LETTERS IN BIOMATHEMATICS
An International Journal

Intercollegiate Biomathematics Alliance

REVIEW ARTICLE

OPEN ACCESS

Algebraic models, inverse problems, and pseudomonomials from biology

Matthew Macauley[a], Raina Robeva[b]

[a]School of Mathematical and Statistical Sciences, Clemson University, Clemson, SC 29634; [b]Department of Mathematics, Randolph-Macon College, Ashland, VA 23005

ABSTRACT

In contrast to the extensive use of linear algebra in biology, problems involving nonlinear polynomials—the field of *algebraic biology*—are more conspicuous. In this article, we highlight two biological problems where similar algebraic structures arise in very different contexts. Specifically, we will look at algebraic models of molecular networks, and combinatorial codes of place fields in neuroscience. Both of these topics involve inverse problems where the data is encoded with *pseudomonomials*, simple algebraic objects that do not seem to have been studied much on their own, but have gained considerable attention lately from their visibility in mathematical biology. Though this article is algebraic in nature, it is written for the general mathematician with a minimal algebra background assumed. It provides two distinctive features that have not yet appeared in the literature: (i) a survey of Boolean and logical modeling from a computational algebra perspective, and (ii) a unification of this topic with algebraic neuroscience by highlighting the role of pseudomonomials in both fields.

ARTICLE HISTORY
Received April 22, 2020
Accepted June 11, 2020

KEYWORDS
Algebraic biology, Boolean model, inverse problem, neural code, pseudomonomial ideal, primary decomposition

1 Introduction

Linear algebra, the study of linear polynomials and their solutions, is a fundamental pillar of mathematical biology. The analysis of nonlinear polynomials and their solutions is considerably more complex, and involves fields such as algebraic geometry and computational algebra. Though these themes are not as ubiquitous in biology as linear algebra is, they nevertheless do arise in a number of biological problems. *Algebraic Biology* is the subfield that encompasses these problems, and the new mathematics that they spawn. A recent survey paper titled *The case for algebraic biology: from research to education* that is forthcoming in a special educational issue of the *Bulletin for Mathematical Biology* (Macauley and Youngs, 2020) gives a brief overview of four diverse problems in the field and then discusses their role within mathematical biology in the classroom and curriculum. Only the bare minimum is given for each of these four problems—just enough to see how polynomials arise, and not much else. Specifically, these applications are biochemical reaction networks, Boolean models of gene regulatory networks, algebraic statistics and evolutionary models in phylogenetics, and place fields in neuroscience.

In this paper, we will revisit two of these topics in algebraic biology where similar mathematical structures arise, despite the fact that the biological problems that they model—gene regulatory networks, and place fields in neuroscience, are very different. In particular, the concept of a pseudomonomial arises in both of these topics. Though pseudomonomials are easy to define, they are objects that do not seem to have been studied much outside of algebraic biology. Our hope is that this paper will serve as a quick crash course on the topics while unifying the common mathematical structures. We will survey these topics to understand which algebraic structures arise, what tools are used in the analysis, and to see several examples of new mathematics that has arisen. This paper provides two distinctive features that have not yet appeared in the literature: (i) a survey of algebraic modeling from a computational algebra perspective, and (ii) a unification of this topic with algebraic neuroscience by highlighting the significance of pseudomonomials in both of these fields. The aim for the level of this paper is similar to a typical "*What Is...*" article in the AMS Notices. It should be generally accessible, but individuals in related fields will gain even more from it. Admittedly, it will be more accessible for an algebraist than for a classical mathematical biologist. However, the presentation of the algebra is only at the undergraduate level, and even then, understanding all of the details is not necessary. We encourage the interested classical

CONTACT Robeva R. RainaRobeva@rmc.edu

mathematical biologist to recruit an algebra-minded colleague to work through this paper together, because both should benefit from such an endeavor, as the topic is multifaceted and interdisciplinary in nature.

The remainder of this paper is structured as follows. In Section 2, we will give a brief history of mathematical biology, from classical topics such as population models to modern methods involving discrete mathematics and algebraic geometry. Section 3 will be a survey of algebraic models of molecular networks. Specifically, after describing the framework and how polynomials arise, we will give a tour of some of the basic questions and methods, and how computational algebra arises in the analysis. We will show how Gröbner bases can be used to find the steady-states, and then how the primary decomposition of square-free monomial ideals can reconstruct biological networks from experimental data. If one wants to determine which interactions are activations and which are inhibitions, then one can use pseudomonomial ideals instead. We will show how the model space—the set of functions that fit a set of data, has an affine ideal structure, much like the solution space to an inhomogeneous system of equations $\mathbf{Ax} = \mathbf{b}$, or an inhomogeneous ODE $\mathbf{x}' = \mathbf{Ax} + \mathbf{b}$. We will show how to compute these solutions using tools from computational algebra—or at least, have a software package compute them. We will discuss the model selection problem of finding the "best fit" function from the large model space, and how this can be done with a Gröbner normal form. Finally, we will discuss how certain functions tend to arise in biological networks. One such example is canalizing functions, and an important class of these can be constructed with pseudomonomials.

In Section 4, we will turn our attention to a completely different biological problem—that of reconstructing neural place fields from experimental data of firing patterns of neurons. We will summarize an algebraic approach that is even a newer topic than algebraic models, as the first paper was only published in 2013. However, similar mathematical structures arise, including finite fields, pseudomonomial ideals, and their primary decompositions. It is not at all our expectation that the readers of this article know any of the algebraic buzzwords that we threw out there in the last two paragraphs. Moreover, we will not even define most of these, though we may give an informal description of what they are. Instead, our goal is to write this article in a way that it will reach readers who do not know these terms.

2 A brief history of mathematical biology

Before we get into any technical details, we will first give some historical context so we can understand how the field of mathematical biology got to where it is today. Traces of mathematical biology can be found in work by Fibonacci (c. 1170–1250) and Daniel Bernoulli (1700–1782). Exponential growth as a population model was published by Thomas Malthus at the turn of the 19th century (Malthus, 1803). Several decades later, the logistic model was published by Pierre-Francois Verhulst (Verhulst, 1838). The field gained real traction in the early 20th century with the development of the Lotka-Volterra predator-prey equations (Lotka, 1925). For the next few decades, mathematical biology almost exclusively consisted of continuous-space models such as differential equations (continuous-time) and difference equations (discrete-time). These type of models are still a major thrust of the field, and will likely always be, for good reason.

Discrete mathematical structures started to creep into biology around the 1960s, from several different angles. The discovery of the lactose operon in *E. coli* by Jacob, Lwoff, and Monod, led to a 1965 Nobel Prize (Jacob et al., 1960). This was the first instance of what now is know as a gene regulatory network, and it put forth the concept of a biological network. Eventually, this would lead to the emergence of the field of systems biology (Alon, 2019). Also around this time, Noam Chomsky was developing his theory of universal grammars, which led him to be known as the "father of modern linguistics" (Chomsky, 2014). These ideas were picked up by biologists and applied to genomics, since Watson, Crick, and Franklin had only recently discovered that the genetic code is basically a language over the alphabet $\{A, C, G, T\}$. This was the precursor to what would become the fields of bioinformatics and computational biology. Also in the late 1960s and early 1970s, several different scientists an ocean apart proposed studying molecular networks using Boolean logic. In North America, theoretical biologist Stuart Kauffman pioneered the concept of a Boolean network to model gene networks (Kauffman, 1969a,b). In Europe, geneticist René Thomas independently introduced the discrete-state *logical modeling* framework (Thomas, 1973). One of the main differences between these two frameworks is that Kauffman applied the local functions synchronously, whereas Thomas considered asynchronous updates. A fantastic historical perspective on these developments can be found in a recent special issue of the *Journal of Theoretical Biology*, dedicated to the late Thomas (Thieffry and Kaufman, 2019). Also around this time, Princeton mathematician John Conway popularized the cellular automaton he called the *Game of Life* (Gardner, 1970). Though this last one was at best, only very loosely connected to actual biology, it helped give support to the idea of discrete-state and agent-based models of biological systems.

The 1990s and early 2000s saw a revolution within computational biology due to the advent of technology such as high-throughput sequencers and DNA microarrays, as well as a massive increase in computer processing power. This led to the emergence of relatively new fields such as computational biology (Waterman, 2014), systems biology (Alon, 2019) and bioinformatics (Mount, 2004). New computationally intensive problems that were pipe dreams just a few years prior, suddenly became thrust within the realm of reality. Examples include sequence alignment, whole genome sequencing, protein and RNA folding, and phylogenetic tree construction. Many of these problems are transdisciplinary in nature, and have attracted researchers from

not just the mathematical and biological sciences, but also computer science, operations research, engineering, physics, neuroscience, and medicine. Many of these problems are more inherently "discrete" than "continuous" in nature due to the presence of networks and genomic data. Despite all having strong mathematical underpinnings, they are often not considered mathematical biology, simply because of their broad reach well beyond just mathematics. Of course, this summary would be incomplete without the mention of biostatistics. Biological research is inherently messy and contains large troves of data. Nearly all of the aforementioned research problems have major statistical components, and data analysis is such an important part of biological research that biostatistics has become its own giant field.

The goal of this paper is two-fold: 1) to show how algebraic methods can be used to provide answers to biological problems that may be difficult or impossible to tackle otherwise and 2) to demonstrate how the process of looking for solutions to those biology-generated questions may lead to new mathematics. As the field of algebraic biology has expanded significantly in the last decade, it would be impossible to present, in a single paper, all types of scientific questions and all types of mathematical tools that tie biology with new algebraic and combinatorial approaches. Instead, we have opted for discussing how questions about inferring unknown biological structures from experimental data arise naturally in two very different settings: molecular networks, and neuroscience. These are examples of *reverse engineering*, and are also sometimes called *inverse problems*. The two inverse problems that we highlight can both be approached by using pseudomonomial ideals—mathematical structures that are of significant interest in their own right and whose full mathematical theory is still waiting to be developed—thus showing how through many of its questions, modern biology now drives mathematical discovery.

3 Algebraic models of molecular networks

3.1 Background and motivation

Cellular processes are governed by a complex system of interactions between proteins, small molecules, RNA, and DNA. These dynamical interactions produce tightly coordinated cellular behaviors such as transcription, translation, energy transport, and internal and external signaling. Correct regulation of these processes ensures that vital biological functions are executed properly and that, in the long run, a cell reaches its physiologically appropriate state or cell fate. Disruption of this biological order may impact signaling pathways and lead to malfunctioning and disease—e.g., mitochondrial diseases, lysosomal storage diseases, diabetes, and cancer (Fernández-Tajes et al., 2019; Laubenbacher et al., 2009; Marí and Fernández-Checa, 2007). Cellular systems may contain hundreds or thousands of components that are coordinated by multiple feedback loops and interact in a nonlinear fashion. Predicting their dynamic behavior, as well as determining possible causes for certain macro effects is extremely challenging, thus making experimental approaches to answering those questions virtually impossible.

The discipline that studies how interactions among a system's components leads to emergent behaviors in living organisms is *systems biology*—an interdisciplinary field that uses mathematical models and computation to study mechanisms by which coordinated action of biological units leads to specific functional performance. Theoretical approaches provide methods that allow us to study cellular systems analytically or through simulation approaches, and determine components that play critical roles in achieving desired behaviors. Others may form "backup channels" to ensure robustness in case of certain malfunctions, yet others may appear to be there as artifacts from evolutionary changes and/or serve functions unknown to us. Classifying those components and understanding their respective roles is critically important in understanding how a system behaves. Furthermore, understanding the main causes for maintaining or disrupting cellular behaviors highlights directions toward designing intervention strategies to control the system and drive it away from unwanted (diseased) states.

Different approaches are being used to model biomolecular systems, including ordinary differential equations (ODE), algebraic models (Boolean, finite-state), and hybrid models. ODE models require detailed knowledge of the kinetic processes that determine the system's dynamics, as well as understanding how they impact one another over time. Such information is usually unavailable for large cellular systems, and this is a real drawback to using ODEs to model cellular processes. Algebraic models, on the other hand, allow only a finite number of states for each of the system's components, and the dynamic behavior of the system is determined by simple (e.g., Boolean) rules describing how these components interact. The term "algebraic" is used because these rules can be represented with polynomials, and analyzed using techniques from computational algebra. Unlike quantitative ODE models, algebraic models are inherently qualitative and can capture emergent system properties without requiring detailed knowledge of the kinetics of cellular interactions. It has been shown that many fundamental dynamic behaviors of a cellular system may be derived solely from the network topology and the rules of interaction between its components (Albert and Othmer, 2003). Algebraic models have also been shown to capture more complex cellular behaviors and describe biological responses depending on environmental stimuli—e.g., excitation, adaptation, multistability (Robeva and Hodge, 2013, Chapter 4). Finally, algebraic models can often handle gaps in knowledge when available information about the network construction is incomplete (Li et al., 2006; Saez-Rodriguez et al., 2009). These properties of algebraic models make them an appealing alternative to ODEs for modeling cellular systems.

For algebraic models, we use a directed graph to represent the network of interactions—the nodes represent the various

biomolecular species and the edges denote the interactions between them. The temporal dynamics is governed by the update rules described earlier. Once a model is constructed and validated, its attractors and steady states are mapped to important known biological states or behaviors. Identifying those states, as well as their basins of attraction (that is, the initial and transient states that converge to each attractor), is of high interest—they can provide insights regarding the long-term dynamics and stabilization of a system (representing, e.g., different cell fates or phenotypes) under changing environmental signals or in response to perturbations.

3.2 Algebraic modeling framework

Consider a molecular network such as a gene regulatory network with n nodes, where x_i is the state of node i. This typically describes a quantity such as a gene expression level or concentration of a protein or enzyme. It can also denote the presence or absence of a feature, such as whether or not a portion of a DNA strand is looped, which is one way to block transcription. In an algebraic model, concentrations and gene expression levels are discretized, such as high vs. low (basal), or high, medium, and low. These can be represented as Boolean or ternary states, e.g., $\mathbb{F}_2 = \{0,1\}$ or $\mathbb{F}_3 = \{0,1,2\}$. Of course, there are other possibilities, but these are the most common. In some models, certain nodes will be Boolean and other might be ternary. However, from a computational algebra perspective, things are mathematically easier and we lose no generality if we assume that the state of every node is from the same set, a field $\mathbb{F} = \mathbb{F}_p = \{0,1,\ldots,p-1\}$. If we want or need a particular quantity to have additional states, then we can achieve this by introducing a new variable. For example, if we want to express three levels of lactose concentration in a Boolean model, we can use two variables, letting L represent high levels and L_m represent (at least) medium levels. As such, $(L, L_m) = (0,0)$ would mean low concentration, $(L, L_m) = (0,1)$ would describe medium concentration, and $(L, L_m) = (1,1)$ would mean high concentration. The fourth possibility of $(L, L_m) = (1,0)$ is meaningless and can be disregarded, just like how certain states in an ODE model are ignored, like when population is negative. Finally, the choice of $\mathbb{F}_p$ for a prime p, rather than say, 4 or 6, is motivated because most of the computational algebraic techniques require $\mathbb{F}_n$ to be a prime field. This also loses no generality, because even if there are naturally 6 states, we can just include that into a larger set, such as $\mathbb{F}_7$.

Time is also usually discretized, as $t = 0,1,2,\ldots$, but there can be variants of this as well. If we let $x_i(t)$ be the state of node i at time t, then the state $x_i(t+1)$ at the following time-step is some function of the states of the nodes. That is, it is described by a *local function* $f_i : \mathbb{F}^n \to \mathbb{F}$. As such, models like this are sometimes called *local models* (Robeva and Macauley, 2018, Chapter 4). Each local function can be expressed as a polynomial in the ring $R = \mathbb{F}[x_1,\ldots,x_n]$. Such a representation is not unique because $x^p = x$ in $\mathbb{F}_p$. However, we can force uniqueness by declaring that $x_i^p = x_i$ for each $i = 1,\ldots,n$. Equivalently, this means that each function can be expressed uniquely as an element of the quotient ring

$$Q = \mathbb{F}[x_1,\ldots,x_n]/\langle x_1^p - x_1, \ldots, x_n^p - x_n\rangle. \tag{1}$$

A simple counting argument shows that there are p^n monomials in Q, and thus p^{p^n} elements, i.e., functions $\mathbb{F}^n \to \mathbb{F}$. Throughout this section, we will refer to R and Q as the aforementioned ring of polynomials, and quotient ring of functions, respectively. Representing local biological functions as polynomials opens the door to using the rich toolbox of computational algebra for the analysis of the models and related algorithms. In light of this viewpoint, these models are sometimes called *algebraic models*. We will primarily use this term in this paper because of the focus on algebraic biology, though the special case when $p = 2$ is often called a *Boolean model*. A number of other terms exist in the literature, often describing specialized cases. For example, Boolean models are often called *Boolean networks*, especially if the study of them is detached from actual models. A large swath of researchers, especially in Europe, use the term *logical model* (Abou-Jaoudé et al., 2016). Other term, such as *automata networks* (Goles and Martínez, 2013), or *generalized cellular automata*, can also be found in the literature.

Every algebraic model has a directed graph called a *wiring diagram* that describes which functions depend on which variables. We say that x_i affects f_j positively if $f_j(\mathbf{x}) < f_j(\mathbf{x}')$ for some input vectors $\mathbf{x} = (x_1,\ldots,x_n)$ and $\mathbf{x}' = (x_1',\ldots,x_n')$ that are identical except in their i^{th} coordinates, where $x_i < x_i'$. Similarly, x_i affects f_j negatively if $f_j(\mathbf{x}) > f_j(\mathbf{x}')$ under the same assumptions. In the special case of Boolean logic, x_i affects f_j positively if x_i (but not $\neg x_i$) appears in the function f_j, and negatively if $\neg x_i$ appears but x_i does not. A variable can affect a function both positively and negatively, like in the logical XOR function

$$f(x+y) = (x \wedge \neg y) \vee (\neg x \wedge y) = x + y \pmod 2.$$

Notice that a general Boolean expression can be written as a polynomial over $\mathbb{F}_2$ by replacing $x \wedge y$ with xy, $x \vee y$ with $x + y + xy$, and $\neg x$ with $1 + x$.

The (unsigned) wiring diagram of an algebraic model has vertex set $[n] = \{1,\ldots,n\}$ with an edge $i \longrightarrow j$ if the function f_j depends on x_i. Often, it is desirable for these edges to indicate whether the relationship is positive or negative. In a *signed wiring diagram*, if x_i affects f_j positively, we depict this as $i \longrightarrow j$, and if the relationship is negative, then we draw $i \longdashv j$. If the interaction is positive and negative, such as the XOR function, then both types edges can be included in the wiring diagram. However, many biological interactions are strictly positive or negative in a fixed coordinate, as they represent activation, and

inhibition or repression, respectively. Such a function is said to be *unate*. Indeed, functions that arise as local functions in biological models are generally much tamer than arbitrary n-variable functions. This means that the dynamics of actual Boolean models are more ordered, or less chaotic, than randomly constructed Boolean networks (Gershenson, 2004).

At this point, we have not actually defined what we mean by the dynamics of an algebraic model, and that is intentional because the framework is quite general. Moreover, the model is just the local functions themselves—hence the name *local model*. The computation and analysis of the dynamics is really just a type of model validation. The functions can be updated synchronously, asynchronously (in multiple ways), or in some hybrid scheme such as a block-sequential update. Sometimes stochasticity is desired, and the local functions can define an object such as a continuous-time Markov chain (Stoll et al., 2012) or probabilistic Boolean network (Shmulevich and Dougherty, 2010). Though biological processes happen at different time-scales, this can be incorporated into a model by adding additional variables, in a similar spirit to how extra variables can be used to capture additional states of the nodes (Hinkelmann and Laubenbacher, 2011).

It is undoubtedly more natural to use some sort of asynchronous update because cells do not have a central clock under which everything evolves simultaneously. However, certain salient features of a model, such as the steady states, are independent of update scheme. Moreover, since biological networks need to be robust due to all sorts of variability such as changes in temperature, pH, mutations, timing, etc., one can make the case that if a proposed algebraic model can be validated despite using an artificial synchronous update, then that provides compelling evidence for its soundness. Also, the synchronous update is simpler and more amenable to certain algebraic methods that we will survey in this article.

At time t, the (local) state of node i is $x_i(t)$, and so the *global state* of the system is a vector

$$\mathbf{x}(t) = \big(x_1(t), \dots, x_n(t)\big) \in \mathbb{F}^n.$$

Updating all vertices synchronously defines a *dynamical system map*

$$f\colon \mathbb{F}^n \longrightarrow \mathbb{F}^n, \qquad \mathbf{x}(t+1) = f(\mathbf{x}(t)) = \big(f_1(\mathbf{x}(t)), \dots, f_n(\mathbf{x}(t))\big).$$

Sometimes we write this map as $f = (f_1, \dots, f_n)$, and it is not hard to see that every dynamical system map $f\colon \mathbb{F}^n \to \mathbb{F}^n$ can be decomposed into local functions in this manner.

3.3 Steady-state analysis

One of the fundamental features of any model is to analyze its long-term behavior, especially any steady-states, or fixed points. For an algebraic model, regardless of whether it is updated synchronously or asynchronously, finding the fixed points amounts to solving $\mathbf{x}(t+1) = \mathbf{x}(t)$. This can be described by the following system of n equations:

$$\big\{f_i(\mathbf{x}) = x_i \mid i = 1, \dots, n\big\}. \tag{2}$$

Systems of polynomial equations such as these can be analyzed using techniques from computational algebraic geometry. For example, if we define the ideal

$$I = \big\langle f_i - x_i \mid i = 1, \dots, n\big\rangle = \big\{(f_i - x_i)h \mid h \in R\big\},$$

then the solution to Eq. (2) is simply the *algebraic variety* $V(I)$ of I. A computer algebra package such as Macaulay2 (Grayson and Stillman, 2020) or Singular (Decker et al., 2020) can find this by computing a Gröbner basis, which returns a simpler system with the same set of solutions, much like how row-reducing a system of linear equations returns a simpler (upper triangular) system with the same solutions.

To see an example of this, consider the following toy Boolean model from (Robeva and Hodge, 2013, Chapter 2) of the lactose operon in *E. coli*, the first gene regulatory network that was discovered:

$$\begin{aligned} x_1(t+1) &= x_3(t) \\ x_2(t+1) &= x_1(t) \\ x_3(t+1) &= (x_2(t) \wedge L_m) \vee L \vee (x_3(t) \wedge \neg x_2(t)). \end{aligned} \tag{3}$$

Here, the variables x_1, x_2, and x_3 represent concentrations of *lac* mRNA, the β-galactosidase enzyme, and allolactose (an isomer of lactose and the operon's inducer), respectively. There are two parameters, L and L_m, which represent extracellular lactose, and together allow us to describe high, medium and low levels, by $(L, L_m) = (1,1)$, $(0,1)$, and $(0,0)$, respectively. These are taken to be constants because their levels change much more slowly than the concentration of the gene products inside of the cell. They should be thought of like initial conditions of a differential equation—we must analyze and validate the model for all three possibilities.

To find the fixed points of the model in Eq. (3), we work in the quotient ring of Boolean functions

$$\mathbb{F}_2[x_1, x_2, x_3, L, L_m]/\langle x_1^2 - x_1, x_2^2 - x_2, x_3^2 - x_3, L^2 - L, L_m^2 - L_m\rangle.$$

The solution to the system of equations $\{f_i - x_i = 0 \mid i = 1, 2, 3\}$ is the variety of the ideal

$$I = \langle f_i - x_i \mid i = 1, 2, 3\rangle = \langle x_3 + x_1,\ x_1 + x_2,\ (1 + x_3 + x_2x_3 + x_2L_m)L + x_2(x_3 + L_m)\rangle.$$

Notice that $-x_i = x_i$ over $\mathbb{F}_2$. A computer algebra package easily computes a Gröbner basis to be

$$\mathcal{G} = \{x_1 + x_3,\ x_2 + x_3,\ (1 + L_m)x_3 + (1 + L_m)L,\ (1 + x_3)L\}.$$

If we call the four polynomials above $g_1, \ldots, g_4$, then the system $\{g_i = 0 \mid i = 1, \ldots, 4\}$ has the same set of solutions as the original system $\{f_i - x_i = 0 \mid i = 1, 2, 3\}$, but it is much easier to solve. For example, consider the initial condition where there are only medium levels of extracellular lactose, i.e., $(L, L_m) = (0, 1)$. The Gröbner basis then becomes $\mathcal{G} = \{x_1 + x_3, x_2 + x_3\}$, which over $\mathbb{F}_2$ means that $x_1 = x_3$ and $x_2 = x_3$. Thus, there are two steady-state solutions,

$$\mathbf{x} = (x_1, x_2, x_3) = (0, 0, 0), \qquad \text{and} \qquad \mathbf{x} = (x_1, x_2, x_3) = (1, 1, 1),$$

corresponding to the operon being off and on, respectively. This is an example of the well-known phenomenon of *bistability*—the system having two stable steady-states when there are medium concentration levels of the inducer. The fact that the *lac* operon is known to exhibit bistability is promising evidence for the soundness of our model (Ozbudak et al., 2004). Of course, this alone does not validate such a model, but it provides evidence. If we repeat the above process for the other two initial conditions, $(L, L_m) = (0, 0)$ and $(1, 1)$, we will see that in each case, there is a single steady-state solution, which matches what is expected biologically. Overall, these are necessary, but not sufficient steps for model validation.

Actual Boolean models are typically larger than just three nodes. The toy example above was specifically chosen from an exercise in (Robeva and Hodge, 2013, Chapter 2), because it is small yet exhibits bistability. A published Boolean model for the *lac* operon (Veliz-Cuba and Stigler, 2011) has 10 nodes and 3 parameters, which is still small enough that a computational algebra software package can easily handle the necessary computations. A published model of the related arabinose (*ara*) operon in *E. coli* has 9 nodes and 4 parameters (Jenkins and Macauley, 2017). However, a number of published algebraic models are simply too large. For example, the model in (Albert and Othmer, 2003) of the segment polarity genes in the fruit fly has 60 nodes, and solving polynomial systems in 60 variables is computationally prohibitive.

To overcome this limitation, it may be necessary to first consider methods for reducing the size of a system in a way that preserves its attractors. Many such algorithms—see e.g., (Naldi et al., 2011; Saadatpour et al., 2011; Veliz-Cuba et al., 2014; Veliz-Cuba and Laubenbacher, 2012; Veliz-Cuba and Geiser, 2016; Zañudo and Albert, 2013), have brought about powerful network reduction approaches. However, each of them comes with some limitations and no method appears to be optimal for all scenarios. For instance, the reduction method introduced in (Veliz-Cuba et al., 2014) is efficient for large networks (up to 1000 nodes) for determining only the fixed points of a system, while Zañudo and Albert (2013) determined all attractors for networks with up to 200 nodes. Their method for performing attractor analysis is based on the expanded network of a system—one that integrates the update rules with the network topology. This framework has also been used successfully for network control, e.g., for cell fate reprogramming in T-cell leukemia (Zañudo and Albert, 2015) and for target control of algebraic models for the epithelial-to-mesenchymal transition (EMT) network and the PI3K mutant ER+ network (Yang et al., 2018). We will discuss control of molecular networks in Section 3.9.

3.4 Network inference, min-sets, and monomial ideals

The process of building and analyzing an algebraic model, such as the example of the *lac* operon we just saw, is an instance of *forward engineering*. It assumes that the network of biological interactions for the model is already known. In these cases, such models are constructed with evidence from the literature, and they are built and tested to reproduce known experimental results—see e.g., (Zhang et al., 2008). However, such methods require much time and effort and rely on the assumption that information on most biomolecular interactions for the system in question are already available. Such knowledge could indeed be obtained experimentally, but the laborious approaches it relies on do not scale up when we need to uncover interactions within networks of hundreds or thousands of nodes.

The opposite problem is known as *reverse-engineering*, which is a type of *inverse problem*. Specifically, given data about the dynamics of the model, what can we infer about the model itself? For example, it may be of interest to infer the wiring diagram, i.e., determine how the genes, proteins, and enzymes interact. This is sometimes called the *network inference* or *network reconstruction* problem. There are many algorithms that approach this using a variety of frameworks and techniques. For example, (Friedman et al., 2000; Hartemink et al., 2000) explored using Bayesian networks, whereas in (Yeung et al., 2002) the authors

approximated a system of linear differential equations near a steady-state and considered the singular value decomposition of the matrix. The method that we will review in this article uses computational algebraic geometry. It was originally published in (Jarrah et al., 2007), and a survey can be found in (Robeva and Macauley, 2018, Chapter 6). A harder problem than inferring the network is to reverse-engineer the actual functions in the model. At times, we prefer the term "reverse engineer" to "inverse problem" because the former is a verb, and the latter is quite general—there are inverse problems for which it would be a stretch to say that something is being reverse engineered.

Let's start with the easier problem of reverse engineering the wiring diagram of an algebraic model. We can ask for just the interactions (directed edges), or we can ask for the signed wiring diagram, i.e., which interactions are inducers (positive) and which are inhibitors (negative). This might require extra assumptions on the functions, because not every function is necessarily unate (either positive or negative in any given coordinate). However, as previously stated, many biological interactions are simple enough that they can be modeled by such functions, and so this assumption is often well-founded.

The available data for a reverse-engineering problem is likely experimentally determined, and might consist of a time-series or several snapshots of gene expressions levels. Naturally, these will be floating point values, but they can be discretized to Boolean, ternary, or more generally, $\mathbb{F} = \{0, 1, \ldots, p-1\}$. Alternatively, one can disregard the numeric values and just look at when the levels are increasing vs. decreasing. There are a number of other challenging problems, such as how to handle noisy data, or how large of a data set these algorithms will scale to, but we will not discuss these in this article. Indeed, the inherent messiness of biological data presents a never-ending problem because algorithms are never as simple as black boxes that spit out a golden answer.

The question of how to discretize the data is a challenge in itself and, once again, there is no one right answer for every setting. For now, we will ignore this problem, refer the interested reader to (Dimitrova, 2010), and assume henceforth that we already have discretized data, and focus on what to do next. We should note however, that none of what we will do depends on how finely the data is discretized, i.e., it works just as well for any $\mathbb{F}_p$. Additionally, we will only consider one local function at a time, because reverse-engineering the network can be done node-by-node. For our purposes, we will define a set of *data* to be m snapshots of input-output pairs of the form

$$\mathcal{D} = \big\{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m)\big\}, \tag{4}$$

where $\mathbf{s}_i = (s_{i1}, \ldots, s_{in}) \in \mathbb{F}^n$ is a vector of gene expression levels or concentrations, and $t_i \in \mathbb{F}$. The input vectors must be unique, but the output values need not be. We say that a local function $f : \mathbb{F}^n \to \mathbb{F}$ *fits the data* if $f(\mathbf{s}_i) = t_i$ for all $i = 1, \ldots, m$. The (local) *model space*, denoted $\mathrm{Mod}(\mathcal{D})$, is the set of functions that fit the data. It makes no difference whether we define this in the ring R of polynomials or the quotient ring Q of functions, where each $x_i^p = x_i$. Formally, the model space is

$$\mathrm{Mod}(\mathcal{D}) = \big\{f \in Q \mid f(\mathbf{s}_i) = t_i \text{ for all } i = 1, \ldots, n\big\}.$$

If we want to reverse-engineer the wiring diagram, then we are interested in what are the possible *supports* of a function $f \in \mathrm{Mod}(\mathcal{D})$, i.e., the sets of variables on which f depends. In terms of the wiring diagram of a graph, we are asking what are the minimal sets of incoming edges to that particular node.

It is easiest to express the support of a function f, denoted $\mathrm{supp}(f)$, as a subset α of $[n] = \{1, \ldots, n\}$, which we will write with a string. For example, we can write $\alpha = 125$ for $\{1, 2, 5\} \subseteq [6]$, which represents the subset $\{x_1, x_2, x_5\}$ of variables. We will denote the complement with a bar, i.e., $\overline{\alpha} = 346$ represents $[6] \setminus \alpha = \{3, 4, 6\}$. Every $\alpha \subseteq [n]$ canonically defines a *square-free monomial* $\mathbf{x}^\alpha$. For example, if $\alpha = 125$, then $\mathbf{x}^\alpha = x_1 x_2 x_5$. Now, we can define a few terms about what sets of variables are sufficient, necessary, or unnecessary to fit a set $\mathcal{D}$ of data.

Definition 1. Let $\mathcal{D}$ be a set of data. A subset $\alpha \subseteq [n]$ is *feasible* with respect to $\mathcal{D}$ if there is some $f \in \mathrm{Mod}(\mathcal{D})$ such that $\mathrm{supp}(f) \subseteq \alpha$. It is *disposable* if there is some $f \in \mathrm{Mod}(\mathcal{D})$ for which $\mathrm{supp}(f) \subseteq \overline{\alpha}$.

In plain English, a set α is feasible if there is some function that fits the data using only those variables. It is disposable if none of those variables are needed to construct a function that fits the data; otherwise it is non-disposable. Note that a subset $\alpha \subseteq [n]$ is (non-)disposable if and only if its complement $\overline{\alpha}$ is (in-)feasible. To reverse engineer the wiring diagram at a particular node, it suffices to find all possible supports of functions in the model space. We will do this by asking which feasible subsets α are *minimal* with respect to subset inclusion, and we will call these *min-sets*.

Definition 2. Let $\mathcal{D}$ be a set of data. A subset $\alpha \subseteq [n]$ is a *min-set* of $\mathcal{D}$ if it is a minimal feasible set, or equivalently, if its complement $\overline{\alpha}$ is a maximal disposable set.

Throughout, we'll assume that $\mathcal{D}$ is a fixed set of data. It is easy to see that disposable sets are closed under taking subsets and intersections, and feasible sets, as well as non-disposable sets, are closed under taking supersets and unions. In other words, the disposable and the infeasible sets each form an *abstract simplicial complex* (a collection $\Delta_{\mathcal{D}} \subseteq 2^X$ of subsets of $X = [n]$, called *faces*, that are closed under subsets) and the feasible sets and the non-disposable sets define monomial ideals. We will not give

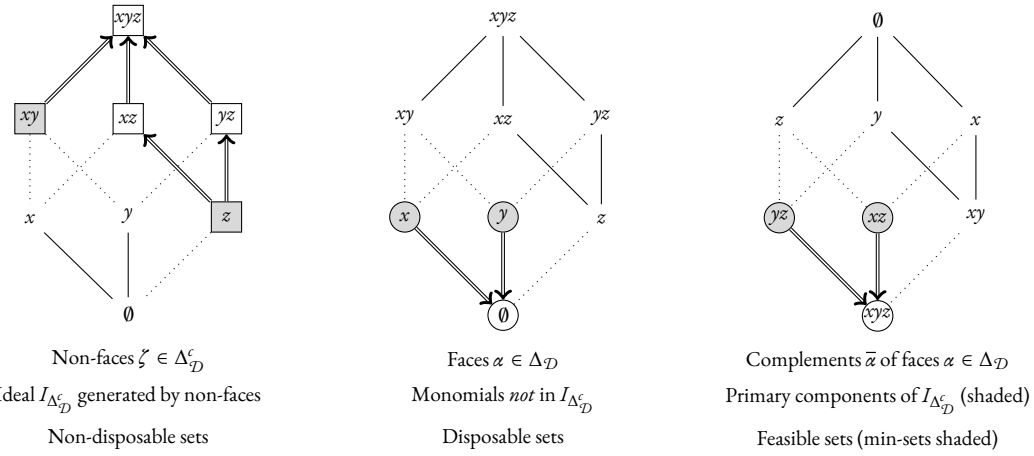

Figure 1: A visualization of the non-disposable, disposable, and feasible sets and the corresponding combinatorial and algebraic structures. This example arises from a data set $\mathcal{D}$ that we will see in Eq. (5).

all of the details here, but the beautiful Stanley-Reisner theory from combinatorial commutative algebra guarantees a bijection between simplicial complexes and square-free monomial ideals (Francisco et al., 2014). The technical details of these are not crucial for this article, and so we will omit them. However, there are two important take-aways. The first key point is that this ideal is generated by monomials $\mathbf{x}^{\alpha}$ from the non-faces, i.e., $\alpha \in \Delta_{\mathcal{D}}^c := 2^X \setminus \Delta_{\mathcal{D}}$. In our language, this means that *the ideal is generated by the non-disposable sets*. The second key point is that upon taking the *primary decomposition* of this ideal—a process analogous to factoring an integer into prime powers, the individual *primary components* correspond to the complements of the maximal faces. In our language, this means that the *primary components describe the complements of the maximal disposable sets, i.e., the min-sets*. The relationship between these algebraic and combinatorial concepts is subtle and tricky, especially because there are two types of set complements involved—the complement $\Delta_{\mathcal{D}}^c = 2^X \setminus \Delta_{\mathcal{D}}$ of a simplicial complex, and the complement $\overline{\alpha} = X \setminus \alpha$ of a face $\alpha \in \Delta_{\mathcal{D}}$. Figure 1 provides a helpful visualization of these concepts and how they are related. Soon, we will see an actual data set $\mathcal{D}$ that corresponds to this example.

Returning to the first key point described above, one way to find a non-disposable set of $\mathcal{D}$ is to look at pairs $\mathbf{s}_i, \mathbf{s}_j$ of input data vectors that have *different* output values, $t_i \neq t_j$. The set of coordinates in which they differ must be non-disposable, because the change in output must have been caused by some of these variables. We can encode this set with a square-free monomial, by multiplying all of these variables together, which we denote as

$$m(\mathbf{s}_i, \mathbf{s}_j) = \prod_{s_{ik} \neq s_{jk}} x_k.$$

The ideal of non-disposable sets is the ideal generated by all such pairs of input vectors in $\mathcal{D}$ that have different output values.

Definition 3. Let $\mathcal{D}$ be a set of data. The *ideal of non-disposable sets* is

$$I_{\Delta_{\mathcal{D}}^c} = \big\langle m(\mathbf{s}_i, \mathbf{s}_j) \mid t_i < t_j \big\rangle.$$

The assumption that $t_i < t_j$ is not necessary, but it is convenient because $m(\mathbf{s}_i, \mathbf{s}_j) = m(\mathbf{s}_j, \mathbf{s}_i)$. However, it will be needed later for the signed analogue of this. As an aside, the reason for the complicated and non-standard notation of this ideal is because it is "*the ideal generated by the non-faces of the simplicial complex* $\Delta_{\mathcal{D}}$." The second key point above characterizes the min-sets in terms of this ideal.

Theorem 4 (Jarrah et al., 2007)**.** *The generators of the primary components of the ideal of non-disposable sets* $I_{\Delta_{\mathcal{D}}^c}$ *are the min-sets of* $\mathcal{D}$.

Let us pause to do a quick example illustrating min-sets. A Boolean function $f : \mathbb{F}_2^3 \to \mathbb{F}_2$ is characterized by the output values of all $2^3 = 8$ input vectors, often called its *truth table*. Suppose we just know three of these, as described by the following data set, where we are writing a vector $\mathbf{x} = (x, y, z)$ as a string xyz:

$$\mathcal{D} = \big\{(\mathbf{s}_1, t_1),\ (\mathbf{s}_2, t_2),\ (\mathbf{s}_3, t_3)\big\} = \big\{(111, 0),\ (000, 0),\ (110, 1)\big\}. \tag{5}$$

There are five unknown output values, and since there are two possibilities for each of them, exactly $2^5 = 32$ functions fit this data. In other words, the model space has size $|\operatorname{Mod}(\mathcal{D})| = 32$.

Two of the three pairs of input vectors in Eq. (5) have different output values, so the ideal of non-disposable sets is

$$I_{\Delta_{\mathcal{D}}^c} = \big\langle m(\mathbf{s}_1, \mathbf{s}_3),\ m(\mathbf{s}_2, \mathbf{s}_3) \big\rangle = \langle z, xy \rangle. \tag{6}$$

The lattice on the left in Figure 1 highlights the square-free monomials in $I_{\Delta_{\mathcal{D}}^c}$, which are the non-faces of $\Delta_{\mathcal{D}}$. The faces are shown in the middle lattice of Figure 1. The primary decomposition is a way to express the ideal as an intersection of primary ideals, called its *primary components*. Though in general this is difficult to compute, for square-free monomials ideals, these are just the ideals generated by the elements in the complements of the maximal faces of the corresponding simplicial complex. For small examples, this can be computed by hand. Otherwise, it is an easy task for any computational algebra software package. Continuing the example from Eq. (6), the corresponding simplicial complex has two maximal faces, $\{x\}$ and $\{y\}$. Thus, the elements in the complements of these generate the primary components, $\langle y, z \rangle$ and $\langle x, z \rangle$. This is shown in the rightmost lattice of Figure 1, and it means that

$$I_{\Delta_{\mathcal{D}}^c} = \langle z, xy \rangle = \langle x, z \rangle \cap \langle y, z \rangle.$$

By Theorem 4, the min-sets of $\mathcal{D}$ are $\{x, z\}$ and $\{y, z\}$. In other words, if a function f fits the data given in Eq. (5), then its support must contain z, and also either x or y.

Not surprisingly, the mathematical tools highlighted in this section have not been widely used for research in biology. The primary reason is likely because the intersection of people or teams who work on problems in system biology and who have a knowledge of computational algebra is quite small. However, there are several notable examples that highlight the potential of these methods. Allen et al. (2006) used this algebraic framework to reverse-engineer a signaling network in the yeast *S. cerevisiae* by collecting time-series data from stress induced by copper. In (Stigler and Chamberlin, 2012), the authors used algebraic methods to infer gene networks involved in the muscle and skin cell fates in the model organism and roundworm *C. elegans*.

3.5 Signed min-sets and pseudomonomial ideals

The monomial $m(\mathbf{s}_i, \mathbf{s}_j)$ is constructed by taking the product of the variables x_k corresponding to the coordinates in which $\mathbf{s}_i$ and $\mathbf{s}_j$ differ. Suppose that additionally, we want to encode *how* they differ, analogous to a discrete version of a partial derivative. One solution to this was first published in (Veliz-Cuba, 2012), and has also appeared in a recent survey in (Robeva and Macauley, 2018, Chapter 6).

Since $\mathbf{s}_i$ and $\mathbf{s}_j$ have different outputs, we can assume without loss of generality that $t_i < t_j$, and ask whether this increase in output value corresponds to an increase or decrease in the k^{th} coordinate of the input vector. We can encode an increase $s_{ik} < s_{jk}$ with $x_k - 1$ and a decrease $s_{ik} > s_{jk}$ with $x_k + 1$, and then multiply all of these terms together. Of course, the resulting polynomial is not a monomial, but it somewhat resembles one. Specifically, if we change variables by replacing $x_k - 1$ with y_k and $x_k + 1$ with z_k—an operation called *polarization* (Güntürkün et al., 2019), then we get a monomial but in a different polynomial ring. Though the original polynomial is not a monomial, we call it a *pseudomonomial*. To do this, we need to be working in a field where $1 \neq -1$, and so we will choose $\mathbb{F}_3 = \{0, 1, -1\}$ rather than $\mathbb{F}_2 = \{0, 1\}$. Formally, for each $t_i < t_j$, we define

$$p(\mathbf{s}_i, \mathbf{s}_j) = \prod_{s_{ik} \neq s_{jk}} \big(x_k - \operatorname{sign}(s_{jk} - s_{ik})\big) \in \mathbb{F}_3[x_1, \ldots, x_n].$$

The polynomials $m(\mathbf{s}_i, \mathbf{s}_j)$ and $p(\mathbf{s}_i, \mathbf{s}_j)$ have the same supports, which are non-disposable sets of the data $\mathcal{D}$. We can extend the definition of concepts like (non-)disposable sets and min-sets to *signed (non-)disposable sets* and *signed min-sets* by keeping track of whether $s_{ik} < s_{jk}$ or $s_{ik} > s_{jk}$. We will discuss notation for this later, when we return to our running example. This time, instead of a monomial ideal generated by the non-disposable sets, we can create a *pseudomonomial ideal* generated by the signed non-disposable sets.

Definition 5. Let $\mathcal{D}$ be a set of data. The *ideal of signed non-disposable sets* is

$$J_{\Delta_{\mathcal{D}}^c} = \big\langle p(\mathbf{s}_i, \mathbf{s}_j) \mid t_i < t_j \big\rangle.$$

The result of Theorem 4, that the generators of the primary components of the ideal of non-disposable sets are the min-sets, follows immediately from the rich theory of Stanley-Reisner. There is no such established theory for pseudomonomials, but the same result still holds for signed min-sets (Veliz-Cuba, 2012). There is one caveat though: we need to make the blanket assumption to only consider unate functions.

Theorem 6 (Veliz-Cuba, 2012)**.** *The primary components of the ideal of signed non-disposable sets $J_{\Delta_{\mathcal{D}}^c}$ are the signed min-sets of $\mathcal{D}$.*

Let's return to our example data set $\mathcal{D} = \{(111, 0),\ (000, 0),\ (110, 1)\}$ from the previous section, and Eq. (4). Since we are keeping track of signs, it is crucial that these input-output pairs be ordered by non-decreasing output values. The first and third input vectors differ in only the last coordinate. Since $0 = t_1 < t_3 = 1$ despite $1 = s_{13} > s_{33} = 0$, the z-variable has a potential negative influence, and so $p(\mathbf{s}_1, \mathbf{s}_3) = z + 1$. The second and third input vectors differ in their first and second coordinates, with $0 = s_{21} < s_{31} = 1$ and $0 = s_{22} < s_{32} = 1$, and thus $p(\mathbf{s}_2, \mathbf{s}_3) = (x - 1)(y - 1)$. Therefore, the ideal of signed non-disposable sets of $\mathcal{D}$, and its primary decomposition, found with the aid of a computational algebra software package, is

$$J_{\mathcal{D}} = \big\langle p(\mathbf{s}_1, \mathbf{s}_3), p(\mathbf{s}_2, \mathbf{s}_3) \big\rangle = \langle z + 1,\ (x - 1)(y - 1) \rangle = \langle x - 1, z + 1 \rangle \cap \langle y - 1, z + 1 \rangle.$$

It is often convenient to write a variable x_i that has a negative influence as $\overline{x_i}$. Using this notation, the *signed min-sets*, i.e., the signed support of a unate function that fits the data must contain $\{x, \overline{z}\}$ or $\{y, \overline{z}\}$. In other words, any unate function in $\mathrm{Mod}(\mathcal{D})$ must depend positively on x or y, but also negatively on z.

3.6 Model space structure

Thus far, we have seen how to determine which sets of variables functions in the model space can depend on. An even more bold question is to determine the actual functions themselves, where each one is assumed to be a function of the form $f_i \colon \mathbb{F}^n \to \mathbb{F}$. This problem gets difficult quickly, because if $\mathbb{F} = \mathbb{F}_p$, there are p^{p^n} such functions on n nodes. However, often we might know extra information that simplifies the problem. For example, gene networks are often sparse (Andrecut and Kauffman, 2008), and so individual functions might only have one or two inputs. For example, if gene A in a Boolean model is activated by enzyme B but repressed by protein C, then there are only two possibilities for the function: it is either an AND gate or an OR gate, i.e.,

$$f_A(B, C) = B \wedge \neg C = B(1 + C), \qquad \text{or} \qquad f_A(B, C) = B \vee \neg C = 1 + C + BC.$$

This is a very natural question to ask biologically. It is also a reverse-engineering problem, but this time we are reverse-engineering the functions instead of the wiring diagram. In this section, we will characterize the model space algebraically. This was first published in (Laubenbacher and Stigler, 2004), and (Robeva and Hodge, 2013, Chapter 3) contains a nice survey. It should be noted that though this does not necessarily answer the *model selection* question of what the correct or best function that fits the data is, understanding the structure of the model space is a necessary first step.

Recall that the model space $\mathrm{Mod}(\mathcal{D})$ consists of all local functions that fit the data $\mathcal{D}$, i.e., all $f \colon \mathbb{F}^n \to \mathbb{F}$ such that $f(\mathbf{s}_i) = t_i$ for each i. It is not hard to show that this set has a nice algebraic structure as

$$\mathrm{Mod}(\mathcal{D}) = f + I = \{f + h \mid h \in I\}, \tag{7}$$

where f is any particular function that fits the data, and I is the set of functions (an ideal) that vanish on the data. That is, $I = \{h \mid h(\mathbf{s}_i) = 0, \text{ for all } i = 1, \ldots, n\}$. One can think of the model space as being analogous to the solution space of an inhomogeneous system of linear equations, $\mathbf{Ax} = \mathbf{b}$, which is $\mathbf{x} = \mathbf{x}_n + \mathbf{x}_p$, where $\mathbf{x}_n$ is the nullspace and $\mathbf{x}_p$ is any particular solution. Alternatively, it is analogous to the general solution of an inhomogeneous systems of ODEs $\mathbf{x}' = \mathbf{Ax} + \mathbf{b}$, which is $\mathbf{x}(t) = \mathbf{x}_h(t) + \mathbf{x}_p(t)$, where $\mathbf{x}_h(t)$ is the solution of the related homogeneous equation and $\mathbf{x}_p(t)$ is any particular solution. In both of these examples, the solution spaces are affine vector spaces. The model space in Eq. (7) has a similar structure and can be thought of as an "affine ideal".

The reverse-engineering problems we have seen thus far have all just focused on data for a single function, consisting of input *vectors* and output *values*. The model space consists of the local functions that fit the data, and the min-sets are the minimal supports from this space. However, once discretized, experimental gene expression data will typically consist of input-output pairs of *vectors* in $\mathbb{F}^n$, i.e.,

$$\mathcal{D} = \big\{(\mathbf{s}_1, \mathbf{t}_1), \ldots, (\mathbf{s}_m, \mathbf{t}_m)\big\}, \qquad \mathbf{s}_i, \mathbf{t}_i \in \mathbb{F}^n. \tag{8}$$

Reverse-engineering the full wiring diagram requires performing the aforementioned min-set algorithm on all n nodes individually. In other words, the full data $\mathcal{D}$ in Eq. (8) can be broken up into n sets $\mathcal{D}_1, \ldots, \mathcal{D}_n$ of what we have been calling "data", where

$$\mathcal{D}_k = \big\{(\mathbf{s}_1, t_{k1}), \ldots, (\mathbf{s}_m, t_{km})\big\}. \tag{9}$$

Naturally, we can define the the model space of the set $\mathcal{D}$ from Eq. (8) as

$$\mathrm{Mod}(\mathcal{D}) = \big\{f \mid f(\mathbf{s}_i) = \mathbf{t}_i \mid \text{ for all } i = 1, \ldots, m\big\}.$$

This "full model space" can be found coordinate-by-coordinate; it is simply the set of n-tuples $(f_1, \ldots, f_n)$ that fit the data sets $\mathcal{D}_1, \ldots, \mathcal{D}_n$. That is,

$$\mathrm{Mod}(\mathcal{D}) = \mathrm{Mod}(\mathcal{D}_1) \times \cdots \times \mathrm{Mod}(\mathcal{D}_n) = (f_1 + I) \times \cdots \times (f_n + I) = (f_1 \ldots, f_n) + I^n,$$

where $\text{Mod}(\mathcal{D}_k) = f_k + I$, for any particular function f_k that fits the data $\mathcal{D}_k$. Since there are p^{p^n} functions $\mathbb{F}^n \to \mathbb{F}$, and $\mathcal{D}_k$ specifies the output for m of the p^n inputs, there are p choices for the remaining $p^n - m$ unknown output values. Therefore, the model spaces have exactly

$$\left|\text{Mod}(\mathcal{D}_k)\right| = \left|f_k + I\right| = \left|I\right| = p^{p^n - m}, \qquad \left|\text{Mod}(\mathcal{D})\right| = |I|^n = p^{n(p^n - m)}$$

functions.

3.7 Computation of model spaces

While it is nice to know that the algebraic structure of the model space of a data set $\mathcal{D}_k$ is $\text{Mod}(\mathcal{D}_k) = f_k + I$, that is really only helpful if one also knows how to construct the functions. Fortunately, there are standard algorithms for constructing a "particular function" f_k, as well as the vanishing ideal I (Laubenbacher and Stigler, 2004). After introducing these, we will turn our attention to *which* choice of particular function f_k should be made, which is the *model selection* problem. Throughout, $\mathcal{D}_k$ will be the data set from Eq. (9), but since we are fixing k, we will let $t_i = t_{ki}$ for convenience.

Let's start with the vanishing ideal I. If we let I_i be the set (an ideal) of polynomials that vanish on a fixed input vector $\mathbf{s}_i$, i.e.,

$$I_i = \left\{h \in Q \mid h(\mathbf{s}_i) = 0\right\},$$

then I is simply the intersection of the ideals $I_1, \ldots, I_m$. That is,

$$I = \left\{h \in R \mid h(\mathbf{s}_i) = 0, \text{ for all } i = 1, \ldots, n\right\} = \bigcap_{i=1}^{m} I_i.$$

Constructing I_i is quite easy: if $\mathbf{s}_i = (s_{i1}, \ldots, s_{in})$, then

$$I_i = \langle x_1 - s_{i1}, \ldots, x_n - s_{in}\rangle = \left\{(x_1 - s_{i1})h_1 + \cdots + (x_n - s_{in})h_n \mid h_1, \ldots, h_n \in Q\right\}.$$

For example, if $\mathbf{s}_i = (1, 0, 2)$,

$$I_i = \langle x - 1, y, z - 2\rangle = \left\{(x-1)h_1(\mathbf{x}) + yh_2(\mathbf{x}) + (z-2)h_3(\mathbf{x}) \mid h_i \in Q\right\},$$

and it is easy to see why all polynomials of this form vanish on the input $\mathbf{x} = (x, y, z) = (1, 0, 2)$. Computing a basis of the intersection of all I_i is an easy task for any computer algebra software package.

Next, let us turn to the question about how to construct a particular solution f_k that fits the data $\mathcal{D}_k$, i.e., a function $f_k \colon \mathbb{F}^n \to \mathbb{F}$ satisfying $f_k(\mathbf{s}_i) = t_i$ for each $i = 1, \ldots, m$. There are several methods to do this, and we will outline one that resembles Lagrange interpolation. Specifically, for each input vector $\mathbf{s}_i$, we construct an "indicator polynomial" $\chi_i(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_n)$, that evaluates to 1 on $\mathbf{s}_i$ but to 0 on all other $\mathbf{s}_j$. That is,

$$\chi_i(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} = \mathbf{s}_i \\ 0 & \mathbf{x} = \mathbf{s}_j,\ j \neq i. \end{cases} \tag{10}$$

There are many polynomials that satisfy the condition in Eq. (10), because we do not care about the value they take on inputs that are not one of the $\mathbf{s}_i$'s. One such construction for $\chi_i(\mathbf{x})$ in the Boolean case is done by the following steps.

1. Individually compare $\mathbf{s}_i$ to each $\mathbf{s}_j$, for $j \neq i$.
2. For each $\mathbf{s}_j \neq \mathbf{s}_i$, pick *any* coordinate $\ell = \ell_j$ in which they differ, and encode this with $x_\ell - s_{j\ell}$.
3. Multiply all $m - 1$ of these terms together to get $\chi_i(\mathbf{x})$.

First, let's observe why this works. If $j \neq i$, then $\chi_i(\mathbf{s}_j) = 0$ because plugging $\mathbf{s}_j$ into the term $x_\ell - s_{j\ell}$ yields zero. In contrast, $\chi_i(\mathbf{s}_i) \neq 0$ because for each $j \neq i$, plugging $\mathbf{s}_i$ into $x_\ell - s_{j\ell}$ is $s_{i\ell} - s_{j\ell}$, which is nonzero due to Step (2).

The construction described above is more involved in the non-Boolean case, because while the product of $m - 1$ non-zero terms over $\mathbb{F}_2$ is necessarily 1, this clearly fails for a more general prime field $\mathbb{F}_p$. Specifically, each term $(x_\ell - s_{j\ell})$ requires multiplication by $(s_{i\ell} - s_{j\ell})^{p-2}$. The technical details of why this works is due to the Chinese remainder theorem from number theory, which does not need to be discussed here.

Once we have the indicator polynomials at our disposal, it should be straightforward to see why the function

$$f(\mathbf{x}) = t_1\chi_j(\mathbf{x}) + t_2\chi_2(\mathbf{x}) + \cdots + t_m\chi_m(\mathbf{x}) \tag{11}$$

fits the data $\mathcal{D}_k = \{(\mathbf{s}_1, t_1), \ldots, (\mathbf{s}_m, t_m)\}$. Specifically, notice that if we plug $\mathbf{s}_i$ into $f(\mathbf{x})$, then because $\chi_j(\mathbf{s}_i) = 0$ for $j \neq i$, Eq. (11) becomes $f(\mathbf{s}_i) = t_i\chi_i(\mathbf{s}_i) = t_i$.

Let's revisit our example and compute the model space of $\mathcal{D} = \{(111, 0),\ (000, 0),\ (110, 1)\}$. Though we are working over $\mathbb{F}_2$, it is often psychologically helpful to use negative signs. To begin, the ideals I_i that vanish on $\mathbf{s}_i$ for $i = 1, 2, 3$ are

$$I_1 = \langle x-1, y-1, z-1\rangle, \qquad I_2 = \langle x, y, z\rangle, \qquad I_3 = \langle x-1, y-1, z\rangle.$$

The ideal that vanishes on all three of the inputs is thus $I = I_1 \cap I_2 \cap I_3$. A computer algebra software package easily finds that $\{x + y, (y + 1)z\}$ is a Gröbner basis for I, which means that

$$I = \langle x + y, (y + 1)z\rangle = \{(x + y)h_1(\mathbf{x}) + (y + 1)z\, h_2(\mathbf{x}) \mid h_1, h_2 \in Q\}.$$

Next, to find a particular solution, we need to construct indicator polynomials χ_1, χ_2, χ_3 that satisfy Eq. (10). We will do this applying the algorithm described above to $\mathbf{s}_1$, $\mathbf{s}_2$, and then $\mathbf{s}_3$. Notice that there is some choice along the way, whenever two input vectors differ in more than one coordinate.

1. $\mathbf{s}_1 = (1, 1, 1)$ differs from $\mathbf{s}_2$ in coordinate 1 (and 2, 3), and from $\mathbf{s}_3$ in coordinate 3, so
$$\chi_1(\mathbf{x}) = (x - s_{21})(z - s_{33}) = xz.$$
2. $\mathbf{s}_2 = (0, 0, 0)$ differs from $\mathbf{s}_1$ in coordinate 1 (and 2, 3), and from $\mathbf{s}_3$ in coordinate 2 (and 1), so
$$\chi_2(\mathbf{x}) = (x - s_{11})(y - s_{32}) = (x - 1)(y - 1) = (x + 1)(y + 1).$$
3. $\mathbf{s}_3 = (1, 1, 0)$ differs from $\mathbf{s}_1$ in coordinate 3, and from $\mathbf{s}_2$ in coordinate 2 (and 1), so
$$\chi_3(\mathbf{x}) = (z - s_{13})(y - s_{22}) = (z - 1)y = y(z + 1).$$

These indicator functions generate a particular solution

$$f(\mathbf{x}) = t_1\chi_1(\mathbf{x}) + t_2\chi_2(\mathbf{x}) + t_3\chi_3(\mathbf{x}) = 0 \cdot xz + 0 \cdot (x + 1)(y + 1) + 1 \cdot y(z + 1) = y(z + 1).$$

Thus, the model space of the data set $\mathcal{D}$ is

$$\mathrm{Mod}(\mathcal{D}) = f + I = y(z + 1) + \langle x + y, (y + 1)z\rangle, \tag{12}$$

and this contains exactly $2^{2^3-3} = 2^5 = 32$ functions.

3.8 Model selection

The goal of reverse-engineering a local function from a set of data is not to find just any function that fits the data, but to find the "best", "most correct", or "most likely" such function. If we express the model space as $\mathrm{Mod}(\mathcal{D}) = f + I$, then how do we choose such a function?

One answer would be to just pick f, but is that really the best answer biologically? Especially because there was a good deal of choice in constructing it. Another answer is to make an Occam's razor assumption and choose the simplest or "most reduced" function that fits the data. But what does this mean, and how would one find it?

As an analogy to motivate this, let us return to two mathematical equations whose solution space also has an affine structure. For example, the general solution to the ODE $y'' + y = 2$ is

$$y(t) = C_1 \cos t + C_2 \sin t + y_p(t),$$

where $y_p(t)$ is any particular solution. Though there are infinitely many choices for $y_p(t)$, such as $y_p(t) = 2\cos t - 5\sin t + 2$, the canonical choice is $y_p(t) = 2$. Loosely speaking, if we start with $y_p(t) = 2\cos t - 5\sin t + 2$, then we can reduce it by repeatedly taking away multiples of $\cos t$ and $\sin t$, which are solutions to the homogeneous equation, until we are left with a remainder of $y_p(t) = 2$.

For a second example, consider the system of linear equations

$$\begin{bmatrix} 2 & 1 & 3 \\ 3 & -5 & -2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}.$$

The set of solutions is $\mathbf{x} = \mathbf{x}_n + \mathbf{x}_p$, where $\mathbf{x}_n = C[1 \quad 1 \quad -1]^T$ is the nullspace, and $\mathbf{x}_p$ is any one particular solution. It is easy to check that $\mathbf{x}_p = [10 \quad 8 \quad -8]^T$ is one such solution, though that is not the canonical solution. If we reduce it by taking away multiples of $[1 \quad 1 \quad -1]^T$, we can get a simpler particular solution, $\mathbf{x}_p = [2 \quad 0 \quad 0]^T$.

Returning to the model selection problem, given $\mathrm{Mod}(\mathcal{D}) = f + I$, how do we simplify or fully reduce f? One answer comes from computational algebra, by computing the *Gröbner normal form* of f. Loosely speaking, given a Gröbner basis $\mathcal{G}$, the Gröbner normal form is the polynomial achieved by repeatedly taking away polynomials in $\mathcal{G}$ from f, analogous to the two examples above from ODEs and linear algebra. There are a number of technical details that we will not get into, but these can be found in (Cox et al., 2015). The easy answer for *how* to do this is to just use a computational algebra software package.

A more subtle issue is that the normal form depends on the choice of *monomial term order*, which specifies how to order monomials of multivariate polynomials so one can perform long division. For polynomnials of a single variable, there is only one choice: $1 \prec x \prec x^2 \prec \cdots$. However, in the multivariate case, it is not so clear what the relative order of monomials such as, e.g., x^3, xy^2, and y^5 should be. The short answer is that there are multiple choices that work, but they often lead to different Gröbner bases, and hence different Gröbner normal forms. Though there are uncountably many monomial orders, there are only finitely many (reduced) Gröbner bases of any ideal, and these can be characterized geometrically by the polyhedral *Gröbner fan* of the ideal. Loosely speaking, the relative volumes of the regions describe the prevalence of that particular Gröbner basis. Gröbner fans have been used for model selection in (Dimitrova et al., 2007), but there are other methods as well.

Let's continue our running example from the previous section and the model space $\mathrm{Mod}(\mathcal{D})$ from Eq. (12). Our algorithm gave us a particular solution of $f = y(z+1)$, which was one of the 32 functions in $\mathrm{Mod}(\mathcal{D})$. It turns out that the Gröbner normal form[1] of f is $\mathrm{NF}(f) = y + z$. This means that an alternative way to write the model space, with a "fully reduced" representative particular solution, is

$$\begin{aligned}\mathrm{Mod}(\mathcal{D}) &= f + I = \mathrm{NF}(f) + I = y + z + \langle x + y, (y+1)z \rangle \\ &= \{ y + z + (x+y)h_1(\mathbf{x}) + (y+1)zh_2(\mathbf{x}) \mid h_1, h_2 \in Q \}.\end{aligned}$$

Another answer to the model selection problem is to try to select functions that agree with known biological information. For example, perhaps a certain interaction is known, and so only functions where that variable appears will be considered. One could search for functions by carefully selecting the coordinate ℓ in the construction of the indicator functions when possible, or by computing the Gröbner normal form for various monomial orderings and selecting one that has the known variable(s).

In other cases, it is not the variables, but rather the functions of a certain type that are given preference, because certain functions are deemed to be more "biologically relevant" than others. One such class are the *canalizing functions* (Waddington, 1942), and especially the *nested canalizing functions* (Kauffman et al., 2003). Loosely speaking, a function $f : \mathbb{F}^n \to \mathbb{F}$ is canalizing if it depends on a variable x_i for which taking a specific value completely determines the function's output. In other words, there is some $a \in \mathbb{F}$, for which $x_i = a$ implies that $f(x) = b$ for some $b \in \mathbb{F}$. On the other hand, if $x_i \neq a$, then the output is some function on $n-1$ variables, and we can ask whether or not that function is canalizing. Repeating this process leads to the notion of *canalizing depth*, and functions with "full depth" are called nested canalizing (He and Macauley, 2016). An example of a nested canalizing function on three variables is $f = x \wedge (y \vee \bar{z})$. The prevalence of canalization in biological networks has been well-studied (Daniels et al., 2018). It has been suggested (Harris et al., 2002), with recent supporting evidence (Kadelka, 2020), that "most" functions that appear in models of molecular networks are nested canalizing, or at least close.

Examples of nested canalizing functions include products of variables and their negations. One of these, written in both Boolean logical form and polynomial form, is

$$f(x,y,z) = x \wedge \bar{y} \wedge \bar{z} = x(y+1)(z+1) = x\bar{y}\bar{z}.$$

The last way to express the polynomial above, as $f = x\bar{y}\bar{z}$ is in some sense a hybrid of Boolean and polynomial form, but it motivates the use of the term *pseudomonomial*. Informally, the function f "is a monomial" in the "variables" x, y, z, $\bar{x}$, $\bar{y}$, $\bar{z}$. (This can be formalized by an operation called *polarization* (Güntürkün et al., 2019), which we will discuss later.) General nested canalizing functions have a more complicated structure than this, but in some sense, they are built recursively from these pseudomonomials. It is interesting that this concept of a pseudomonomial, which does not seem to be prevalent in classical mathematics at all, has already arisen in two very different problems involving algebraic models in mathematical biology.

3.9 Control of molecular networks

So far, we have examined how algebraic models can be constructed, how they can be used to determine the system's attractors, and how those attractors may be viewed as representing biologically meaningful states (e.g., a healthy state vs. a diseased state, different cell fates or system behaviors). Once a mathematical model of a molecular system is developed and validated, a major

[1] with respect to the standard default monomial ordering, grevlex

goal is to employ the model to suggest ways for control and for designing interventions (e.g., gene knockout, sustained external signals, protein synthesis inhibition) to drive a system into an attractor that corresponds to a desirable biological state. At this stage, the model can also assist, e.g., with determining signaling pathways or with identifying subsets of nodes and interactions most relevant to the system function(s).

The connection between network structure, type of interactions, and stabilization is complex and not completely understood, even though this field has attracted considerable interest in the last few decades. However, successful strategies have been designed for identifying control targets (Murrugarra et al., 2016), controlling gene regulation (Li and Wang, 2017; Murrugarra and Dimitrova, 2015), cell fate control (Zañudo and Albert, 2015), stem cell programming (Yachie-Kinoshita et al., 2018), and disruption of oncogenic pathways (Sordo Vieira et al., 2020), among many others. In general, the question is how to determine interventions to drive the systems into desirable states and away from those leading to disease or disadvantage. There is a rich arsenal of algebraic approaches, including some based on structures that we have already seen. For example, in (Murrugarra and Dimitrova, 2015), the authors propose a technique based on canalization and two types of control actions: edge deletion and constant expression of edges. In (Murrugarra et al., 2016), the authors translate the problem of finding control candidates into the problem of solving a system of polynomial equations, and then employ computational algebra techniques to find such controllers. The latter approach is based on finding a Gröbner basis, similar to what we did in Section 3.6. These control strategies have been validated on well-studied examples, e.g., the *p53-mdm2* network (Choi et al., 2012), a mammalian cell cycle network (Fauré et al., 2006), and a blood T cell lymphocyte granular leukemia survival signaling network (Saadatpour et al., 2011).

Graph-theoretic techniques provide another way for identifying the attractors of a system and developing control strategies. Specifically, in a series of publications, Albert, Zañudo and collaborators use a so-called "expanded network of interactions" for a system to encode the Boolean update rules in addition to the network topology. The expanded network is constructed by introducing additional nodes in a way that makes it possible to reflect the precise nature of multiple inputs, thus combining both the structure and function of molecular interactions (resolving in particular, the OR/AND ambiguity that is inherent to wiring diagrams). We refer the reader to (Zañudo and Albert, 2013) for the specifics. Chapter 4 in (Robeva, 2015) also contains multiple examples.

The problem of *stabilization* in network control is to determine properties of the network and/or external interventions that would (i) guarantee that the network or some of its vital components settle in certain states regardless of the system's initial conditions, and then (ii) use its strongly connected components to identify *stable motifs*—network components that stabilize in a fixed state. In (Yang et al., 2018) and (Zañudo and Albert, 2015), for example, the authors provide algorithms that could drive a system, through transitions between stable motifs, into a desirable system attractor. Other approaches to controlling Boolean models include matrix methods and semi-tensor products (Li and Wang, 2017; Liu et al., 2016; Zhong et al., 2017), integer programming (Qiu et al., 2014), and genetic algorithms (Vera-Licona et al., 2013), among many others.

Furthermore, large biological networks are known to exhibit inherent modularity—that is, a specific function of the cell, tissue, system, or organism that is performed by a specific set of interactions responsible for its proper execution. In many cases, after the mechanism of such interactions is well-understood, this network may be encapsulated as a module and included in a larger model as a "black box" for which certain sets of inputs produce outputs representing proper biological signaling and function. The interdependence of these functions could then be represented by aggregate networks of modules describing their combined and interconnected behavior. When the outputs from one of the modules are considered as inputs for another, Chaves and Tournier (2018) have shown that the attractors of the resulting interconnected network may be constructed only from the set of attractors of the modules. This synthetic approach underscores once again the importance of finding control strategies that drive each module into a desirable attractor, which would in turn allow for controlling the interconnected system.

Circling back to the question of update regimes, we should stress that different methods for analysis and control may work for certain update schemes and not for others—e.g., among the approaches mentioned above, the analysis and control methods based on the study of stable motifs in (Yang et al., 2018; Zañudo and Albert, 2015) and those for interconnected Boolean networks in (Chaves and Tournier, 2018) apply to asynchronous stochastic updates, whereas the results for feedback stabilization control in (Li and Wang, 2017) have been proven to work only for deterministic update schemes.

The use of asynchronous updates is not the only way to introduce stochasticity in logical models. Shmulevich et al. (2002) introduced probabilistic Boolean network (PBN) models by allowing multiple regulatory functions for each node. Which function is used to update that node at each step is determined by a probabilistic distribution on the inputs (predictors) and the modeling process is about the choice of update rule on each iteration. A different strategy was used in 2007 where Ribeiro and Kauffman (2007) considered noisy Boolean networks for which the state of a node flips from 0 to 1 and vice versa with a certain probability p independent from the other nodes. Yet another type of algebraic model, the so-called stochastic discrete dynamical systems (SDDS), was proposed in 2012 by Murrugarra et al. (2012). These model stochasticity at the functional level by introducing two update probabilities, $p^{\uparrow}$ and $p^{\downarrow}$, for each node. The parameter $p^{\uparrow}$ gives the probability of updating the node when it would produce a positive change (activation), that is, if the update makes the variable increase its value from 0 to 1. Likewise, the parameter $p^{\downarrow}$ gives the probability of updating the node when it makes the variable change its value from 1 to 0. This framework allows for an activation or inhibition determined by the update rule to be forgone due to stochasticity. In general, all of the

aforementioned types of models can be analyzed and controlled by using various algebraic strategies.

Another group of models, the so-called agent-based or individual-based models (ABM, IBM) of biological systems, has also benefited from the use of algebraic approaches to describe such models mathematically and design control strategies. ABMs feature a (usually large) set of discrete interactive components, called agents, rather than interactions between homogeneous compartments/populations. Each agent's spatiotemporal dynamics are determined by a finite number of rules that depend on the state of the agent, as well as those of other agents and their environment. ABMs have been used to model complex systems at all levels of biological organization—from molecular interactions to communities and ecosystems. Their main advantages over the so-called equation-based models are that they can be used to model heterogeneous systems and can produce a large diversity of system behaviors emerging from relatively small sets of local interaction rules—see, e.g., (Railsback and Grimm, 2019). The use of ABMs for simulating systems of interactive components has a long history—a good summary for their use in biology could be found in (Trexler et al., 2011, Chapter 12).

As with all other types of models discussed so far, the question of how to find a set of inputs for an ABM that achieves a specific outcome is of particular interest. This may be challenging, as determining a system's drivers toward desired behaviors in an ABM is not completely understood. In fact, until Grimm et al. (2006, 2010) introduced a standard protocol for mathematical descriptions of ABMs (the so-called ODD protocol—for Overview, Design concept, Detail), there was no broadly-accepted norm for a mathematical encoding of ABM models, pretty much leaving the use of repeated simulations as a sole methods for their analysis. Subsequently, an ODD extension proposed in (Hinkelmann et al., 2011) allowed for recasting ABMs as polynomial dynamical systems that preserve the models' features while providing tools from computational algebra for their analysis. An introduction to this strategy for optimal control of ABMs, together with examples, can be found in (Robeva and Hodge, 2013, Chapter 5). The perspective article by An et al. (2017) examines and references other systems-level control approaches for ABMs. Still, a comprehensive solution to the optimal control problem for ABMs is still lacking.

In summary, the field of network control is yet another rapidly expanding area of research where algebraic biology plays an important role. Advancements in molecular biology, coupled with algebraic models and increased computational power, hold significant promise for future medical breakthroughs—from finding cures for diseases, to designing targeted interventions in signaling systems, to moving us closer to personalized medicine.

4 Inferring place fields in neuroscience

The theme in the previous section of this paper was how algebraic structures arise in models of molecular networks. In particular, we saw how computational algebra is well-suited for solving certain inverse problems. This mathematical framework involves polynomial rings over finite fields, their ideals, and corresponding algebraic varieties. Along the way, we encountered Gröbner bases, square-free monomial ideals, simplicial complexes, and primary decompositions. A new object called a pseudomonomial arose in various settings, and this led to the concept of a pseudomonomial ideal, and a generalization of some of the ideas of the classical Stanley-Reisner theory. This is a firsthand example of how biology can indeed lead to new mathematics (Sturmfels, 2005). In this section, we will survey a very different biological application—one coming from neuroscience, where all of the aforementioned mathematical structures and concepts also arise.

4.1 Place fields as binary codes

Place cells are types of neurons located in the hippocampus, and they fire when an animal enters a particular region in space, called a *place field*. These place fields are generally overlapping and two-dimensional, and so they somewhat resemble a Venn diagram. As such, when an animal moves around, different subsets of place cells collectively fire in its brain. The location and shape of the specific place fields are not known, so *one goal is to infer, or reverse engineer their structure, from experimental data*. This is another instance of an inverse problem. And like the inverse problems that we saw in the previous section involving molecular networks, an algebraic framework is a natural way to analyze it. An example of a collection of five place fields is shown in Figure 2.

As an animal moves around its environment, typically an enclosure such as a room or cage, data of which neurons are firing can be collected at various points in time. Curto et al. (2013) proposed studying the structure of place fields from a coding theory perspective. If there are n place cells, then this experimental data can be represented as a collection of binary vectors or strings of length n. Such a collection is called a *neural code*, and denoted $\mathcal{C}$. The elements of $\mathcal{C}$ are called *codewords*, and they are usually written as a vector $\mathbf{c} = (c_1, \ldots, c_n)$ or just as a string $\mathbf{c} = c_1 c_2 \cdots c_n$. Binary strings of length n that are not in $\mathcal{C}$ are called *non-codewords*.

Every collection $\mathcal{U} = \{U_1, \ldots, U_m\}$ of place fields canonically generates a code $\mathcal{C}(\mathcal{U})$. Loosely speaking, each nonempty region in the "Venn diagram" of the place fields defines a codeword, with $c_i = 1$ if that region is contained in U_i and $c_i = 0$

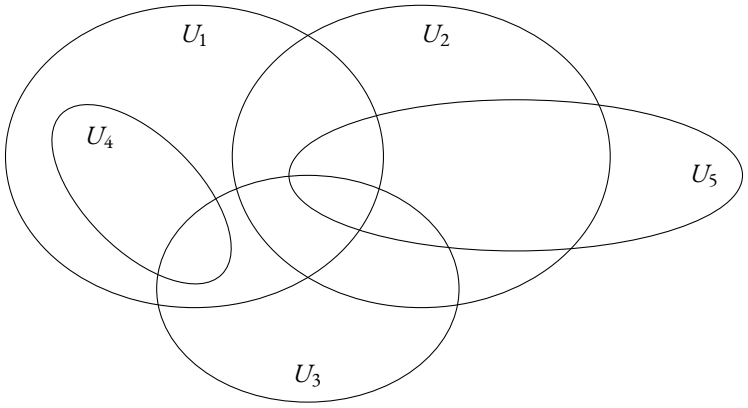

Figure 2: A collection $\mathcal{U} = \{U_1, U_2, U_3, U_4, U_5\}$ of five place fields. Each of the regions can be canonically represented by a binary string $\mathbf{c} = c_1c_2c_3c_4c_5$ called a codeword, and the set of all such strings is its code, $\mathcal{C}(\mathcal{U})$.

otherwise. For example, the code defined by the place fields in Figure 2 is

$$\mathcal{C}(\mathcal{U}) = \{00000,\ 10000,\ 01000,\ 00100,\ 00001,\ 11000,\ 10100,\ 10010,\ 01100,\ 01001,\ 11100,\ 11001,\ 10110,\ 01101,\ 11101\}.$$

Notice how the diagram of place fields in Figure 2 actually has 16 "regions", but two of them correspond to the same codeword, $\mathbf{c} = 01000$, because the subset of U_2 that is disjoint from the other U_i's is disconnected.

We say that a code $\mathcal{C}$ can be *realized* if there is some collection $\mathcal{U}$ of place fields for which $\mathcal{C}(\mathcal{U}) = \mathcal{C}$. It is not too hard to show that every code can be realized if we place no restrictions on the dimension of the ambient space, or on the structure of the place fields—not requiring them to be open, convex, or even connected. However, such a construction is completely detached from the original biological problem from which they arose. Place cells in rats were discovered in the early 1970s by O'Keefe, and this work earned him a Nobel Prize in 2014 (O'Keefe and Dostrovsky, 1971). Place cells were later found in other rodents and primates (Hori et al., 2005), including humans (Ekstrom et al., 2003). Since these animals do not fly, their place fields are regions in two-dimensional space, and they also end up generally being convex. This provides a strong restriction on the structure of place fields, and it makes the question about which codes are realizable more complex. Though there has been experimental evidence demonstrating three-dimensional place fields in bats (Geva-Sagiv et al., 2015), we will assume throughout that all place fields are two-dimensional.

One of the first major mathematical problems in this new field was to characterize which neural codes $\mathcal{C}$ are *convex realizable*, especially in two dimensions (Curto et al., 2017). For a few years, there was a thrust to characterize this property algebraically (Cruz et al., 2019; Curto et al., 2019), as we will soon see how every code can be represented with a pseudomonomial ideal, called a *neural ideal*. A good deal of progress was made along these lines, though necessary and sufficient conditions remained elusive. Recently, it was shown using the theory of oriented matriods that this problem is NP-hard (Kunin et al., 2020). Despite this, there are still many open problems about convex realizability. There has also been a lot of interest in studying neural ideals and the associated combinatorics on their own right. Indeed, much of the current research along these lines lacks direct connections back to the original problem from neuroscience. Rather, it involves a rich mathematical theory drawing from ideas in geometric and algebraic combinatorics, topology, commutative algebra, algebraic geometry, category theory, and theoretical computer science. Despite this, there are still researchers who use this framework for studying problems in neuroscience. This is another example of how biology can inspire entirely new problems and subfields in mathematics.

4.2 Pseudomonomials of codewords

Recall from the previous section on algebraic models how the disposable sets with respect to a set $\mathcal{D}$ of data form a simplicial complex. The corresponding Stanley-Reisner ideal is generated by the non-disposable sets, and its primary decomposition describes the min-sets. Furthermore, this can be extended to signed min-sets, using pseudomonoimal ideals, and the underlying mathematical theory still holds. A similar construction can be done for neural ideals, but the algebraic and combinatorial structures have different interpretations. It is interesting that pseudomonomials arise in both settings, whereas they seem to not really have been studied much before the emergence of algebraic biology in the 21st century.

As we did with a data set $\mathcal{D}$ in the setting of algebraic models, we can similarly encode neural code data with pseudomonomials. These are a little different, because this time, every variable will appear in every polynomial. Specifically, every codeword

$\mathbf{c}$ has a *characteristic polynomial* $\chi_{\mathbf{c}}(\mathbf{x})$, where $\mathbf{x} = (x_1, \ldots, x_n)$, such that

$$\chi_{\mathbf{c}}(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} = \mathbf{c} \\ 0 & \mathbf{x} \neq \mathbf{c}. \end{cases}$$

It is not hard to see how to construct such a pseudomonomial: include x_i if $c_i = 1$ and $(x_i - 1)$ if $c_i = 0$, and then multiply these all together for $i = 1, \ldots, n$. The *support* of a codeword $\mathbf{c}$ is defined to be the coordinates in which $c_i = 1$. Biologically, the support describes a set of neurons that fire together.

At this point, we have not specified which field the polynomials are over. If we use $\mathbb{F}_2$, then $x_i - 1 = x_i + 1$, which we can abbreviate as $\overline{x_i}$ for "NOT x_i." Despite this, there is undeniably something psychologically pleasing about using $x_i - 1$, because it makes it clear that $x_i = 1$ is a root. As an example, the characteristic polynomial of $\mathbf{c} = 011001$ over $\mathbb{F}_2$ is

$$\chi_{\mathbf{c}}(\mathbf{x}) = (x_1 - 1)x_2x_3(x_4 - 1)(x_5 - 1)x_6 = \overline{x_1}x_2x_3\overline{x_4}\,\overline{x_5}x_6.$$

Once again, if we express $\chi_{\mathbf{c}}(\mathbf{x})$ in a "hybrid" Boolean logic and polynomial form, as on the right above, we see the motivation for the term "pseudomonomial"—it is a monomial in the x_i's and $\overline{x_i}$'s. As before, this can be formalized via an operation called *polarization*, basically considering each $\overline{x_i}$ as a new variable, y_i; see (Güntürkün et al., 2019).

4.3 Rings and ideals of neural codes

Now, we are ready to define some basic algebraic objects arising from a neural code $\mathcal{C}$. The first is its *vanishing ideal*,

$$I_{\mathcal{C}} = \{f \in \mathbb{F}_2[x_1, \ldots, x_n] \mid f(\mathbf{c}) = 0 \text{ for all } \mathbf{c} \in \mathcal{C}\}.$$

Notice that for any non-codeword $\mathbf{n} \notin \mathcal{C}$, the characteristic polynomial $\chi_{\mathbf{n}}(\mathbf{x})$ vanishes on all $\mathbf{c} \in \mathcal{C}$. Therefore, the ideal generated by all of these polynomials is contained in the vanishing ideal $I_{\mathcal{C}}$. We call this the *neural ideal* of $\mathcal{C}$, denoted

$$J_{\mathcal{C}} = \langle \chi_{\mathbf{n}}(\mathbf{x}) \mid \mathbf{n} \notin \mathcal{C} \rangle \subseteq I_{\mathcal{C}}.$$

Another ideal that is trivially contained in the vanishing ideal is the set of all polynomials that vanish on *every* Boolean vector. This is called the *Boolean ideal*,

$$\mathcal{B} = \{b \in \mathbb{F}_2[x_1, \ldots, x_n] \mid b(\mathbf{x}) = 0 \text{ for all } \mathbf{x} \in \mathbb{F}_2^n\} = \langle x_i(1 - x_i) \mid i = 1, \ldots, n \rangle.$$

Recall that we have seen this ideal in the previous section, in Eq. (1) for the case $p = 2$. The ring $R = \mathbb{F}_2[x_1, \ldots, x_n]$ is the set of all Boolean polynomials, and since $x_i^2 = x_i$, the quotient ring $Q = R/\mathcal{B}$ is the set of all Boolean functions.

It turns out that the vanishing ideal is generated by the generators of the neural ideal $J_{\mathcal{C}} \subseteq I_{\mathcal{C}}$ and the Boolean ideal $\mathcal{B} \subseteq I_{\mathcal{C}}$. In other words,

$$I_{\mathcal{C}} = J_{\mathcal{C}} + \mathcal{B} = \langle \{\chi_{\mathbf{n}}(\mathbf{x}) \mid \mathbf{n} \notin \mathcal{C}\} \cup \{x_i^2 - x_i \mid i = 1, \ldots, n\} \rangle$$

for any code $\mathcal{C}$. The proof of this uses a version of Hilbert's Nullstellensatz for finite fields, sometimes called the *strong Nullstellensatz*. For details, see the Appendix of (Curto et al., 2013).

One more algebraic structure that is used for analyzing neural codes is called the *neural ring*, which is the quotient

$$R_{\mathcal{C}} = \mathbb{F}_2[x_1, \ldots, x_n]/I_{\mathcal{C}}.$$

This is a quotient of the ring $\mathbb{F}_2[x_1, \ldots, x_n]/\mathcal{B}$ of all Boolean functions $\mathbb{F}_2^n \to \mathbb{F}_2$. Specifically, if we associate $\mathcal{C} \subseteq \mathbb{F}_2^n$, then the neural ring $R_{\mathcal{C}}$ can be thought of as the set of all functions $\mathcal{C} \to \mathbb{F}_2$.

4.4 Simplicial complexes of neural codes

Thus far, we have been writing neural codes as collections of binary strings. Since each such string canonically describes a subset of $[n] = \{1, \ldots, n\}$, we can think of a code $\mathcal{C}$ alternatively as a collection of subsets. This is not necessarily a simplicial complex because it is does not need to be closed under taking subsets. However, it *generates* a simplicial complex if we throw in all codewords corresponding to subsets. Specifically the simplicial complex of a code $\mathcal{C}$ is defined as

$$\Delta(\mathcal{C}) = \{\sigma \subseteq [n] \mid \sigma \subseteq \mathbf{c} \text{ for some } \mathbf{c} \in \mathcal{C}\}.$$

Every simplicial complex is determined by its maximal faces (also called facets), which describe the maximal subsets of neurons that fire together. Equivalently, these are the maximal subsets of place fields that have nonempty intersections. In topology,

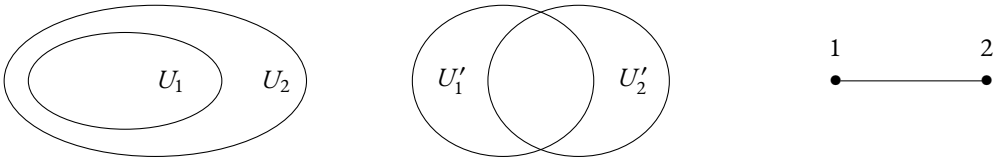

Figure 3: At left is a realization of the code $\mathcal{C} = \{01, 11\}$, and at right is its simplicial complex $\Delta(\mathcal{C}) = \{00, 10, 01, 11\}$. In the middle is a different collection of place fields with the same simplicial complex.

every open cover defines a simplicial complex called its *nerve* in this manner. In this setting, *nerve* of a collection $\mathcal{U}$ of place fields describes the set of all nonempty intersections in any realization of $\mathcal{U}$. Formally, this is the set

$$\mathcal{N}(\mathcal{U}) = \left\{\alpha \subseteq [n] \mid \bigcap_{i\in\alpha} U_i \neq \emptyset\right\}.$$

It is easy to verify that the simplicial complex of a code is simply the nerve of any of its realizations, i.e., $\Delta(\mathcal{C}) = \mathcal{N}(\mathcal{U})$. As an example of this, consider the code $\mathcal{C} = \{01, 11\}$, which does not describe a simplicial complex because it is missing 00 and 10. A realization of $\mathcal{C}$ is shown on the left in Figure 3, and at right is the simplicial complex $\Delta(\mathcal{C}) = \mathcal{N}(\mathcal{U})$.

Notice that different codes may have the same simplicial complex. For example, we can throw in 00 to the code on the left in Figure 3 without changing the simplicial complex; this would represent expanding the stimulus space to the complement U_2^c. Of course, there is no way to include 10 in the code without fundamentally changing the place fields to a configuration such as the one shown in the middle of Figure 3, which has the same simplicial complex.

4.5 Receptive field relationships

At this point, we have seen how a neural code can be described algebraically by pseudomonimal ideals, or combinatorially by its simplicial complex and nerve. Our goal is to tie these different viewpoints together, via the primary components of the neural ideal, much like what was done for algebraic models earlier in this paper.

First, notice how certain combinatorial features of the place fields are encoded by algebraic relationships. For example, if $c_1 = c_2 = 1$ for some $\mathbf{c} \in \mathcal{C}$, then we can conclude that $U_1 \cap U_2 \neq \emptyset$. This is called a *receptive field (RF) relationship*, introduced in (Curto et al., 2013). For another example, if $c_1 = 1$ implies that $c_2 = 1$, then $U_1 \subseteq U_2$, like the example on the left in Figure 3.

RF relationships can be deduced algebraically as well. Recall that the neural ideal $J_\mathcal{C}$ is generated by the characteristic polynomials $\chi_\mathbf{n}(\mathbf{x})$ of the non-codewords $\mathbf{n}$. This means that $\chi_\mathbf{n}(\mathbf{c}) = 0$ for any $\mathbf{c} \in \mathcal{C}$, and hence, $f(\mathbf{c}) = 0$ for any polynomial $f \in J_\mathcal{C}$. Suppose we have a polynomial

$$r(\mathbf{x}) = x_1(1 - x_2) \in J_\mathcal{C}.$$

Since $r(\mathbf{c}) = 0$ for any $\mathbf{c} \in \mathcal{C}$, then if $c_1 = 1$, it must be the case that $c_2 = 1$. This is a way to capture the RF relationship that $U_1 \subseteq U_2$ algebraically.

The example above generalizes from two coordinates to arbitrary subsets. For $\sigma \subseteq [n]$, let $x_\sigma = \prod_{i\in\sigma} x_i$. Next, for disjoint subsets σ and τ, let $p_{\sigma,\tau}$ be the pseudomonomial that is the product of all of the variables x_i from σ and terms $(1 - x_j)$ from τ. That is,

$$p_{\sigma,\tau}(\mathbf{x}) = x_\sigma \prod_{j\in\tau}(1 - x_j).$$

It is easy to check that $p_{\sigma,\tau}(\mathbf{x}) \in J_\mathcal{C}$ if and only if

$$\bigcap_{i\in\sigma} U_i \subseteq \bigcup_{j\in\tau} U_j, \tag{13}$$

where the empty intersection is the entire stimulus space X, and the empty union is $\emptyset$. There are two special cases of this, corresponding to σ or τ being empty:

1. $p_{\sigma,\emptyset}(\mathbf{x}) = x_\sigma \in J_\mathcal{C}$ if and only if $\bigcap_{i\in\sigma} U_i = \emptyset$;

2. $p_{\emptyset,\tau}(\mathbf{x}) = \prod_{j\in\tau}(1 - x_j) \in J_\mathcal{C}$ if and only if $\bigcup_{j\in\tau} U_j \supseteq X$.

An example of (1) can be found in Figure 2, with $\sigma = \{1, 2, 4\} \subseteq [5]$. By definition, J_C containing $p_{\sigma,\emptyset}(\mathbf{x}) = x_1x_2x_4$ means $p_{\sigma,\emptyset}(\mathbf{c}) = 0$ for all $\mathbf{c} \in C$. Equivalently, no word in C has the form $\mathbf{c} = 11c_31c_5$, because $U_1 \cap U_2 \cap U_4 = \emptyset$.

Let's consider an example of (2) with $\tau = \{1, 2\}$. Here, J_C containing $p_{\emptyset,\tau}(\mathbf{x}) = (1 - x_1)(1 - x_2)$ means $p_{\emptyset,\tau}(\mathbf{c}) = 0$ for all $\mathbf{c} \in C$. Equivalently, no word in C has the form $\mathbf{c} = 00\cdots$. That is, every point in the stimulus space X is either in U_1 or U_2, so $X \subseteq U_1 \cup U_2$. We saw an example of this on the left in Figure 3.

4.6 Canonical forms and primary decompositions

We say that a pseudomonomial J_C is *minimal* if there is no lower-degree pseudomonomial in J_C that divides it. In some sense, these describe minimal intrinsic RF relationships.

Definition 7. The *canonical form* of J_C, denoted $\mathrm{CF}(J_C)$, is the set of minimal pseudomonomials in J_C.

Theorem 8. *Every neural ideal is generated by its canonical form, i.e.,* $J_C = \langle \mathrm{CF}(J_C) \rangle$.

It should be noted that while elements in canonical forms describe minimal RF relationships, there are other intrinsic relationships that do not arise in this manner. Garcia et al. (2018) found additional RF relationships by studying the Gröbner bases of neural ideals.

An algorithm for computing the canonical form of a neural ideal was given in (Petersen et al., 2018). Alternatively, the canonical form can be constructed from the primary decomposition of J_C (Curto et al., 2013). We will briefly summarize the construction here. Each primary component is an ideal generated by *linear terms*: some x_i's and some other $(1 - x_j)$'s, with no variable appearing in both, and some x_k's possibly not appearing at all. We can encode these three possibilities with $a_i = 0$, $a_j = 1$, and $a_k = *$, respectively, and thus describe each primary component with a length-n word $a \in \{0, 1, *\}^n$. In other words, the primary ideal defined by $a \in \{0, 1, *\}^n$ is

$$\mathfrak{p}_a = \langle x_i - a_i \mid a_i \neq * \rangle = \langle \{x_i \mid a_i = 0\},\ \{1 - x_j \mid a_j = 1\} \rangle.$$

The primary decomposition of the neural ideal J_C thus has the form

$$J_C = \bigcap_{a \in \mathcal{A}} \mathfrak{p}_a, \tag{14}$$

for some collection of words $\mathcal{A} \subseteq \{0, 1, *\}^n$. The canonical form of J_C is the set of its minimal pseudomonomials, and this can be constructed from this primary decomposition, which can easily be found with a computational algebra software package. Specifically, let $\mathcal{P}$ be the set of pseudomonomials formed by taking the product of *at most* one linear term from each $\mathfrak{p}_a$. The minimal pseudomonomials of this set is precisely $\mathrm{CF}(J_C)$.

In Stanley-Reisner theory, the primary decomposition connects the algebraic "factorization" of a square-free monomial ideal with the combinatorial structure of the associated simplicial complex. Via polarization (i.e., replacing each $1 - x_i$ with y_i), a pseudomonomial ideal becomes a square-free monomial ideal in $\mathbb{F}_2[x_1, \ldots, x_n, y_1, \ldots, y_n]$, and the corresponding simplicial complex is called the *polar complex* (Güntürkün et al., 2019). This is related to an object called the *factor complex* (de Perez et al., 2020), which is the simplicial complex of the ideal defined by polarizing the primary components in Eq. (14). Though Stanley-Reisner theory does not apply directly to pseudomonomials, many tools can still be used to uncover combinatorial relationships in the algebra, and vice-versa, leading to new structure theorems in algebraic models of molecular networks, mathematical neuroscience, and pure mathematics.

5 Closing remarks

In this paper we chose to present two very different examples from biology and how they can be examined through the lens of pseudomonomials. These are two examples from the growing field of *algebraic biology*, a discipline with a repertoire that also features alternative approaches based on graph theory, linear algebra, algebraic statistics, algebraic geometry, and many others (Macauley and Youngs, 2020). In addition, stochastic treatments can lead to probabilistic models, including Markov and semi-Markov processes. The three edited volumes (Robeva, 2015; Robeva and Hodge, 2013; Robeva and Macauley, 2018) may be used as an entry point to its rich toolkit, as they present a variety of biology problems benefiting from the vast arsenal of algebraic and combinatorial tools, including, e.g., food chains, multi-scale biomolecular structures, RNA and DNA rearrangements, biochemical reaction networks, multistationarity of biological systems, neural network dynamics, phylogenetics, metabolic pathways, evolutionary landscapes, quantitative traits analysis, and the spread of infectious diseases on networks, and many others. The recent collection of papers published as a special Research Topic of *Frontiers in Physiology* titled "Logical modeling of cellular processes: from software development to network dynamics" (Barberis and Helikar, 2019) is yet another great resource on algebraic models.

In closing, we reiterate that the field of algebraic biology has expanded tremendously in the last decade and is now a booming area of active research. This paper provides a glimpse into the field through topics where authentic questions from biology may be answered in the language of pseudomonomials, tying together two unrelated subfields in a way that had not been done before in the literature. Our hope is that the reader will be intrigued and inspired to explore the literature independently—the references included in this article and beyond—and to further discover how the symbiosis of mathematics and biology drives new discoveries in each of these fields.

Acknowledgments

The first author was partially supported by Simons Foundation Grant #358242. The second author was partially supported by the Karl Peace Fellowship in Mathematics, Randolph-Macon College.

References

Abou-Jaoudé, W., P. Traynard, P. T. Monteiro, J. Saez-Rodriguez, T. Helikar, D. Thieffry, and C. Chaouiya (2016). Logical modeling and dynamical analysis of cellular networks. *Front. Genet. 7*, 94. 84

Albert, R. and H. Othmer (2003). The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*. *J. Theor. Biol. 223*(1), 1–18. 83, 86

Allen, E., J. Fetrow, L. Daniel, S. Thomas, and D. John (2006). Algebraic dependency models of protein signal transduction networks from time-series data. *J. Theor. Biol. 238*(2), 317–330. 89

Alon, U. (2019). *An introduction to systems biology: design principles of biological circuits* (2 ed.). CRC press. 82

An, G., B. Fitzpatrick, S. Christley, P. Federico, A. Kanarek, R. M. Neilan, M. Oremland, R. Salinas, R. Laubenbacher, and S. Lenhart (2017). Optimization and control of agent-based models in biology: a perspective. *Bull. Math. Biol. 79*(1), 63–87. 95

Andrecut, M. and S. A. Kauffman (2008). On the sparse reconstruction of gene networks. *J. Comp. Biol, 15*(1), 21–30. 90

Barberis, M. and T. Helikar (2019). *Logical Modeling of Cellular Processes: From Software Development to Network Dynamics*. Lausanne, Switzerland: Frontiers Media. 99

Chaves, M. and L. Tournier (2018). Analysis tools for interconnected Boolean networks with biological applications. *Front. Physiol. 9*, 586. 94

Choi, M., J. Shi, S. H. Jung, X. Chen, and K.-H. Cho (2012). Attractor landscape analysis reveals feedback loops in the p53 network that control the cellular response to DNA damage. *Sci. Signal. 5*(251), ra83. 94

Chomsky, N. (2014). *Aspects of the Theory of Syntax*, Volume 11. MIT press. 82

Cox, D., J. Little, and D. O'Shea (2015). *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra* (4 ed.). Springer. 93

Cruz, J., C. Giusti, V. Itskov, and B. Kronholm (2019). On open and closed convex codes. *Discrete Comput. Geom. 61*(2), 247–270. 96

Curto, C., E. Gross, J. Jeffries, K. Morrison, M. Omar, Z. Rosen, A. Shiu, and N. Youngs (2017). What makes a neural code convex? *SIAM J. Appl. Alg. Geom. 1*(1), 222–238. 96

Curto, C., E. Gross, J. Jeffries, K. Morrison, Z. Rosen, A. Shiu, and N. Youngs (2019). Algebraic signatures of convex and non-convex codes. *J. Pure Appl. Alg. 223*(9), 3919–3940. 96

Curto, C., V. Itskov, K. Morrison, Z. Roth, and J. L. Walker (2013). Combinatorial neural codes from a mathematical coding theory perspective. *Neural Comput. 25*(7), 1891–1925. 95

Curto, C., V. Itskov, A. Veliz-Cuba, and N. Youngs (2013). The neural ring: an algebraic tool for analyzing the intrinsic structure of neural codes. *Bull. Math. Biol. 75*(9), 1571–1611. 97, 98, 99

Daniels, B. C., H. Kim, D. Moore, S. Zhou, H. B. Smith, B. Karas, S. A. Kauffman, and S. I. Walker (2018). Criticality distinguishes the ensemble of biological regulatory networks. *Phy. Rev. Lett. 121*(13), 138102. 93

de Perez, A. R., L. F. Matusevich, and A. Shiu (2020). Neural codes and the factor complex. *Adv. Appl. Math. 114*, 101977. 99

Decker, W., G.-M. Greuel, G. Pfister, and H. Schönemann (2020). SINGULAR 4-1-3 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de. 85

Dimitrova, E. (2010). Estimating the relative volumes of the cones in a Gröbner fan. *Math. Comput. Sci. 3*(4), 457–466. 87

Dimitrova, E., A. Jarrah, R. Laubenbacher, and B. Stigler (2007). A Gröbner fan method for biochemical network modeling. In *Proc. Internat. Symposium Symb. Algebraic Comput.*, pp. 122–126. ACM. 93

Ekstrom, A. D., M. J. Kahana, J. B. Caplan, T. A. Fields, E. A. Isham, E. L. Newman, and I. Fried (2003). Cellular networks underlying human spatial navigation. *Nature 425*(6954), 184–188. 96

Fauré, A., A. Naldi, C. Chaouiya, and D. Thieffry (2006). Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics 22*(14), e124–e131. 94

Fernández-Tajes, J., K. J. Gaulton, M. van de Bunt, J. Torres, M. Thurner, A. Mahajan, A. L. Gloyn, K. Lage, and M. I. McCarthy (2019). Developing a network view of type 2 diabetes risk pathways through integration of genetic, genomic and functional data. *Genome Med. 11*(1), 19. 83

Francisco, C. A., J. Mermin, and J. Schweig (2014). A survey of Stanley–Reisner theory. In *Connections Between Algebra, Combinatorics, and Geometry*, pp. 209–234. Springer. 88

Friedman, N., M. Linial, I. Nachman, and D. Pe'er (2000). Using Bayesian networks to analyze expression data. *J. Comp. Biol. 7*(3-4), 601–620. 86

Garcia, R., L. D. G. Puente, R. Kruse, J. Liu, D. Miyata, E. Petersen, K. Phillipson, and A. Shiu (2018). Gröbner bases of neural ideals. *Int. J. Algebr. Comput. 28*(04), 553–571. 99

Gardner, M. (1970). Mathematical games: The fantastic combinations of John Conway's new solitaire game "life". *Sci. Am. 223*, 120–123. 82

Gershenson, C. (2004). Introduction to random Boolean networks. *arXiv:0408006 [nlin.AO]*. 85

Geva-Sagiv, M., L. Las, Y. Yovel, and N. Ulanovsky (2015). Spatial cognition in bats and rats: from sensory acquisition to multiscale maps and navigation. *Nat. Rev. Neurosci. 16*(2), 94–108. 96

Goles, E. and S. Martínez (2013). *Neural and automata networks: dynamical behavior and applications*, Volume 58. Springer Science & Business Media. 84

Grayson, D. and M. Stillman (2020). Macaulay2, a software system for research in algebraic geometry. Available at http://www2.macaulay2.com/Macaulay2/. 85

Grimm, V., U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S. K. Heinz, G. Huse, et al. (2006). A standard protocol for describing individual-based and agent-based models. *Ecol. Model. 198*(1-2), 115–126. 95

Grimm, V., U. Berger, D. L. DeAngelis, J. G. Polhill, J. Giske, and S. F. Railsback (2010). The ODD protocol: a review and first update. *Ecol. Model. 221*(23), 2760–2768. 95

Güntürkün, S., J. Jeffries, and J. Sun (2019). Polarization of neural rings. *J. Algebra Appl.*, 2050146 (17 pages). 89, 93, 97, 99

Harris, S. E., B. K. Sawhill, A. Wuensche, and S. Kauffman (2002). A model of transcriptional regulatory networks based on biases in the observed regulation rules. *Complexity 7*(4), 23–40. 93

Hartemink, A. J., D. K. Gifford, T. S. Jaakkola, and R. A. Young (2000). Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Biocomputing 2001*, pp. 422–433. World Scientific. 86

He, Q. and M. Macauley (2016). Stratification and enumeration of Boolean functions by canalizing depth. *Physica D 314*, 1–8. 93

Hinkelmann, F. and R. Laubenbacher (2011). Boolean models of bistable biological systems. *Discrete Cont. Dyn. Sys. Ser. S 4*(6), 1443–1456. 85

Hinkelmann, F., D. Murrugarra, A. S. Jarrah, and R. Laubenbacher (2011). A mathematical framework for agent based models of complex biological networks. *Bull. Math. Biol. 73*(7), 1583–1602. 95

Hori, E., Y. Nishio, K. Kazui, K. Umeno, E. Tabuchi, K. Sasaki, S. Endo, T. Ono, and H. Nishijo (2005). Place-related neural responses in the monkey hippocampal formation in a virtual space. *Hippocampus 15*(8), 991–996. 96

Jacob, F., D. Perrin, C. Sánchez, and J. Monod (1960). L'opéron: groupe de gènes à expression coordonnée par un opérateur. *C.R. Acad. Sci. 250*, 1727–1729. 82

Jarrah, A., R. Laubenbacher, B. Stigler, and M. Stillman (2007). Reverse-engineering of polynomial dynamical systems. *Adv. Appl. Math. 39*(4), 477–489. 87, 88

Jenkins, A. and M. Macauley (2017). Bistability and asynchrony in a Boolean model of the L-arabinose operon in *Escherichia coli*. *Bull. Math Biol. 79*(8), 1778–1795. 86

Kadelka, C. (2020). Personal communication; paper in preparation. 93

Kauffman, S. (1969a). Homeostasis and differentiation in random genetic control networks. *Nature 224*, 177–178. 82

Kauffman, S. (1969b). Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol. 22*(3), 437–467. 82

Kauffman, S., C. Peterson, B. Samuelsson, and C. Troein (2003). Random Boolean network models and the yeast transcriptional network. *Proc. Natl. Acad. Sci. 100*(25), 14796–14799. 93

Kunin, A., C. Lienkaemper, and Z. Rosen (2020). Oriented matroids and combinatorial neural codes. *arXiv:2002.03542 [math.CO]*. 96

Laubenbacher, R., V. Hower, A. Jarrah, S. Torti, V. Shulaev, P. Mendes, F. Torti, and S. Akman (2009). A systems biology view of cancer. *Biochim. Biophys. Acta 1796*(2), 129–139. 83

Laubenbacher, R. and B. Stigler (2004). A computational algebra approach to the reverse engineering of gene regulatory networks. *J. Theor. Biol. 229*(4), 523–537. 90, 91

Li, H. and Y. Wang (2017). Further results on feedback stabilization control design of Boolean control networks. *Automatica 83*, 303–308. 94

Li, S., S. Assmann, and R. Albert (2006). Predicting essential components of signal transduction networks: a dynamic model of guard cell abscisic acid signaling. *PLoS Biol. 4*(10), e312. 83

Liu, R., C. Qian, S. Liu, et al. (2016). State feedback control design for Boolean networks. *BMC Syst Biol. 10*(70). 94

Lotka, A. J. (1925). *Elements of Physical Biology*. Baltimore, MD: Williams and Wilkins Co. 82

Macauley, M. and N. Youngs (2020). The case for algebraic biology: from research to education. *Bull. Math. Biol.*, to appear. 81, 99

Malthus, T. R. (1803). *An essay on the principle of population; a view of its past and present effects on human happiness; with an inquiry into our prospects respecting the future removal or mitigation of the evils which it occasions.* (2 ed.). 82

Marí, M. and J. C. Fernández-Checa (2007). Sphingolipid signalling and liver diseases. *Liver Int. 27*(4), 440–450. 83

Mount, D. W. (2004). *Bioinformatics: sequence and genome analysis* (2 ed.). Cold Spring Harbor Laboratory Press. 82

Murrugarra, D. and E. Dimitrova (2015). Molecular network control through Boolean canalization. *EURASIP J. Bioinformatics Sys. Biol. 2015*(1), 1–8. 94

Murrugarra, D., A. Veliz-Cuba, B. Aguilar, S. Arat, and R. Laubenbacher (2012). Modeling stochasticity and variability in gene regulatory networks. *EURASIP J. Bioinformatics Sys. Biol. 2012*(1), 1–11. 94

Murrugarra, D., A. Veliz-Cuba, B. Aguilar, and R. Laubenbacher (2016). Identification of control targets in Boolean molecular network models via computational algebra. *BMC Syst. Biol. 10*(1), 94. 94

Naldi, A., E. Remy, D. Thieffry, and C. Chaouiya (2011). Dynamically consistent reduction of logical regulatory graphs. *Theor. Comput. Sci. 412*(21), 2207–2218. 86

O'Keefe, J. and J. Dostrovsky (1971). The hippocampus as a spatial map: preliminary evidence from unit activity in the freely-moving rat. *Brain Res.*. 96

Ozbudak, E. M., M. Thattai, H. N. Lim, B. I. Shraiman, and A. Van Oudenaarden (2004). Multistability in the lactose utilization network of *Escherichia coli*. *Nature 427*(6976), 737. 86

Petersen, E., N. Youngs, R. Kruse, D. Miyata, R. Garcia, and L. D. G. Puente (2018). Neural ideals in sagemath. In *ICMS*, pp. 182–190. Springer. 99

Qiu, Y., T. Tamura, W. Ching, et al. (2014). On control of singleton attractors in multiple Boolean networks: integer programming-based method. *BMC Syst. Biol. 8*(S7). 94

Railsback, S. F. and V. Grimm (2019). *Agent-based and individual-based modeling: a practical introduction*. Princeton University Press. 95

Ribeiro, A. S. and S. A. Kauffman (2007). Noisy attractors and ergodic sets in models of gene regulatory networks. *J. Theor. Biol. 247*(4), 743–755. 94

Robeva, R. (2015). *Algebraic and Discrete Mathematical Methods for Modern Biology*. Elsevier. 94, 99

Robeva, R. and T. Hodge (2013). *Mathematical concepts and methods in modern biology: using modern discrete models*. Academic Press. 83, 85, 86, 90, 95, 99

Robeva, R. and M. Macauley (2018). *Algebraic and Combinatorial Computational Biology*. Elsevier. 84, 87, 89, 99

Saadatpour, A., R.-S. Wang, A. Liao, X. Liu, T. Loughran, I. Albert, and R. Albert (2011, Nov). Dynamical and structural analysis of a t cell survival network identifies novel candidate therapeutic targets for large granular lymphocyte leukemia. *PLoS Comput Biol 7*(11). 86, 94

Saez-Rodriguez, J., L. G. Alexopoulos, J. Epperlein, R. Samaga, D. A. Lauffenburger, S. Klamt, and P. K. Sorger (2009). Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Mol. Syst. Biol. 5*(1). 83

Shmulevich, I. and E. R. Dougherty (2010). *Probabilistic Boolean networks: the modeling and control of gene regulatory networks*. SIAM. 85

Shmulevich, I., E. R. Dougherty, S. Kim, and W. Zhang (2002). Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. *Bioinformatics 18*(2), 261–274. 94

Sordo Vieira, L., R. Laubenbacher, and D. Murrugarra (2020). Control of intracellular molecular networks using algebraic methods. *Bull. Math. Biol. 82*(2). 94

Stigler, B. and H. Chamberlin (2012). A regulatory network modeled from wild-type gene expression data guides functional predictions in *Caenorhabditis elegans* development. *BMC Syst. Biol. 6*(1), 77. 89

Stoll, G., E. Viara, E. Barillot, and L. Calzone (2012). Continuous time Boolean modeling for biological signaling: application of Gillespie algorithm. *BMC Syst. Biol. 6*(1), 116. 85

Sturmfels, B. (2005). Can biology lead to new theorems? *Annu. Rep. Clay Math. Inst.*, 13–26. 95

Thieffry, D. and M. Kaufman (2019). Prologue to the special issue of JTB dedicated to the memory of René Thomas (1928–2017): A journey through biological circuits, logical puzzles and complex dynamics. *J. Theor. Biol. 474*, 42. 82

Thomas, R. (1973). Boolean formalization of genetic control circuits. *J. Theor. Biol. 42*(3), 563–585. 82

Trexler, M., S. E. Jørgensen, and D. DeAngelis (2011). *Modelling Complex Ecological Dynamics: An Introduction into Ecological Modelling for Students, Teachers & Scientists*. Springer Science & Business Media. 95

Veliz-Cuba, A. (2012). An algebraic approach to reverse engineering finite dynamical systems arising from biology. *SIAM J. Appl. Dyn. Syst. 11*(1), 31–48. 89

Veliz-Cuba, A., B. Aguilar, F. Hinkelmann, and R. Laubenbacher (2014). Steady state analysis of Boolean molecular network models via model reduction and computational algebra. *BMC Bioinformatics 15*(1), 221. 86

Veliz-Cuba, A. and L. Geiser (2016). The number of fixed points of AND-OR networks with chain topology. *arXiv:1609.02526 [math.CO]*. 86

Veliz-Cuba, A. and R. Laubenbacher (2012). On the computation of fixed points in Boolean networks. *J. Appl. Math. Comput. 39*(1-2), 145–153. 86

Veliz-Cuba, A. and B. Stigler (2011). Boolean models can explain bistability in the *lac* operon. *J. Comp. Biol. 18*(6), 783–794. 86

Vera-Licona, P., E. Bonnet, E. Barillot, and A. Zinovyev (2013, 04). OCSANA: optimal combinations of interventions from network analysis. *Bioinformatics 29*(12), 1571–1573. 94

Verhulst, P.-F. (1838). Notice on the law that the population follows in its growth. *Match. Math. Phys. 10*, 113–126. 82

Waddington, C. (1942). Canalization of development and the inheritance of acquired characters. *Nature 150*(3811), 563–565. 93

Waterman, M. S. (2014). *Introduction to computational biology: maps, sequences and genomes* (2 ed.). Chapman and Hall/CRC. 82

Yachie-Kinoshita, A., K. Onishi, J. Ostblom, M. A. Langley, E. Posfai, J. Rossant, and P. W. Zandstra (2018). Modeling signaling-dependent pluripotency with Boolean logic to predict cell fate transitions. *Mol. Syst. Biol. 14*(1). 94

Yang, G., J. Gómez Tejeda Zañudo, and R. Albert (2018). Target control in logical models using the domain of influence of nodes. *Front. Physiol. 9*, 454. 86, 94

Yeung, M. S., J. Tegnér, and J. J. Collins (2002). Reverse engineering gene networks using singular value decomposition and robust regression. *Proc. Natl. Acad. Sci. 99*(9), 6163–6168. 86

Zañudo, J. and R. Albert (2015, 04). Cell fate reprogramming by control of intracellular network dynamics. *PLoS Comput. Biol. 11*(4), e1004193. 86, 94

Zañudo, J. and R. Albert (2013). An effective network reduction approach to find the dynamical repertoire of discrete dynamic networks. *Chaos 23*(2), 025111. 86, 94

Zhang, R., M. V. Shah, J. Yang, S. B. Nyland, X. Liu, J. K. Yun, R. Albert, and T. P. Loughran (2008). Network model of survival signaling in large granular lymphocyte leukemia. *Proc. Natl. Acad. Sci. 105*(42), 16308–16313. 86

Zhong, J., D. W. Ho, J. Lu, and W. Xu (2017). Global robust stability and stabilization of Boolean network with disturbances. *Automatica 84*, 142–148. 94