Probabilistic Signaling Networks: Dynamic Modeling and Inference for Biology and Beyond

Marius P. Furter[*]
Institute of Mathematics
University of Zurich
Zurich, Switzerland
marius.furter@math.uzh.ch

Probabilistic signaling networks (PSNs) provide a high-level framework for dynamically modeling networked systems. They are realized as lenses in the Markov category of standard Borel spaces and Markov kernels. As such, they are inherently composable and support inference for the quantities they describe. By choosing appropriate activation maps, PSNs may be tailored to applications. We describe variants suitable for modeling biological signaling and transcription networks. We demonstrate their utility by inferring the logical structure of a feed-forward motif from simulated data.

1 Introduction

Biological organisms are complex systems that span many orders of magnitude in space and time. For instance, the micrometer long bacteria *E. Coli* contains around 4 million proteins of 4500 types, each several nanometers in size. While small molecules bind proteins and alter their function within milliseconds, it takes minutes to transcribe and translate a gene. Due to this separation of scales, higher-level processes can ignore the details of the lower levels they are built upon. Life is also surprisingly modular. Individual genes can be transplanted across distant species while retaining their function. Moreover, biological networks are often composed of a small number of conserved patterns called motifs. Taken together, these features allow evolution to efficiently prototype new designs.

Systems Biology takes a holistic approach to the study of life. In this endeavor, the use of mathematical modeling is indispensable for linking the realm of concepts to quantitative experimentation. The importance of such models grows continually as advances in high-throughput technologies generate ever-increasing amounts of data. Most textbook accounts of biomodeling take a concrete approach by directly formulating ideas using differential equations and linear algebra. Although flexible, this has the downside that practitioners must master many mathematical details before constructing interesting models. Moreover, a lack of overarching structure makes reusing and combining models challenging.

Here we propose a high-level mathematical framework for modeling networked systems that imitates the layered and modular architecture that has been time-tested by Nature. These features are achieved by layering the category-theoretic abstractions of Markov categories and lenses. The resulting probabilistic signaling networks are specified using a visual syntax that requires no understanding of the underlying mathematics apart from the biological meaning of the parameters. Moreover, they are inherently composable, customizable to the domain of application and support inference for all quantities they describe.

The paper is organized into three parts. Section 2 introduces ideas informally using minimal mathematics. Section 3 describes the mathematics and presupposes familiarity categories and measure theory. Practical utility is demonstrated in Section 4. Future work will implement this theory in software.

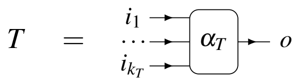
[*]I thank the University of Zurich and the Digital Society Initiative for funding this work. I also acknowledge the partial support of SNF Grant No. 200020_192080.

Submitted to:
Applied Category Theory 2023

2 Probabilistic signaling networks informally

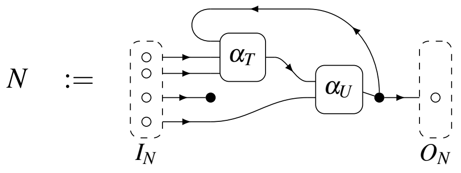
2.1 Structure

A probabilistic signaling network (PSN) is a dynamic open network constructed by wiring together a set of primitives called *transducers*. At each time-step, a transducer T changes its activation level $\alpha_T \in [0,1]$ according to the value of the input signals $i_1, \ldots, i_{k_T} \in [0,1]$ it receives. Subsequently, its activation level can be used as input signal for other transducers in the network. We depict a transducer T as a box



The manner in which input signals are transformed into an activation level is governed by the activation map $f_T^\sharp \colon [0,1] \times [0,1]^{k_T} \rightsquigarrow [0,1]$. It describes how the current activation level $\alpha_T(t)$ and the current inputs $i_1(t), \ldots, i_{k_T}(t)$ are converted into the activation level $\alpha_T(t+1)$ at the next time-step. Depending on the type of PSN, different families of activation maps may be used. Importantly, these maps are probabilistic maps called Markov kernels (Definition 3.1), as indicated by the squiggly arrow. As a special case, they can represent any measurable deterministic function $\bar{f}_T^\sharp \colon [0,1] \times [0,1]^{k_T} \to [0,1]$.

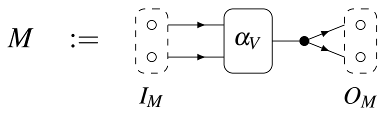
The following example shows a network containing two transducers T and U:



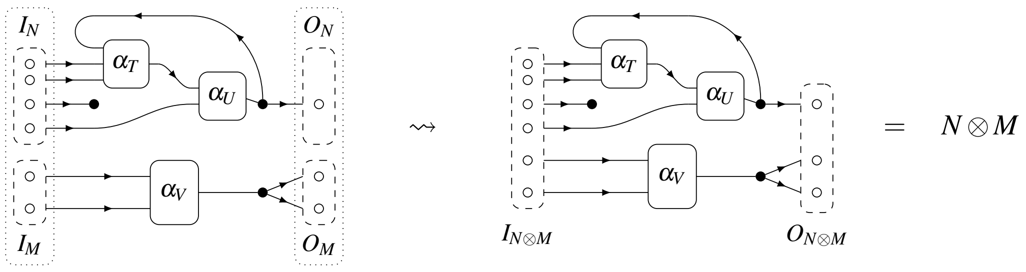
Building a network involves routing the outputs of some transducers into the inputs of others in such a way that each transducer's inputs are hooked up. We are allowed to copy signals by splitting wires and delete signals by capping wires, both indicated by black dots. Crucially, we allow for unconnected input and output wires to provide an interface to the outside world. We split the interface into an input side I which only has outgoing wires and and output side O which only has incoming wires. Both the input and output interfaces come with a fixed number of ports indicated by empty circles.

2.2 Composition

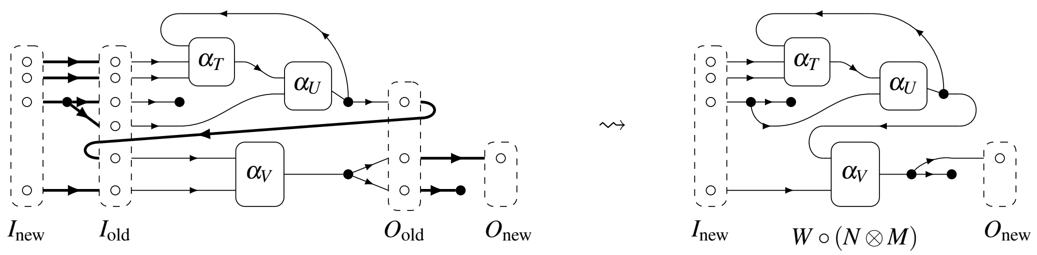
The key feature of open networks is that they can be composed using two basic operations. The first allows us to put two networks in parallel. For instance, given a second network



we can form the *parallel product* $N \otimes M$ by first stacking N above M and then combining their input and output interfaces:



The second operation, called *composition*, allows us to change interfaces while introducing new connections. For this, we must specify a wiring pattern W between the old interfaces $I_{\text{old}}, O_{\text{old}}$ and the new interfaces $I_{\text{new}}, O_{\text{new}}$. Each port in O_{new} must be connected to a port of O_{old}. Furthermore, each port in I_{old} must be connected to a either a port of I_{new} or a port of O_{old}. The second option allows for the creation of loops. During both processes we are allowed to copy and delete. The new network is then obtained by forgetting the old interfaces. The rules ensure that each transducer receives exactly one signal at each of its inputs. The composition process is illustrated on the network $N \otimes M$ where the new connections supplied by the wiring pattern W are highlighted in bold.



Complex networks can be built by just using these two operations. In particular, the operation of plugging the outputs of one network into the inputs of another follows as a special case: We first put two compatible networks A, B in parallel and then wire all of the outputs of A into the inputs of B by composition.

2.3 Dynamics

A PSN evolves in discrete time according to the dynamics of the transducers it contains. At each time-step, a transducer T uses the values present at its inputs to update its state using its activation map $f_T^\sharp \colon [0,1] \times [0,1]^{k_T} \rightsquigarrow [0,1]$. The wiring of the network dictates where T takes its k_T input values from. In this manner, all transducer activation levels are updated simultaneously. For a network containing n transducers and k external inputs, this yields a global update map $\mathsf{up}\colon [0,1]^n \times [0,1]^k \rightsquigarrow [0,1]^n$ describing how the current activation levels in the network, along with the external input levels to the network are converted into activation levels in the next time-step. Furthermore, the network structure induces a global output map $\mathsf{out}\colon [0,1]^n \to [0,1]^j$ describing which activation levels are surfaced at the j output ports. While activation and update maps may be probabilistic, the output map is deterministic.

To track a PSN's evolution over time, we iterate its update map $\mathsf{up}\colon [0,1]^n \times [0,1]^k \rightsquigarrow [0,1]^n$. During each application of the map, we must supply the current state vector $\boldsymbol{\alpha}(t) \in [0,1]^n$ and a vector of inputs $\mathbf{i}(t) \in [0,1]^k$. Hence, given an initial state $\boldsymbol{\alpha}(0)$, along with a sequence of input vectors $\mathbf{i}(0), \dots, \mathbf{i}(t)$, we can simulate the evolution of the system up to time $t+1$ by $\boldsymbol{\alpha}(s+1) = \mathsf{up}(\boldsymbol{\alpha}(s), \mathbf{i}(s))$ for $0 \leq s \leq t$.

2.4 Activation map families

The choice of activation maps determines the behavior of a PSN. In general, these could be any family of Markov kernels $f^\sharp \colon [0,1] \times [0,1]^k \rightsquigarrow [0,1]$. Most often, they will be based on a family of deterministic functions $\overline{f}_\vartheta^\sharp \colon [0,1] \times [0,1]^{k_\vartheta} \to [0,1]$ parametrized by a vector $\vartheta \in \mathbb{R}^m$ for some $m \in \mathbb{N}$. The number of inputs k_ϑ may vary across the family. Probability may be introduced by adding noise to allow for deviation from the ideal response. Changing the parameter values allows the PSN to learn from empirical data. Since Markov kernels express conditional probabilities, we may use Bayes' theorem to invert these and make probabilistic statements about what parameter values are most likely given the data.

2.4.1 Hill-type activation maps

This section describes activation maps suitable for modeling biological signaling and transcription networks. In signaling networks, proteins relay signals from the environment to the interior of the cell where they can affect appropriate responses. Signals are passed from one protein to another by chemical modification. Transcription networks regulate gene expression through proteins which bind promoter regions of DNA and activate or inhibit gene transcription. The expression of these regulatory proteins is itself controlled by the network. In both cases, networks can often be decomposed into a conserved set of network motifs, making them ideally suited for the compositional approach presented here. Alon gives clear and concise introductions to these topics in the short article [2] and excellent textbook [3].

Activation profiles in biological networks often have sigmoidal shapes described well by Hill functions (Figure 1, left). Usually, the interactions between different activators and inhibitors can be captured by boolean logical functions. Hyper-Hill functions combine these aspects by interpolating any function $\{0,1\}^m \to [0,1]$ defined on the corners of the hypercube while retaining Hill-type profiles.

Definition 2.1. For $n \in \mathbb{R}_{\geq 0}$ and $k \in [0,1]$, the normalized *Hill function* of degree n with threshold k is

$$h^{n,k}\colon [0,1] \to [0,1], \qquad x \mapsto \frac{x^n(1+k^n)}{k^n+x^n}.$$

The *Hill coefficient* n determines the strength of activation which is half maximal at k. ◊

Definition 2.2 (Wittmann et. al. [16])**.** Given a map $p : \{0,1\}^m \to [0,1]$, we extend it to

$$\bar{p}\colon [0,1]^m \to [0,1], \qquad (x_1,\ldots,x_m) \mapsto \sum_{b\in\{0,1\}^m} \left[p(b) \prod_{i=1}^{m} (b_i x_i + (1-b_i)(1-x_i)) \right].$$

For $\mathbf{n} \in \mathbb{R}^m_{\geq 0}$ and $\mathbf{k} \in [0,1]^m$, the *hyper-Hill function* associated to p with degrees $\mathbf{n}$ and thresholds $\mathbf{k}$ is

$$H_p^{\mathbf{n},\mathbf{k}}\colon [0,1]^m \to [0,1], \qquad H_p^{\mathbf{n},\mathbf{k}}(x) := \bar{p}(h^{n_1,k_1}(x_1),\ldots,h^{n_m,k_m}(x_m)).$$

While p describes the values on the corners $\{0,1\}^m$, $\mathbf{n}$ and $\mathbf{k}$ determine the shape along each input. ◊

Hyper-Hill functions capture many possible interactions between inputs. Figure 1 shows some typical profiles including one which is nearly indistinguishable from an experimental activation profile shown in [3]. The cost of this expressivity is that the parameter count grows exponentially with the number of inputs. Luckily, the number of regulators for a single component in transcription and signaling networks is quite low. For instance, in the bacteria *E. coli* most genes are regulated by 1-3 regulators [15]. If many regulators are needed, one can replace hyper-Hill functions by simpler maps.

The hyper-Hill functions can be used in two ways. First we may use them directly as the activation maps for transducers. In this case, the value of the hyper-Hill function directly represents the activity of the transducer after receiving a given set of inputs $f_T^\sharp(\alpha,\mathbf{i}) := H(\mathbf{i})$. Such PSNs form a continuous generalization of the boolean networks used for the study of genetic regulation [12].

Alternatively, we can view the hyper-Hill functions as describing the change in activity, rather than its absolute level. Normalizing a simple production-decay model[1] yields

$$\frac{\Delta\alpha}{\Delta t} = r(H(\mathbf{i}) - \alpha), \tag{1}$$

[1] Suppose production of A is regulated via some nonnegative input function $H(\mathbf{i})$ and assume a constant probability for a unit of A to be degraded. Then the change in A is given by $\Delta A/\Delta t = s \cdot H(\mathbf{i}) - r \cdot A$, where s is a production rate and r a degradation rate. The steady-state concentration is $A_0 = s \cdot H(\mathbf{i})/r$. Provided that $H(\mathbf{i})$ is not always 0, we may assume its maximal value is 1 by scaling s. Then the maximal steady state value is $A_{\max} = s/r$. Normalizing to $\alpha := A/A_{\max}$ yields the desired equation.

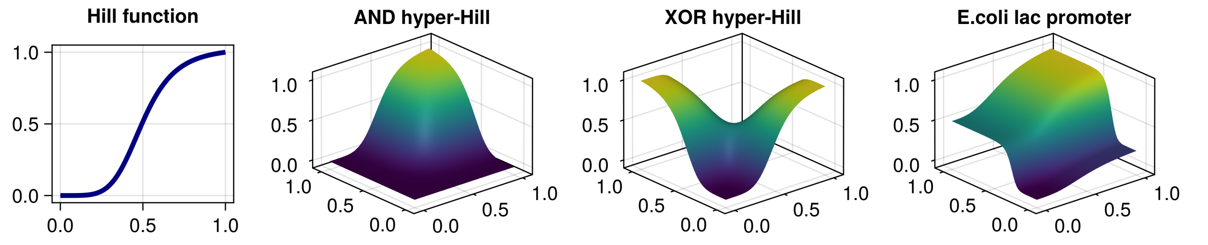


Figure 1: **Left**: Hill function with $n = 5$ and $k = 0.5$. **Middle**: Hyper-Hill functions expressing conjunction (AND) and exclusive or (XOR) with $n = (5,5)$ and $k = (0.5,0.5)$. **Right**: Hyper-hill profile emulating empirical *lac* promoter activity in *E. Coli* for inputs cAMP and IPTG shown in Alon [3], Figure 1.8. The parameters are set to $p = (0,0.2,0.5,1)$, $n = (3,9)$, and $k = (0.5,0.3)$. Plot made with Julia [4] using Makie [6].

where $r \in [0,1]$ is a rate parameter describing responsiveness. This translates into the activation map $f^\sharp_T(\alpha,\mathbf{i}) := \alpha + r(H(\mathbf{i}) - \alpha)$ which reduces to the case above when $r = 1$. Since (1) has steady-state $\alpha = H(\mathbf{i})$ for constant $\mathbf{i}$ and $r > 0$, we can view the case when $r = 1$ as a steady-state description.

2.4.2 Neural net-like activation maps

Neural networks can also be described as PSNs. Ridge activation maps are parametrized functions of the form $f^\sharp_{\mathbf{w}}(\alpha, i_1,\dots,i_n) := g(b + \sum_{l=1}^n w_l \cdot i_l)$, where $\mathbf{w} \in \mathbb{R}^n$ is a vector of weights, $b \in \mathbb{R}$ is the bias, and $g\colon \mathbb{R} \to [0,1]$ is an activation function. A standard choice for g is the logistic function $g(x) := (1+e^{-x})^{-1}$. Alternatively, radial activation maps $f^\sharp_{\mathbf{c}}(\alpha,\mathbf{i}) := h(\|\mathbf{i}-\mathbf{c}\|)$ that measure the distance of an input vector $\mathbf{i} = (i_1,\dots,i_n)$ to a center $\mathbf{c} \in \mathbb{R}^n$ are used. Here h could be the bump function $h(x) := \exp(-x^2/2\sigma^2)$.

One can model the detailed dynamics of such neural nets by letting the activation maps describe the change in activation rather than its absolute level. Moreover, both ridge and radial activation maps can represent simplified interactions in biological networks by choosing Hill-type maps for g and h.

2.5 Additional wiring elements

It is possible to augment the wiring patterns with additional elements aside from copy and delete. In general, these can be any Markov kernel $w^\sharp\colon [0,1]^n \rightsquigarrow [0,1]^m$. Useful deterministic functions include max, min, and the extended boolean functions $\neg(x) := 1-x$, $\wedge(x,y) := xy$ and $\vee(x,y) := x+y-xy$.

Probabilistic maps provide even greater expressiveness. For instance, probabilistic switches take two inputs x_1, x_2 and output x_1 with probability p and x_2 with probability $1-p$. Such switches can encode stochastic behavior in our system where a signal might be sent to a different place at each time-step. Alternatively, such switches can express our uncertainty in how the wires are connected in the network. The parameter p can then be inferred based on measured data to give the most plausible wiring.

3 PSNs as lenses

This section formalizes the ideas presented above by describing PSNs as lenses in the Markov category $\mathsf{BorelStoch}$ of standard Borel spaces and Markov kernels. Lenses can describe both the transducers and the wiring pattern of a PSN. Moreover, lenses form a symmetric monoidal category whose operations implement the composition and parallel product for open networks described above.

3.1 Markov kernels and Markov categories

We first describe Markov kernels and the category they inhabit. The following is standard, see [5] or [7].

Definition 3.1. A *measurable space* (X,Σ_X) is a set X equipped with a σ-algebra Σ_X. Given measurable spaces (X,Σ_X) and (Y,Σ_Y), a *Markov kernel* $f\colon X \rightsquigarrow Y$ is a function $f\colon \Sigma_Y \times X \to [0,1]$ such that

(i) for each $x \in X$, $f(-\,|\,x)$ is a probability measure on (Y,Σ_Y),

(ii) for each $B \in \Sigma_Y$, $f(B\,|\,=)\colon X \to [0,1]$ is Σ_X-measurable.

We think of $f(B\,|\,x)$ as the probability that $y \in B$, given x.

Markov kernels $f\colon X \rightsquigarrow Y$ and $g\colon Y \rightsquigarrow Z$ can be composed. For any $C \in \Sigma_Z$ and $x \in X$,

$$(g\circ f)(C\,|\,x) := \int_{y\in Y} g(C\,|\,y)\, df(-\,|\,x).$$

The Markov kernel $\mathrm{id}_X\colon X \rightsquigarrow X$ given by $\mathrm{id}_X(A\,|\,x) := \delta_x(A) := \mathbb{1}(x\in A)$ serves as the identity. Composition is associative by the monotone convergence theorem. We denote the *category of measurable spaces and Markov kernels* by Stoch. The full subcategory of standard Borel spaces[2] is denoted $\mathsf{BorelStoch}$. $\diamondsuit$

Definition 3.2. Stoch has the structure of a symmetric monoidal category with $(X,\Sigma_X)\otimes(Y,\Sigma_Y) := (X\times Y, \Sigma_X\otimes\Sigma_Y)$, where $\Sigma_X\otimes\Sigma_Y$ is the product σ-algebra generated by the set of rectangles $\{A\times B \,|\, A\in\Sigma_X, B\in\Sigma_Y\}$. The singleton space $I := \{*\}$ acts as monoidal unit. On maps $f_1\colon X_1 \rightsquigarrow Y_1$ and $f_2\colon X_2 \rightsquigarrow Y_2$, the product $f_1\otimes f_2$ is defined on rectangles by

$$(f_1\otimes f_2)(B_1\times B_2\,|\,x_1,x_2) := f_1(B_1\,|\,x_1)f_2(B_2\,|\,x_2).$$

Stoch has two further pieces of useful structure. The *copy* map $\mathsf{cp}_X\colon X \rightsquigarrow X\otimes X$ is given on rectangles by $\mathsf{cp}_X(A\times B\,|\,x) := \delta_x(A)\delta_x(B)$, while the *delete* map $\mathsf{del}_X\colon X \rightsquigarrow I$ is given by $\mathsf{del}_X(A\,|\,x) := \delta_*(A)$. $\diamondsuit$

The following notion of conditional density is useful for explicit computations.

Definition 3.3. Let $f\colon X \rightsquigarrow Y$ be a Markov kernel. A *(conditional) density* $\varphi(y\,|\,x)$ for f with respect to a measure μ_f on Y is a measurable function $\varphi\colon Y\times X \to \mathbb{R}_{\geq 0}$ satisfying $f(B\,|\,x) = \int_B \varphi(y\,|\,x)\,d\mu_f$ for all $B\in\Sigma_Y$. In this case, $\int h(y)\,df(-\,|\,x) = \int h(y)\varphi(y\,|\,x)\,d\mu_f$ for any integrable function $h\colon Y\to\mathbb{R}$. $\diamondsuit$

Lemma 3.4. *Let $\mathscr{F}$ denote the class of Markov kernels in* Stoch *that have densities with respect to σ-finite measures. Then $\mathscr{F}$ is closed under the following operations:*

(i) Pre-composition with any Markov kernel,

(ii) post-composition with elements of $\mathscr{F}$,

(iii) monoidal products with elements of $\mathscr{F}$,

(iv) post-composition with lifts of injective measurable functions whose inverse is measurable,

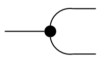
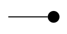
(v) post-composition with del.

Proof. Points (i)-(iii) follow by definition, Tonelli's theorem, and checking measurability of the density. (iv): Let g be a measurable injective map and $f\in\mathscr{F}$. Then $(\delta_g\circ f)(C\,|\,x) = \int_Y \delta_{g(y)}(C)\varphi(y\,|\,x)\,d\mu_f$. Writing $\mathrm{id} = g^{-1}\circ g$ yields $\int_Y \delta_{g(y)}(C)\varphi(g^{-1}\circ g(y)\,|\,x)\,d\mu_f = \int_{g(Y)}\delta_z(C)\varphi(g^{-1}(z)\,|\,x)\,dg_*\mu_f$ by change of variables. Rewriting as $\int_C \mathbb{1}_{g(Y)}\varphi(g^{-1}(z)\,|\,x)\,dg_*\mu_f$ shows $\delta_g\circ f$ has a x,z-measurable density since g^{-1} is measurable. The pushforward $g_*\mu_f$ is σ-finite since g preserves measurable sets: If Y_n are measurable sets of finite μ_f-measure covering Y, then $Z_n := g(Y_n)\cup(Z\setminus g(Y))$ are measurable sets of finite $g_*\mu_f$-measure covering Z. (v): Note $(\mathsf{del}_Y\circ f)(C\,|\,x) = \int_Y \delta_*(C)\varphi(y\,|\,x)\,d\mu_f = \int_C\int_Y \varphi(y\,|\,x)\,d\mu_f\,d\delta_*(-)$. $\square$

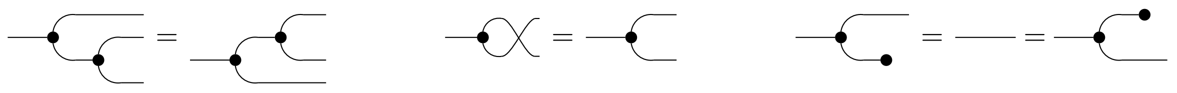
[2]A measurable space (X,Σ_X) is standard Borel if it is isomorphic to a separable complete metric space with the Borel σ-algebra. In particular, the closed unit interval $[0,1]$ with its Borel σ-algebra is standard Borel.

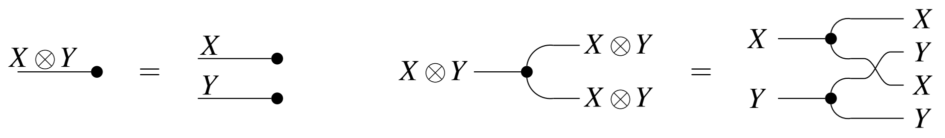
Stoch provides the prototypical example of a category in which one can do probability theory [7]. Markov kernels express conditional probabilities and specialize to a variety of standard concepts. Distributions on X can be represented by maps $I \rightsquigarrow X$, while random variables $f\colon X \to Y$ can be lifted to Markov kernels. Moreover, $\otimes$ and del can express joint distributions and marginalization. Abstracting these structures yields the definition of a Markov category.

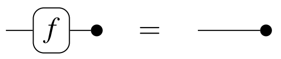
Definition 3.5 (Fritz [7], Def. 2.1)**.** A *Markov category* C is a symmetric monoidal category $(C, \otimes, I)$ where each $X \in \mathsf{C}$ is equipped with copy $\mathsf{cp}_X\colon X \to X \otimes X$ and delete $\mathsf{del}_X\colon X \to I$, in string diagrams

$\mathsf{cp}_X =$  and $\mathsf{del}_X =$ 

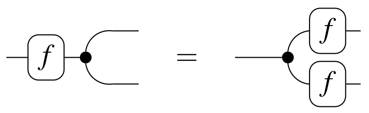
satisfying the commutative comonoid equations, compatibility with $\otimes$, and naturality of del:





 ◊

Definition 3.6. A map f in a Markov category is called *deterministic* if



This means that copying the output of f is the same as generating two separate outputs using f. ◊

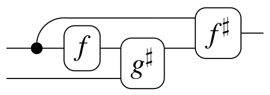
In Stoch, determinism states $f(B \cap C|x) = f(B|x)f(C|x)$ for all measurable B and C. Thus any measurable function $\bar{f}\colon X \to Y$ induces a deterministic map $f\colon X \rightsquigarrow Y$ by $f(B|x) := \delta_{\bar{f}(x)}(A)$. In BorelStoch every deterministic map is of this form, as explained in Fritz [7], Examples 10.4-10.5.

3.2 Lenses in a Markov category

We follow Myers and Spivak by using lenses to organize our data. We highly recommend Myers [11] for intuition. The following instantiates Definition 2.9 of [13] in the setting of Markov categories.

Definition 3.7. A *lens* $\binom{f^\sharp}{f}\colon \binom{X}{C} \leftrightarrows \binom{Y}{D}$ in a Markov category $(\mathsf{C}, \otimes, I)$ consists of objects X, C, Y, D, along with a *passback* map $f^\sharp\colon C \otimes Y \to X$ and a deterministic *passforward* map $f\colon C \to D$. ◊

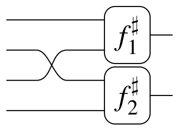
Definition 3.8. Given lenses $\binom{f^\sharp}{f}\colon \binom{X}{C} \leftrightarrows \binom{Y}{D}$ and $\binom{g^\sharp}{g}\colon \binom{Y}{D} \leftrightarrows \binom{Z}{E}$, their *composite* $\binom{g^\sharp}{g} \circ \binom{f^\sharp}{f}\colon \binom{X}{C} \leftrightarrows \binom{Z}{E}$ has passforward map $g \circ f$ and passback map



This defines a category $\mathsf{Lens}(\mathsf{C})$ with identities $\binom{\pi_2}{\mathrm{id}}$, where $\pi_2 := \lambda \circ (\mathsf{del} \otimes \mathrm{id})$. ◊

Proof Sketch. Unitality and associativity of the composition can be shown using string diagrams and the Markov category axioms. Determinism of the passforward is required for associativity. □

Definition 3.9. Given lenses $\binom{f_1^\sharp}{f_1}\colon \binom{X_1}{C_1} \leftrightarrows \binom{Y_1}{D_1}$ and $\binom{f_2^\sharp}{f_2}\colon \binom{X_2}{C_2} \leftrightarrows \binom{Y_2}{D_2}$, we define their *parallel product* $\binom{f_1^\sharp}{f_1} \otimes \binom{f_2^\sharp}{f_2}\colon \binom{X_1 \otimes X_2}{C_1 \otimes C_2} \leftrightarrows \binom{Y_1 \otimes Y_2}{D_1 \otimes D_2}$ to have passforward map $f_1 \otimes f_2$ and passback map:



◊

Proposition 3.10. *The category* $\mathsf{Lens}(\mathsf{C})$ *is symmetric monoidal with the parallel product defined above.*

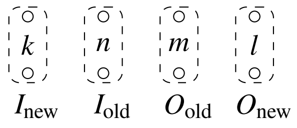
Proof Sketch. This follows indirectly from a monoidal Grothendieck construction [10], as sketched in Section 3.3 of [13] for strict C. We sketch a direct proof. The monoidal unit is $\binom{I}{I}$. Functoriality of $\otimes$ is checked using string diagrams. The coherence isomorphisms are defined as $\binom{\triangle^{-1} \circ \pi_2}{\triangle}$ where $\triangle$ are λ, ρ, α, or the symmetry swap. Naturality is again checked using string diagrams. Checking the triangle, pentagon and hexagon identities is facilitated by noticing that $\binom{f_1 \circ \pi_2}{g_1} \circ \binom{f_2 \circ \pi_2}{g_2} = \binom{f_2 \circ f_1 \circ \pi_2}{g_1 \circ g_2}$ and $\binom{f_1 \circ \pi_2}{g_1} \otimes \binom{f_2 \circ \pi_2}{g_2} = \binom{(f_1 \otimes f_2) \circ \pi_2}{g_1 \otimes g_2}$. This allows the requisite identities to be reduced to those of C. □

3.3 Probabilistic signaling networks

We can now describe transducers and wiring patterns as lenses in $\mathsf{BorelStoch}$.

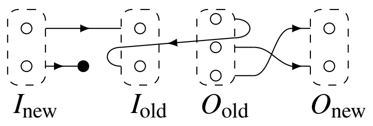
Definition 3.11. Let F be a family of Markov kernels $f_j^\sharp\colon [0,1] \times [0,1]^{k_j} \rightsquigarrow [0,1]$. A *transducer* with k inputs of *type* F is a lens in $\mathsf{BorelStoch}$ of the form $\binom{f^\sharp}{\mathrm{id}}\colon \binom{[0,1]}{[0,1]} \leftrightarrows \binom{[0,1]^k}{[0,1]}$ whose passback map lies in F and whose passforward is the identity. We think of $f^\sharp\colon [0,1] \times [0,1]^k \rightsquigarrow [0,1]$ as an activation map describing the probability $f^\sharp(A \mid \alpha(t), i_1(t), \dots, i_k(t))$ that the activation level $\alpha(t+1)$ lies in A, given that the current level is $\alpha(t)$ and inputs $i_1(t), \dots, i_k(t)$ were received. We will typically describe the family F using conditional densities w.r.t. σ-finite measures. This covers all usual distributions. ◊

Definition 3.12. A *wiring pattern* is a lens in $\mathsf{BorelStoch}$ of the form $\binom{w^\sharp}{w}\colon \binom{[0,1]^n}{[0,1]^m} \leftrightarrows \binom{[0,1]^k}{[0,1]^l}$ whose passback and passforward maps are constructed by composing and tensoring id, cp, del, the symmetry swap, and coherence isomorphisms. The numbers n, m, k, l describe the number of ports in the inner and outer interfaces:



Moreover, for a set A of Markov kernels of the form $a_j^\sharp\colon [0,1]^{n_j} \rightsquigarrow [0,1]^{m_j}$, we define an *A-augmented wiring pattern* as a wiring pattern whose passback map may additionally use the maps in A. ◊

Example 3.13. Consider the wiring pattern:



Based on the number of ports, it is a lens $\binom{w^\sharp}{w}\colon \binom{[0,1]^2}{[0,1]^3} \leftrightarrows \binom{[0,1]^2}{[0,1]^2}$. The map $w\colon [0,1]^3 \rightsquigarrow [0,1]^2$ is determined by the wires from O_{old} to O_{new}. It is given by lifting the function $(o_1, o_2, o_3) \mapsto (o_3, o_2)$ to a Markov kernel. Up to coherence isomorphisms, this may me expressed as $w = \mathsf{swap} \circ (\mathsf{del} \otimes \mathsf{id} \otimes \mathsf{id})$. The map $w^\sharp\colon [0,1]^3 \times [0,1]^2 \rightsquigarrow [0,1]^2$ is determined by the wires from O_{old} and I_{new} to I_{old}. It is given by lifting the function $((o_1, o_2, o_3), (i_1, i_2)) \mapsto (i_1, o_1)$, which may be written as $w^\sharp = (\mathsf{id} \otimes \mathsf{del}) \otimes (\mathsf{id} \otimes \mathsf{del} \otimes \mathsf{del})$. //

Combining transducers and wiring patterns gives us probabilistic signaling networks.

Definition 3.14. A *probabilistic signaling network* (PSN) of type (F,A) is a lens of the form $W \circ (P_1 \otimes \ldots \otimes P_n)$ where the P_i are transducers of type F and W is an A-augmented wiring pattern. We denote its passback map up and its passforward out. Hence, a PSN is a lens

$$\begin{pmatrix}\mathsf{up}\\ \mathsf{out}\end{pmatrix} : \begin{pmatrix}[0,1]^n\\ [0,1]^n\end{pmatrix} \leftrightarrows \begin{pmatrix}[0,1]^k\\ [0,1]^j\end{pmatrix}$$

where n is the number of transducers, k the number of inputs, and j the number of outputs. ◊

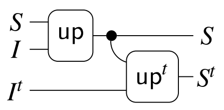
Lemma 3.15. *PSNs are closed under parallel products and composition with wiring patterns. This closure property restricts to PSNs whose update map has a density w.r.t. a σ-finite measure.*

Proof. Wiring patterns are closed under composition and parallel products since both operations only use cp, swap and coherence isomorphisms. Hence $W' \circ W \circ (P_1 \otimes \ldots \otimes P_n)$ and $[W \circ (P_1 \otimes \ldots \otimes P_n)] \otimes [W' \circ (P'_1 \otimes \ldots \otimes P'_m)] = [W \otimes W'] \circ [(P_1 \otimes \ldots \otimes P_n) \otimes (P'_1 \otimes \ldots \otimes P'_m)]$ are both PNSs. The second assertion follows from Lemma 3.4 by examining the definition of the two operations. □

The dynamics of a PSN are obtained by iterating its update map.

Definition 3.16. Given a PSN $\begin{pmatrix}\mathsf{up}\\ \mathsf{out}\end{pmatrix} : \begin{pmatrix}[0,1]^n\\ [0,1]^n\end{pmatrix} \leftrightarrows \begin{pmatrix}[0,1]^k\\ [0,1]^j\end{pmatrix}$ denote the state space $S := [0,1]^n$, the input space $I := [0,1]^k$ and the output space $O := [0,1]^j$. Define the *t-fold update map* inductively by $\mathsf{up}^1 := \mathsf{up}$ and

$\mathsf{up}^{t+1} \quad :=$



We interpret $\mathsf{up}^{t+1} : S \times I^{t+1} \rightsquigarrow S^{t+1}$ as the probability of the sequence $\boldsymbol{\alpha}(1), \ldots, \boldsymbol{\alpha}(t+1)$, given initial state $\boldsymbol{\alpha}(0)$ and input sequence $\mathbf{i}(0), \ldots, \mathbf{i}(t)$. We compose up^{t+1} with $\bigotimes_{s=0}^{t} \mathsf{out}$ to obtain the *input-output* map $\mathsf{io}^{t+1} : S \times I^{t+1} \rightsquigarrow O^{t+1}$, describing the analogous conditional probabilities for output sequences. ◊

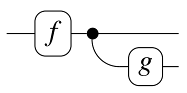
Given probabilities $\mathbb{P}[\boldsymbol{\alpha}(s+1) \mid \boldsymbol{\alpha}(s), \mathbf{i}(s)]$ for $0 \le s \le t$, we can calculate

$$\mathbb{P}[\boldsymbol{\alpha}(1), \ldots, \boldsymbol{\alpha}(t+1) \mid \boldsymbol{\alpha}(0), \mathbf{i}(0), \ldots, \mathbf{i}(t)] = \textstyle\prod_{s=0}^{t} \mathbb{P}[\boldsymbol{\alpha}(s+1) \mid \boldsymbol{\alpha}(s), \mathbf{i}(s)]$$

using the product rule and the fact that $\mathbb{P}[\boldsymbol{\alpha}(s+1) \mid \boldsymbol{\alpha}(s), \mathbf{i}(s)] = \mathbb{P}[\boldsymbol{\alpha}(s+1) \mid \boldsymbol{\alpha}(s), \mathbf{i}(0), \ldots, \mathbf{i}(t)]$ since $\boldsymbol{\alpha}(s+1)$ is independent of $\mathbf{i}(r)$ for $r \neq s$, given $\boldsymbol{\alpha}(s)$ and $\mathbf{i}(s)$. The next lemma shows that up^{t+1} generalizes this idea and that iterating updates preserves densities w.r.t. σ-finite measures.

Lemma 3.17 (Product rule for densities)**.** *In* Stoch*, the composition*

$h \quad :=$



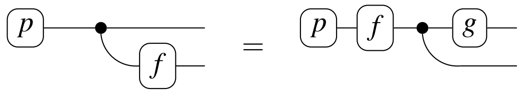
is given by $h(A \times B \mid x) = \int_B g(A \mid y)\, df(- \mid x)$. If f and g have densities $\varphi(y \mid x)$ and $\psi(z \mid y)$ w.r.t σ-finite measures μ_f and μ_g, then h has density $\varphi(y \mid x)\psi(z \mid y)$ w.r.t. the product measure $\mu_f \otimes \mu_g$.

3.4 Bayesian inference for PSNs

Bayes' theorem allows us to invert probabilities $\mathbb{P}[y \mid x]$ to $\mathbb{P}[x \mid y]$, provided we supply a prior $\mathbb{P}[x]$:

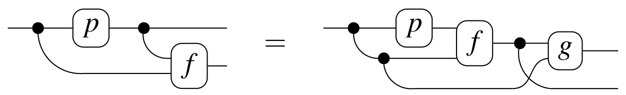
$$\mathbb{P}[x \mid y] = \frac{\mathbb{P}[y \mid x]\, \mathbb{P}[x]}{\sum_x \mathbb{P}[y \mid x]\, \mathbb{P}[x]}.$$

In Markov categories, we convert maps $f\colon X \to Y$ and $p\colon I \to X$ to a Bayesian inverse $g\colon Y \to X$, where



(2)

For discrete distributions in Stoch this precisely expresses $\mathbb{P}[x \mid y] \sum_x \mathbb{P}[y \mid x]\mathbb{P}[x] = \mathbb{P}[y \mid x]\mathbb{P}[x]$ by the product rule for densities and the definition of composition. In $\mathsf{BorelStoch}$, Bayesian inverses satisfying (2) always exist [5]. We need a more general inversion formula, where all maps are conditioned on an additional input. Given maps $p\colon X \to Y$ and $f\colon Y \otimes X \to Z$, we want an inverse $g\colon Z \otimes X \to Y$ satisfying



(3)

The Bayesian inverse g can be explicitly calculated when f and p have nice conditional densities.

Proposition 3.18. *Suppose that p and f have densities $\pi(y \mid x)$ and $\varphi(z \mid y,x)$ with respect to σ-finite measures μ_p and μ_f. Let $N := \{(x,z) \mid \int_Y \varphi(z \mid y,x)\pi(y \mid x)\, d\mu_p \in (0,\infty)\}$. Define*

$$g(B \mid z,x) := \frac{\int_B \varphi(z \mid y,x)\pi(y \mid x)\, d\mu_p}{\int_Y \varphi(z \mid y,x)\pi(y \mid x)\, d\mu_p} \qquad \forall (x,z) \in N$$

and $g(B \mid z,x) := 0$ otherwise. Then g satisfies equation (3) and is unique on N μ_f-almost everywhere.

Proof. Note that $g(- \mid z,x)$ is a probability measure. Since $\varphi(z|y,x)\pi(y|x)$ is x,y,z-measurable, $I(x,z) := \int_Y \varphi(z \mid y,x)\pi(y \mid x)\, d\mu_p$ is z,x-measurable by Tonelli's theorem. Since $g(B \mid z,x)$ is z,x-measurable on the measurable N and N^{c}, it is z,x-measurable. Equation (3) expresses that for all $x \in X$, $B \in \Sigma_Y$ and $C \in \Sigma_Z$,

$$\int_C \int_B \varphi(z \mid y,x)\pi(y \mid x)\, d\mu_p\, d\mu_f = \int_C g(B \mid z,x) \int_Y \varphi(z \mid y,x)\pi(y \mid x)\, d\mu_p\, d\mu_f. \tag{4}$$

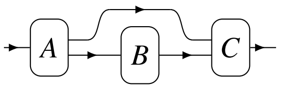
The left expression is bounded by $\int_Z I(x,z)\, d\mu_f = 1$ by Tonelli. Hence $Z_x^\infty := \{z \mid \int_Y I(x,z) = \infty\}$ has μ_f measure 0. For $l_x(z) := \int_B \varphi(z \mid y,x)\pi(y \mid x)\, d\mu_p$ and $r_x(z) := g(B \mid z,x)I(x,z)$, comparing the integrals on the sets $\{l_x(z) < r_x(z)\}$ and $\{l_x(z) > r_x(z)\}$ shows that (4) holds iff $l_x(z) = r_x(z)$ (μ_f-a.e.). On N, we can solve for g to obtain the claimed formula and uniqueness. When $I(x,z) = 0,\infty$, any value works. $\square$

We now describe inference on a PSN N with state space $S := [0,1]^n$, input space $I := [0,1]^k$ and output space $O := [0,1]^j$. Call a family of Markov kernels $f_\theta\colon X \rightsquigarrow Y$ in $\mathsf{BorelStoch}$ nicely parametrized by θ if they describe a kernel $f\colon \Theta \times X \rightsquigarrow Y$ having a density w.r.t. a σ-finite measure. Suppose the activation map of each transducer T_i in N is nicely parametrized by ϑ_{T_i}. Viewing the ϑ_{T_i} as additional inputs and carrying them through the construction of N, the update map of N becomes nicely parametrized by $\theta := (\vartheta_{T_1}, \ldots, \vartheta_{T_n})$ (Lemma 3.15). By copying the parameter values at each time-step, up_θ^{t+1} becomes nicely parametrized by θ (Lemma 3.17). Constructing the input-output map yields a kernel $\mathsf{io}^{t+1}\colon \Theta \times S \times I^t \rightsquigarrow O^t$ describing the input-output relationship induced by N, given $\theta \in \Theta$. Moreover, io^{t+1} will have a density[3] w.r.t a σ-finite measure which can be explicitly computed. Fixing a prior $p\colon S \times I^t \rightsquigarrow \Theta$ (which will typically not depend on S or I^t), we apply Proposition 3.18 to get an explicit map $\mathsf{post}\colon O^t \times S \times I^t \rightsquigarrow \Theta$, describing the posterior conditional distribution of θ, given output sequence $\mathbf{o}(1), \ldots, \mathbf{o}(t+1)$, initial state $\boldsymbol{\alpha}(0)$, and input sequence $\mathbf{i}(0), \ldots, \mathbf{i}(t)$. In most cases, it is not possible to obtain closed form solutions for the posterior due to the integral. Instead, computational techniques must be used as described in [8, 9]. Similar manipulations allow us to infer other quantities of interest as well.

[3]Observe that the construction of the iterated update map up^{t+1} gives us a density w.r.t. the product $\bigotimes_{s=0}^t (\mu_{T_1} \otimes \ldots \otimes \mu_{T_n})$. We can rewrite $\bigotimes_{s=0}^t \mathsf{out} = r \circ \bigotimes_{s=0}^t (d_1 \otimes \ldots \otimes d_n)$, where d_j is del or id and r is a remainder not containing delete. Composing $\bigotimes_{s=0}^t (d_1 \otimes \ldots \otimes d_n)$ with up^{t+1} has the effect of marginalizing out components where $d_j = \mathsf{del}$. This retains a density. Composing with r is unproblematic since it is made of lifts of injective measurable maps with measurable inverses.

4 Inference of a feed-forward network motif

To demonstrate the practical utility of PSNs we apply them to infer the logical structure of a network motif called the feed-forward loop (FFL), defined by having the following connectivity structure:



The *E. coli* transcription network contains 42 FFLs, whereas random networks with the same mean connectivity only contain 2 on average [3]. This suggest that FFLs perform a useful function. We will study the *coherent type 1 FFL* (C1-FFL) whose arrows are all activating. These makes up around 35% of the FFLs occurring in *E. coli* and yeast [3]. To fix the logical structure of the C1-FFL we must say how inputs to C are combined. Let c_{ab} denote the response of C for boolean inputs $a \in A$ and $b \in B$. We will assume that the inputs are combined by $a \wedge b$, meaning that $c_{ab} = 1$ if $a = b = 1$ and zero otherwise.

We are now able to simulate the temporal evolution of the $\wedge$-type C1-FFL by considering it as a PSN with the hyper-Hill dynamics introduced in Section 2.4.1. Explicitly, we set

$$\begin{aligned}
A(t+1) &= A(t) + r_A \cdot (H_{\mathbf{a}}^{n_A,k_A}(i(t)) - A(t)) \\
B(t+1) &= B(t) + r_B \cdot (H_{\mathbf{b}}^{n_B,k_B}(A(t)) - B(t)) \\
C(t+1) &= C(t) + r_C \cdot (H_{\mathbf{c}}^{n_C,k_C}(A(t),B(t)) - C(t))
\end{aligned}$$

where $\mathbf{a} = (a_0, a_1)$, $\mathbf{b} = (b_0, b_1)$ and $\mathbf{c} = (c_{00}, c_{10}, c_{01}, c_{11})$ are the coefficients describing the level of activation on the corners of the hypercube. For a $\wedge$-type C1-FFL we set $\mathbf{a} := \mathbf{b} := (0,1)$ and $\mathbf{c} := (0,0,0,1)$. The connectivity of the network has been encoded in the way we chose the input variables for each hyper-Hill function. Setting explicit values for the remaining parameters $r_A := r_B := r_C := 0.05$, $n_A := n_B := 5$, $n_C := (5,5)$, $k_A := k_B := 0.5$ and $k_C := (0.5, 0.5)$ allows us to simulate the system for any choice of initial values and input sequence. For $A(0) = B(0) = C(0) = 0$ and constant input sequence $i(t) \equiv 1$, the simulated activities in the top left panel of Figure 2 show a delayed response in C. Hence, the $\wedge$-type C1-FFL can filter out transient inputs making the network robust to noise.

Forward modeling allows us to simulate the behavior of a system once we fix its parameters. In practice, we must solve the inverse problem: Given measured behavior, we must determine what system is most likely to have produced it. Let us suppose that we are dealing with a feed-forward motif with unknown logical structure. To elucidate it, we perform an experiment by exposing the system to a constant input signal $i(t) \equiv 1$ and measuring the output $C(t)$. We simulate such measurements by adding normal noise ($\sigma = 0.08$) to the simulated values of the $\wedge$-type C1-FFL (Figure 2, top middle).

To apply inference, we modify our deterministic model to the following probabilistic model:

$$\begin{aligned}
A(t+1) &\sim N(\mu_A(t), \sigma_A), & \mu_A(t) &:= A(t) + r_A \cdot (H_{\mathbf{a}}^{n_A,k_A}(i(t)) - A(t)) \\
B(t+1) &\sim N(\mu_B(t), \sigma_B), & \mu_B(t) &:= B(t) + r_B \cdot (H_{\mathbf{b}}^{n_B,k_B}(A(t)) - B(t)) \\
C(t+1) &\sim N(\mu_C(t), \sigma_C), & \mu_C(t) &:= C(t) + r_C \cdot (H_{\mathbf{c}}^{n_C,k_C}(A(t),B(t)) - C(t))
\end{aligned}$$

We must decide which parameters to hold fixed and which to infer. The experiment with constant input sequence does not give us much information on A, so we will assume that $\mathbf{a} = (0,1)$, as in the generating FFL. For simplicity we will also fix n_A, n_B, n_C and k_A, k_B, k_C to their true values. We set the standard deviations $\sigma_A = \sigma_B = 0.01$ and $\sigma_C = 0.1$. The values for σ_A and σ_B express our uncertainty in the dynamical equations while σ_C additionally incorporates the measurement error whose magnitude we could experimentally determine by repeatedly measuring a reference value.

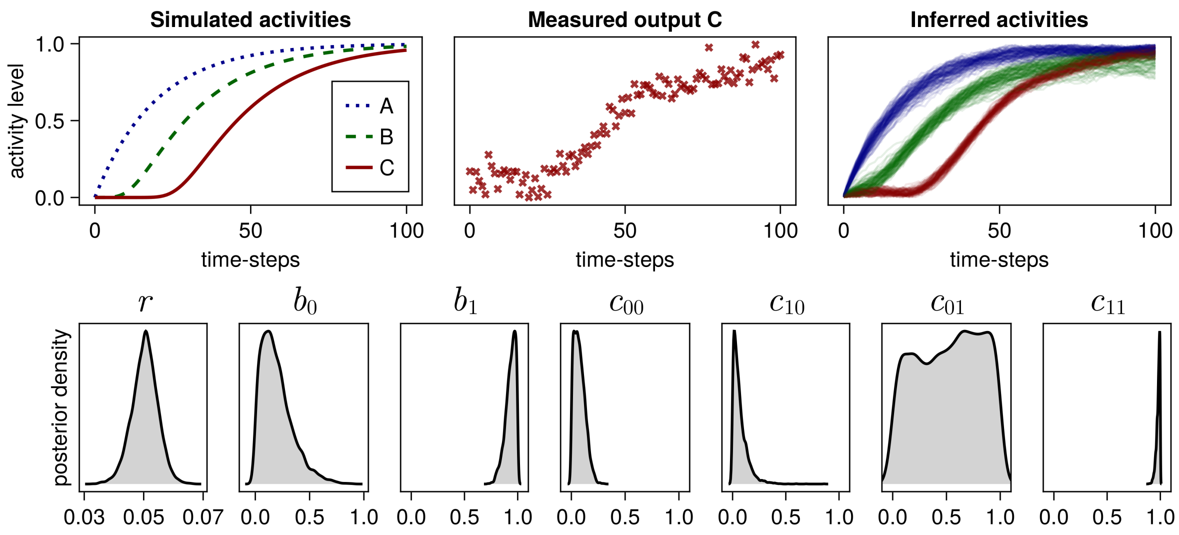


Figure 2: **Top left**: Simulated dynamics of $\wedge$-type C1-FFL with parameter values indicated in the main text. **Top middle**: Simulated measurements of $C(t)$ using normal noise ($\sigma = 0.08$). **Top right**: Inferred activities of A, B, and C. **Bottom**: Inferred posterior densities for the free parameters. Plot made with Julia [4] using Makie [6].

On the other hand, we will leave the coefficients **b** and **c** which determine the logical structure of the FFL completely free. This covers 64 different FFL structures, if we restrict to boolean values. We will also infer the responsiveness, but assume that all transducers share the same value $r := r_A = r_B = r_C$. Finally, we must set priors for all the unknown parameters. In our case, all the parameters take values in $[0,1]$. We will express our complete lack of knowledge about $\mathbf{b},\mathbf{c}$ and r by setting uniform priors. In our model, $A(t), B(t), C(t)$ are also unknown parameters. For $t \geq 1$ their likelihoods are given by the modeling equations. For $t = 0$ we set priors $N(0, 0.01)$ to indicate that the values start at around 0.

These specification can now be translated into a probabilistic programming language that supports Bayesian inference. We will use STAN [14] through its Julia interface Stan.jl [1] since it is widely known and well documented. Future work will develop software that generates STAN code automatically based on visual specification of a PSN's structure. STAN uses Hamiltonian Markov chain Monte Carlo (HMC) to perform inference. Because this algorithm uses gradients, it is important for all likelihoods to be smooth functions, once the known parameters are fixed.

The HMC inference takes around a minute to run on a MacBook Air. The results are displayed in Figure 2. We observe that the mode of the posterior distribution of the response parameter r has the true value of 0.05. The posterior distributions for **b** indicate that b_0 most likely lies below 0.5 and b_1 is very close to 1 which agrees with the true values. Similarly, c_{00}, c_{10} and c_{11} all are peaked around their true values. The only parameter we could not identify is c_{10} whose posterior is uniformly distributed.

In summary, assuming the parameter values we fixed, if we see the measured delayed output response, we can conclude that A activates B (since $b_0 \leq b_1$) and that A and B together activate C (since $c_{11} \approx 1$), while only A without B is insufficient to activate C (since $c_{10} \approx 0$). However, we are unable to make any claims about what would happen if B where present without A (since c_{01} uniform). It is astonishing that we can recover the parameters so well using a single noisy experiment. These results could be improved and additional parameters inferred by incorporating additional experiments with different input sequences.

References

[1] *Stan.jl Library*. Available at `https://github.com/StanJulia/Stan.jl`.

[2] Uri Alon (2003): *Biological Networks: The Tinkerer as an Engineer*. Science 301(5641), pp. 1866–1867, doi:10.1126/science.1089072.

[3] Uri Alon (2019): *An Introduction to Systems Biology*. Taylor & Francis Inc.

[4] Jeff Bezanson, Alan Edelman, Stefan Karpinski & Viral B. Shah (2017): *Julia: A Fresh Approach to Numerical Computing*. SIAM Review 59(1), pp. 65–98, doi:10.1137/141000671.

[5] Kenta Cho & Bart Jacobs (2019): *Disintegration and Bayesian inversion via string diagrams*. Mathematical Structures in Computer Science 29(7), pp. 938–971, doi:10.1017/s0960129518000488. arXiv:arXiv:1709.00322.

[6] Simon Danisch & Julius Krumbiegel (2021): *Makie.jl: Flexible high-performance data visualization for Julia*. Journal of Open Source Software 6(65), p. 3349, doi:10.21105/joss.03349.

[7] Tobias Fritz (2020): *A synthetic approach to Markov kernels, conditional independence and theorems on sufficient statistics*. Advances in Mathematics 370, p. 107239, doi:10.1016/j.aim.2020.107239. arXiv:arXiv:1908.07021.

[8] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari & Donald B. Rubin (2013): *Bayesian Data Analysis*. Taylor & Francis Ltd. Available at `http://www.stat.columbia.edu/~gelman/book/`.

[9] Richard Mcelreath (2020): *Statistical Rethinking*. Taylor & Francis.

[10] Joe Moeller & Christina Vasilakopoulou (2018): *Monoidal Grothendieck construction*. Theory and Applications of Categories, *Vol. 35, 2020, No. 31, pp 1159-1207.* arXiv:arXiv:1809.00727.

[11] David Jaz Myers (2023): *Categorical Systems Theory*. Available at `http://davidjaz.com/Papers/DynamicalBook.pdf`. Work in progress.

[12] Julian D. Schwab, Silke D. Kühlwein, Nensi Ikonomi, Michael Kühl & Hans A. Kestler (2020): *Concepts in Boolean network modeling: What do they all mean?* Computational and Structural Biotechnology Journal 18, pp. 571–582, doi:10.1016/j.csbj.2020.03.001.

[13] David I. Spivak (2019): *Generalized Lens Categories via functors* $\mathscr{C}^{\mathrm{op}} \to \mathsf{Cat}$. arXiv:1908.02202.

[14] Stan Development Team (2023): *Stan Modeling Language Users Guide and Reference Manual*. Available at `https://mc-stan.org`. Version 2.31.

[15] Denis Thieffry, Araceli M. Huerta, Ernesto Pérez-Rueda & Julio Collado-Vides (1998): *From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in Escherichia coli*. BioEssays 20(5), pp. 433–440, doi:10.1002/(sici)1521-1878(199805)20:5<433::aid-bies10>3.0.co;2-2.

[16] Dominik M. Wittmann, Jan Krumsiek, Julio Saez-Rodriguez, Douglas A. Lauffenburger, Steffen Klamt & Fabian J. Theis (2009): *Transforming Boolean models to continuous models: methodology and application to T-cell receptor signaling*. BMC Systems Biology 3(1), doi:10.1186/1752-0509-3-98.