., Chang, C., Breitkreutz, B.-J., Stark, C., Willems, A., Boucher, L., Leung, G., Kolas, N., Zhang, F., Dolma, S., Coulombe-Huntington, J., Chatr-Aryamontri, A., Dolinski, K., and Tyers, M. (2021). The BioGRID database: A comprehensive biomedical resource of curated protein, genetic, and chemical interactions. *Protein Science: A Publication of the Protein Society*, 30(1):187–200.

Palsson, B. (2006). *Systems Biology : Properties of Reconstructed Networks*. Cambridge University Press.

Petzold, L. (1983). Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations. *SIAM Journal on Scientific and Statistical Computing*, 4(1):136–148. Publisher: Society for Industrial and Applied Mathematics.

Prugger, M., Einkemmer, L., Beik, S. P., Harris, L. A., and Lopez, C. F. (2020). Unsupervised logic-based mechanism inference for network-driven biological processes. *bioRxiv*, page 2020.12.15.422874. Publisher: Cold Spring Harbor Laboratory Section: New Results.

Raser, J. M. and O'Shea, E. K. (2004). Control of Stochasticity in Eukaryotic Gene Expression. *Science*, 304(5678):1811–1814. Publisher: American Association for the Advancement of Science.

Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144, New York, NY, USA. Association for Computing Machinery.

Rodriguez, A., Crespo, I., Androsova, G., and Sol, A. d. (2015). Discrete Logic Modelling Optimization to Contextualize Prior Knowledge Networks Using PRUNET. *PLOS ONE*, 10(6):e0127216. Publisher: Public Library of Science.

Rosin, C. D. (2011). Nested Rollout Policy Adaptation for Monte Carlo Tree Search. *IJCAI*.

Ross, L. N. (2021). Causal Concepts in Biology: How Pathways Differ from Mechanisms and Why It Matters. *The British Journal for the Philosophy of Science*, 72(1):131–158. Publisher: The University of Chicago Press.

Saez-Rodriguez, J., Alexopoulos, L. G., Epperlein, J., Samaga, R., Lauffenburger, D. A., Klamt, S., and Sorger, P. K. (2009). Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Molecular Systems Biology*, 5:331.

Sam, S. A., Teel, J., Tegge, A. N., Bharadwaj, A., and Murali, T. (2017). XTalkDB: a database of signaling pathway crosstalk. *Nucleic Acids Research*, 45(D1):D432–D439.

Shockley, E. M., Vrugt, J. A., and Lopez, C. F. (2018). PyDREAM: high-dimensional parameter inference for biological models in python. *Bioinformatics*, 34(4):695–697.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144. Publisher: American Association for the Advancement of Science.

Städter, P., Schälte, Y., Schmiester, L., Hasenauer, J., and Stapor, P. L. (2021). Benchmarking of numerical integration methods for ODE models of biological systems. *Scientific Reports*, 11:2696.

Suderman, R., Bachman, J. A., Smith, A., Sorger, P. K., and Deeds, E. J. (2017). Fundamental trade-offs between information flow in single cells and cellular populations. *Proceedings of the National Academy of Sciences*, 114(22):5755–5760.

Szklarczyk, D., Gable, A. L., Nastou, K. C., Lyon, D., Kirsch, R., Pyysalo, S., Doncheva, N. T., Legeay, M., Fang, T., Bork, P., Jensen, L. J., and von Mering, C. (2021). The STRING database in 2021: customizable protein-protein networks, and functional characterization of user-uploaded gene/measurement sets. *Nucleic Acids Research*, 49(D1):D605–D612.

Wachter, S., Mittelstadt, B., and Russell, C. (2017). Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR. SSRN Scholarly Paper ID 3063289, Social Science Research Network, Rochester, NY.

Waters, A. M. and Der, C. J. (2018). KRAS: The Critical Driver and Therapeutic Target for Pancreatic Cancer. *Cold Spring Harbor Perspectives in Medicine*, 8(9):a031435.

Westheimer, F. H. (1987). Why Nature Chose Phosphates. *Science*, 235(4793):1173–1178. Publisher: American Association for the Advancement of Science.

Wrede, F. and Hellander, A. (2019). Smart computational exploration of stochastic gene regulatory network models using human-in-the-loop semi-supervised learning. *Bioinformatics*, 35(24):5199–5206.

Yordanov, B., Dunn, S.-J., Kugler, H., Smith, A., Martello, G., and Emmott, S. (2016). A method to identify and analyze biological programs through automated reasoning. *npj Systems Biology and Applications*, 2(1):1–16. Number: 1 Publisher: Nature Publishing Group.

Zoller, B., Little, S. C., and Gregor, T. (2018). Diverse Spatial Expression Patterns Emerge from Unified Kinetics of Transcriptional Bursting. *Cell*, 175(3):835–847.e25.