Learning Biomolecular Models using Signal Temporal Logic

Hanna Krasowski KRASOWSKI@BERKELEY.EDU
University of California, Berkeley

Eric Palanques-Tost ERICPT@BU.EDU
Boston University, Boston

Calin Belta CALIN@UMD.EDU
University of Maryland, College Park

Murat Arcak ARCAK@BERKELEY.EDU
University of California, Berkeley

Editors: N. Ozay, L. Balzano, D. Panagou, A. Abate

Abstract

Modeling dynamical biological systems is key for understanding, predicting, and controlling complex biological behaviors. Traditional methods for identifying governing equations, such as ordinary differential equations (ODEs), typically require extensive quantitative data, which is often scarce in biological systems due to experimental limitations. To address this challenge, we introduce an approach that determines biomolecular models from qualitative system behaviors expressed as Signal Temporal Logic (STL) statements, which are naturally suited to translate expert knowledge into computationally tractable specifications. Our method represents the biological network as a graph, where edges represent interactions between species, and uses a genetic algorithm to identify the graph. To infer the parameters of the ODEs modeling the interactions, we propose a gradient-based algorithm. On a numerical example, we evaluate two loss functions using STL robustness and analyze different initialization techniques to improve the convergence of the approach.

Keywords: biological models, system inference, system synthesis, signal temporal logic

1. Introduction

Biological models are central to understanding and manipulating processes such as immune system response, wound healing, and cancer. For example, a model of an organism with a chronic disease can provide detailed insights into its malfunctioning (Moise and Friedman, 2019), or a model can be used as instruction to build a synthetic biological system that could be used to regulate an immune response (Nielsen et al., 2016). The two tasks, i.e., understanding and manipulating, are usually addressed by inference and synthesis, respectively. Biological network inference aims to uncover the underlying dynamics of an existing system by constructing models that align with usually quantitative data to replicate observed behaviors (Delgado and Gómez-Vela, 2019; St. John et al., 2019). Synthesis aims to build systems that comply with a desired behavior (Nielsen et al., 2016). Despite differing goals, both approaches share a common challenge: modeling dynamical systems that satisfy constraints derived from experimental data, qualitative observations, or desired outcomes.

Both inference and synthesis are challenging for the typically data-scarce problems in biology due to expensive real-world measurements. Constraining the model candidates by leveraging domain knowledge is therefore often a necessity, e.g., assuming the model is described by a set of

differential equations (Brunton et al., 2016) or adding prior knowledge about specific genes (Penfold et al., 2012). Furthermore, the majority of available data of biological systems is qualitative and implicitly embedded in natural language; for example: "If the concentration of IL-6 and TGFβ exceeds a threshold for a certain period, the concentration of Th17 cells should increase, while the induction of Treg cells should decrease after a delay" (Martinez-Sanchez et al., 2018). While such descriptions are rich in insight, they do not readily translate to computational procedures.

In this paper, we introduce a genetic algorithm that is combined with a gradient-based optimization to learn biomolecular models from Signal Temporal Logic (STL) specifications. We propose to formalize qualitative information with STL so that the information becomes computationally tractable. We represent the biomolecular system as a graph, where nodes are biological entities (i.e., cells, cytokines, or genes) and edges represent interactions (i.e., inhibition and activation) between them. The graph is transformed to an ordinary differential equation (ODE) model leveraging biological domain knowledge. Our genetic algorithm searches for the optimal graph structure while evaluating each population with respect to edge sparsity and satisfaction of STL specifications. We optimize the unknown ODE parameters for each candidate of a population by employing a differentiable measure of STL in the loss of a gradient-based algorithm.

Our contributions are: (1) We introduce an approach for learning interpretable and biologically-grounded ODE models from STL statements, which is applicable when only qualitative data is available. (2) The combination of a genetic algorithm with a gradient-based optimization allows for simultaneously identifying the network structure and unknown parameters for biomolecular models of intracellular or intercellular processes. (3) The approach is suitable for both inference and synthesis, as the STL specifications can describe existing or intended system behavior. (4) We propose and evaluate an adaptive loss function and a parameter initialization strategy to robustify convergence.

The remainder of this paper is structured as follows. In Section 2, we contextualize our contributions with respect to existing literature. In Section 3 and Section 4, we define relevant concepts and formulate the main problem. We describe our algorithm to learn biomolecular models in Section 5, and show its effectiveness on a numerical example of a gene network in Section 6. Discussions and conclusions are in Section 7 and Section 8.

2. Related Work

ODE inference for biological systems. A common problem in systems biology is the inference of the parameters of the ODEs modeling a system. Early methods, such as Varah (1982), estimate the derivative of the observable times-series data and use optimization techniques such as least squares to fit ODE parameters. However, these methods were not accurate for complex ODEs. Recently, more nuanced methods have been proposed, such as Bayesian inference for parameter fitting (Huang et al., 2020) or a combination of Particle Swarm Optimization (PSO) with reinforcement learning to estimate the parameters ODEs (Sun et al., 2024). Since the structure of the ODEs is not necessarily known, several methods were developed to infer the parameters and the structure of the governing ODEs simultaneously. A key contribution is the SINDy approach by Brunton et al. (2016), which creates a large matrix with candidate terms for the ODEs and then applies sparse regression to select the most relevant ones. An alternative approach is using genetic algorithms, as by Rossi et al. (2024), starting from an initial population of equations and selecting based on a fitness function that is tailored to the desired features. While these approaches identify interpretable ODEs, pre-

selecting a trackable amount of terms of the ODEs and achieving sparsity of biological systems (Busiello et al., 2017) from a fully connected initialization can be challenging.

To avoid selecting ODE terms, black-box neural networks can be used to represent differential equations (Chen et al., 2018; Raissi, 2018; Raissi et al., 2019). By inducing the underlying structure of differential into the learning, fewer data samples and more robust predictions can be obtained than using neural networks without enforcing any structure. More recent research aims to increase the interpretability of the final system model by combining neural differential equations with known components. For example, the work by Ahmadi Daryakenari et al. (2024) proposes a two-step process: training neural networks for unknown components of the system model and performing symbolic regression to identify explicit expressions for the trained neural networks.

Our approach jointly learns both the structure and parameters of interpretable and sparse ODEs based on modeled interactions of biological species. In contrast to SINDy (Brunton et al., 2016), which relies on a pre-defined set of basis functions, we derive the ODEs from a graph that represents the interactions between species and is not a linear combination of the underlying interaction ODEs. Since the interactions between species are commonly sparse in nature, we use the fitness function of the structure-searching genetic algorithm to identify sparse networks.

Biological network synthesis. Synthetic biology aims to design genetic circuits that satisfy a particular behavior and that can be engineered in a cell. One of the most common tools, Cello, by Nielsen et al. (2016), is a comprehensive framework for designing genetic circuits, translating specified behaviors into DNA sequences for implementation in organisms like Escherichia coli. However, in Cello the desired behavior is specified with aggregation of logic functions and does not directly handle temporal dependencies that can be expressed with temporal logic. Several approaches for circuit synthesis have been proposed, such as those by Goldfeder and Kugler (2019); Bernot and Comet (2010), which extend biological circuit synthesis for specifications in temporal logic. Specifically, these methods aid network design by ensuring compliance with expected temporal logic specifications. Bartocci et al. (2013) propose to use STL to identify parameters of a pre-defined circuit and develop a closed-form parameter identification algorithm for specific circuit modules. In contrast, our approach uses STL robustness for parameter optimization and network design simultaneously.

3. Preliminaries

3.1. Notation

We use $\mathbf{x} : [0, T] \mapsto \mathbb{R}^N$ to denote the vector of species concentrations over the time interval $[0, T]$; $x \in \mathbb{R}^N$ is the concentration at a particular time and its ith entry x_i is the concentration of species i. We represent the interactions in a system with a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathcal{T})$, where $\mathcal{N}$ is the set of nodes, i.e., species, $\mathcal{E}$ is the set of edges, i.e., interactions among species, and $\mathcal{T}$ is the set of edge types. An edge is specified with a tuple $e = (u, v, j) \in \mathcal{E}$ meaning that it goes from node u to v and has type j. The numbers of nodes and edges are defined by $|\mathcal{N}|$ and $|\mathcal{E}|$, respectively. We denote the set $\mathcal{F} = \mathcal{N} \times \mathcal{N} \times \mathcal{T}$ to represent all edges and edge types that could exist in a graph. We also define a mapping $\Psi : \mathcal{G} \mapsto \mathcal{D}$ that converts the information in $\mathcal{G}$ into a dynamical system represented by ODEs with parameters $\theta \in \mathcal{P}$ and external inputs a. The mapping associates each edge e with specific terms such as the ones presented in Section 3.3, depending on the type of edge.

For instance, an edge e of type $j = 1$ could be modeled with Eq. (2) between two species with concentrations z and y.

3.2. Signal Temporal Logic

We use temporal specifications defined by STL formulas; see Maler and Nickovic (2004) for the formal syntax and semantics. STL formulas have three components: (1) predicates $\mu := g(x) > 0$, with $g : \mathbb{R}^N \to \mathbb{R}$; (2) Boolean operators, such as $\neg$ (*negation*), $\wedge$ (*conjunction*), and $\implies$ (*implication*); and (3) temporal operators, such as $\Diamond$ (*eventually*) and $\Box$ (*always*). Given as STL formula φ, $\Diamond_{[t_1,t_2]}\psi$ is true if ψ is satisfied for at least one time point $t \in [t_1, t_2]$, while $\Box_{[t_1,t_2]}\psi$ is true if ψ is satisfied for all time points $t \in [t_1, t_2]$.

The semantics of STL is defined over functions of time, or signals, such as trajectories produced by a system of ODEs. STL has both qualitative (Boolean) and quantitative semantics. In the Boolean semantics, a trajectory $\mathbf{x}$ either satisfies a formula φ (written as $\mathbf{x} \models \varphi$) or violates it. The quantitative semantics (Donzé and Maler, 2010) is defined using a function $\rho(\mathbf{x}, \varphi)$, called robustness, which measures how strongly formula φ is satisfied by $\mathbf{x}$. In particular, $\rho(\mathbf{x}, \varphi) \geq 0$ if and only if $\mathbf{x} \models \varphi$. The time horizon of an STL formula φ, denoted as $hrz(\varphi)$, is the minimum amount of time required to decide the satisfaction of φ (Sadraddini and Belta, 2015). For simplicity, we assume that $T = hrz(\varphi)$. For example, the quantitative semantics for the conjunction of formulas φ_1 and φ_2 is defined as (Leung et al., 2023):

$$\rho(\mathbf{x}, \varphi_1 \wedge \varphi_2) = \min(\rho(\mathbf{x}, \varphi_1), \rho(\mathbf{x}, \varphi_2)). \tag{1}$$

3.3. Mathematical Modeling of Biological Systems

Typical models in systems biology employ ODEs with equations that describe the dynamics of molecular and cellular interactions; Murray (2001); Keener and Sneyd (2004); Edelstein-Keshet (2005); Alon (2006). Among them, Hill equations are widely used to describe the non-linear binding and regulation of biological molecules. As an example, Eq. (2) below indicates positive regulation of a molecule, whose concentration is denoted by y, by another molecule with concentration z, while Eq. (3) describes negative regulation. In both cases, ν represents the maximum activation/inhibition rate, K represents the affinity between the two molecules, and the *Hill coefficient* n determines the steepness of the curve, with larger parameters tending towards a step-like behavior. The term $-\gamma \cdot y$ in each equation models degradation at a rate proportional to the concentration:

$$\dot{y} = -\gamma \cdot y \; + \; \nu \cdot \frac{K \cdot z^n}{1 + K \cdot z^n} \tag{2}$$

$$\dot{y} = -\gamma \cdot y \; + \; \nu \cdot \frac{1}{1 + K \cdot z^n}. \tag{3}$$

Note that a differential equation model can also include a combination of positive and negative regulation terms. There are other modeling paradigms in systems biology, including those that account for stochasticity, such as Chemical Master Equation models and their diffusion approximations leading to the Chemical Langevin Equation; see, e.g., van Kampen (2007). As a starting point, we focus on ODE models with standard activation and inhibition terms listed above due to their computational tractability.

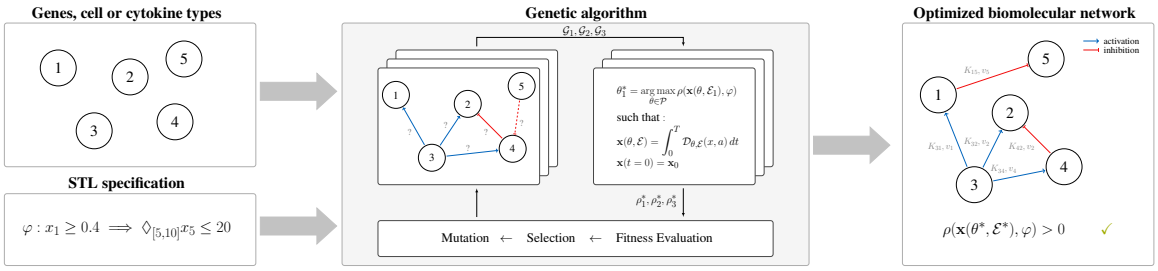

Figure 1: Model learning approach: Given a set of species and a formalized specification, a genetic algorithm searches for the optimal topology while the parameters for each candidate graph are optimized based on the robustness of the formalized specification. The result is a specification-compliant biomolecular model with optimized parameters that is modeled by ODEs.

4. Problem Statement and Approach

Given species of a system that are represented as nodes $\mathcal{N}$ of a graph $\mathcal{G}$, an observable or intended behavior for that system in the form of an STL formula φ, and a mapping Ψ that converts graphs $\mathcal{G}$ to dynamical systems $\mathcal{D}$ with parameters θ, we want to find the edges of the graph $\mathcal{E}^* \in \mathcal{F}$ and the parameters of its respective dynamical system $\theta^* \in \mathcal{P}$ such that the robustness of the traces $\mathbf{x}$ generated by $\mathcal{D}$ is maximized. Mathematically:

$$\theta^*, \mathcal{E}^* = \underset{\theta \in \mathcal{P},\, \mathcal{E} \in \mathcal{F}}{\arg\max}\, \rho(\mathbf{x}(\theta, \mathcal{E}), \varphi). \tag{4}$$

Our approach to the above problem is illustrated in Figure 1. A major challenge is that the parameters θ can only be optimized and evaluated for a given edge set $\mathcal{E}$. Thus, we develop a genetic algorithm where the fitness function is calculated based on the topology of the graph and the robustness of the optimized system. Additionally, we propose two extensions: a pre-optimized parameter initialization to decrease the sensitivity to the initialization of the parameters θ, and an adaptive loss for conjunction STL specifications that switches to an additive robustness when the STL specification is satisfied.

5. Biomolecular Model Learning

In our approach, we consider two possible edge types: inhibition and activation. Our mapping Ψ relates inhibition and activation edges to Eq. (3) and (2), respectively. The mapping also incorporates the degradation term of each species, which, intuitively, in the graph would be a self-loop around each node. In particular, we define the mapping between $\mathcal{G}$ and the dynamical system $\mathcal{D}$ with the ODEs $\mathcal{D}_{\theta,\mathcal{E}}(x, a)$:

$$\dot{x}_u = -\gamma_u x_u + \nu_u \frac{\mathbb{1}_d + \mathbb{1}_r \sum_{\{e \mid \texttt{sink}(u,e) \wedge \texttt{type}(a,e)\}} K_e\, x_v^n}{1 + \sum_{\{e \mid \texttt{sink}(u,e)\}} K_e\, x_v^n} + a_u, \tag{5}$$

Where $a \in \mathbb{R}^N$ is the external input and a_u is the external input of node u, $\mathbb{1} \in \{0, 1\}$ denotes a binary variable, $\mathbb{1}_d = 1$ if there are solely inhibition edges incoming to u, and $\mathbb{1}_r = 1$ if there is at least one activation edge incoming to edge u. The function $\texttt{sink}(u, e)$ evaluates to true if u is a sink node for the edge e, and the function $\texttt{type}(j, e)$ that evaluates to true if the edge e is of type j. Based on this formulation, our approach aims to find the set of edges $\mathcal{E} \in \mathcal{F}$ and parameters $\theta \in \mathcal{P}$, i.e., decay rates γ, maximum activation/inhibition rate ν, and affinity factor K, that maximize the robustness ρ of the STL specification φ.

5.1. Network Learning

We learn the biological network structure with a genetic algorithm. We start from an initial population pop_0 where we randomly create edges with random edge types. Since sparse connections are common in biological problems, we bias the initial population towards sparse connections by sampling between $|\mathcal{N}| - 1$ and $\frac{|\mathcal{N}|^2}{4}$ edges. In case not all species are connected after sampling, we add the minimum number of edges necessary to connect subgraphs to one graph and randomly sample an edge type. For each candidate graph, we optimize the parameters θ with respect to the STL specifications as detailed in Section 5.2. The selection function identifies the top k performers based on the fitness function which considers the robustness and edge sparsity. Specifically, the fitness function consist of a robustness measure $\mathcal{L}_S(\rho^*, \rho_{best})$ and an edge sparsity measure $\mathcal{L}_E(|\mathcal{E}|)$:

$$\mathcal{L}_{GA}(\rho^*, |\mathcal{E}|, \rho_{best}) = \underbrace{\frac{\rho^*}{\rho_{best}}}_{=\mathcal{L}_S} + \underbrace{1 - \frac{|\mathcal{E}| - |\mathcal{N}| + 1}{|\mathcal{N}|^2 - |\mathcal{N}|}}_{=\mathcal{L}_E}, \tag{6}$$

where ρ^* is the optimized robustness of the candidate and ρ_{best} is the best-observed robustness up to the current generation. The edge sparsity measure is 1 if the minimum amount of edges to connect all nodes is used and converges to 0 with an increasing amount of edges. Thus, the fitness evaluation is biasing towards sparse graphs. For the next population, we mutate the top population with three mutation types with probability α each: removing one edge, adding one edge, and resampling the edge types. Note that we post-process the edge removal mutation to ensure the graph is still fully connected.

5.2. Parameter Optimization

The quantitative semantics of the STL specification enables the use of the robustness of a trace as the loss function in our optimization, summarized in Algorithm 1: First, a trace for the system with the current parameters is computed. The loss $\mathcal{L}_\rho$ is the negative robustness of the trace. Then, the parameters are updated with a gradient descent method, such as Adam. Note that we approximate the continuous-time differential equation with a discrete-time Euler step update, as this allows us to efficiently backpropagate the gradient. The optimization is terminated when the robustness change between two optimization steps is smaller than ϵ or the maximum iterations are reached. We propose two extensions to our algorithm to improve efficiency and effectiveness: a robust parameter initialization strategy and an adaptive loss function.

Parameter initialization. Initialization of the ODE parameters plays a critical role in the convergence of the algorithm, especially when more complex STL specifications are provided. We suggest performing a parameter search to initialize the parameters with the lowest loss function possible.

Algorithm 1 Gradient-based parameter optimization with STL specifications

Initialize: ODEs $\mathcal{D}_{\theta,\mathcal{E}}(x,a)$, parameters $\theta = K_i \forall i \in \{1,...,|\mathcal{E}|\}, \nu_j, \gamma_j, j \in \{1,...,|\mathcal{N}|\}$, STL specification φ, initial state $\mathbf{x}_0$, parameter set $\mathcal{P}$, maximum iterations $n_{\max}$, hyperparameters λ, ϵ
Initialize iteration $n \leftarrow 0$
while $\neg$converged $\wedge$ $n \leq n_{\max}$ **do**
  Compute trace $\mathbf{x}$ with $\mathcal{D}_{\theta,\mathcal{E}}(x,a)$ and $\mathbf{x}_0$
  Compute loss $\mathcal{L}_\rho = -\rho^n(\mathbf{x}, \varphi)$
  Update parameters $\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}_\rho$
  Project parameters on parameter set $\theta \leftarrow \texttt{clip}(\theta, \mathcal{P})$
  converged $\leftarrow$ **true** if $|\rho^n - \rho^{n-1}| \leq \epsilon \wedge n \neq 0$
  $n \leftarrow n + 1$
end while

That is, given a bounded set of possible parameters $\mathcal{P}$, we initialize the parameters that lead to the minimum loss for a specific candidate graph $\mathcal{G}$:

$$\theta = \arg\min_{\theta_i \in \Theta_0} \underbrace{-\rho(\mathbf{x}(\theta_i), \varphi)}_{=\mathcal{L}_\rho} \tag{7}$$

The set $\Theta_0 \subset \mathcal{P}$ can be created using a grid or from random samples. Here, we use Sobol sequences to sample from the parameter space $\mathcal{P}$.

Adaptive loss function. Often, the STL specification will be based on multiple qualitative statements that have to hold simultaneously. This translates to a conjunction of multiple subspecifications reflecting each of the statements. The robustness of a conjunction is the minimal robustness of the subspecifications (see Eq. (1)). Thus, the parameters are optimized with respect to the least robust specification. While this objective is meaningful as long as the specification is not fulfilled with the current parameters, it is not clear if this is the best choice once all subspecifications are fulfilled, especially since it is difficult in practice to scale all predicates to one range. Additionally, the least robust subspecification might not be optimizable beyond a specific threshold, while for the other subspecifications improvement might be possible without decreasing the robustness of the least robust subspecification. To address these concerns, we propose to use a loss function that changes the objective once the specification is fulfilled, i.e., an adaptive loss function:

$$\mathcal{L}_A = \begin{cases} \mathcal{L}_\rho & \rho \leq 0 \\ -\sum_{j=1}^{J} \rho_j - \beta \sum_{j=1}^{J} \log(\rho_j), & \rho > 0 \end{cases} \tag{8}$$

where $\mathcal{L}_\rho$ is the original loss as formulated in Algorithm 1, β is a weighting factor, ρ_j denotes the robustness of subspecification φ_j, and the STL specification consists of J subspecifications. The first sum aims to maximize the robustness gain while the second term penalizes stepping toward the satisfaction boundary, i.e., $\rho = 0$.

6. Numerical Example

Our biomolecular model learning can be applied to multiple biological networks such as cell-cell and cell-cytokine networks, or gene regulatory networks. For our evaluation, we investigate signal

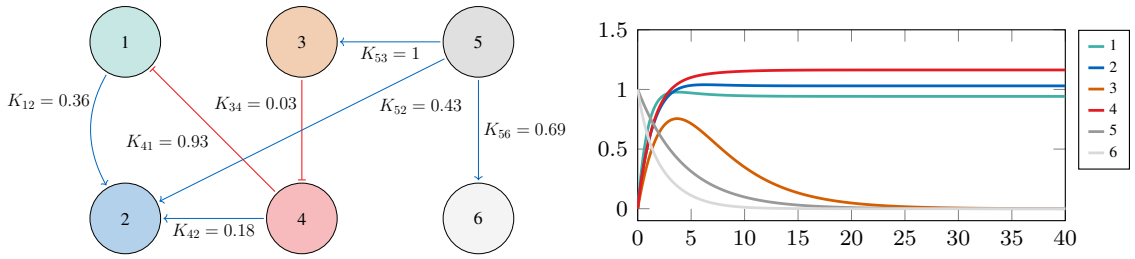

Figure 2: Gene network with highest robustness. Left: Graph including optimized affinity factors while the decay rates are $\gamma = [0.77, 0.81, 0.20, 0.63, 0.22, 0.93]$ and the activation/inhibition rates are $\nu = [0.51, 0.93, 1, 0.73, 0.22, 0.91]$. Right: Trajectory for gene network showing satisfaction of the specification ϕ.

propagation which is an important phenomenon in gene networks (Matsuda et al., 2012). An informal specification of a propagation system could be: "Given protein X, produce protein Y within a short time interval". For our numerical evaluation, we investigate such a specification for a system with six species, i.e., $|\mathcal{N}| = 6$, where if species 1 or 2 is above a threshold, two other species 3 and 4 increase in concentration, while never exceeding a maximum concentration. In particular, we consider the STL specification:

$$\begin{aligned}\phi : &\big(x_1 \geq 0.2 \wedge x_2 \geq 0.3 \implies (\Diamond_{[0,10]}\, x_3 \geq 0.5) \wedge (\Diamond_{[0,10]} \Box\, x_4 \geq 0.9)\big) \wedge \\ &\big(x_4 \geq 0.6 \implies \Diamond_{[0,20]} \Box\, x_3 \leq 0.3\big) \\ &\wedge (\Box x_1 \leq 1.5) \wedge (\Box x_2 \leq 1.5) \wedge (\Box x_3 \leq 1.5) \wedge (\Box x_4 \leq 1.5)\,. \end{aligned} \tag{9}$$

The initial state is specified as $\mathbf{x}_0 = [0, 0, 0, 0, 1, 1]$, the propagation signal is $a = [0.5, 0.5, 0, 0, 0, 0]$. We restrict all, i.e., p, parameters in the interval $\mathcal{P} = [0.001, 1]^p$ and set the Hill factor to $n = 2$. The parameters for the evolutionary algorithm are top population $k = 5$, mutation rate $\alpha = 0.7$, the initial population is $pop_0 = 15$, and we search for 10 generations. For the gradient-based optimization, we use Adam for gradient descent and set the learning rate to $\lambda = 0.04$, the convergence threshold $\epsilon = 0.00001$, the maximum iterations $n_{\max} = 80$, the weight in the adaptive loss to $\beta = 1$. For the parameter initialization, we sample 20 times using Sobol sequences from the parameters space $\mathcal{P}$ to identify the best initial parameters.

6.1. Results

We observe that in 9 of the 10 generations, our genetic algorithm finds at least one system with robustness above 0.15. Figure 2 shows the gene network with the highest robustness, 0.251, obtained by the genetic algorithm. The species 5, which starts from a high concentration, activates species 3, which inhibits species 4. This relates to the second subspecification. To fulfill the first subspecification, species 5 regulates species 3 and 4. Thus, the implication does not necessarily lead to a path between the species for the given initial state and propagation signal.

Since the parameter optimization of Algorithm 1 is repeatedly used for determining the fitness of a candidate graph, it needs to converge robustly. Thus, we investigate the effectiveness of the parameter initialization and the differences between loss functions in Table 1. We observe that our

	$\mathcal{L}_\rho$		$\mathcal{L}_\mathcal{S}$	
	uniform	Sobol	uniform	Sobol
$\rho(\mathbf{x}, \phi)$	0.167 ± 0.087	0.194 ± 0.093	0.116 ± 0.067	0.162 ± 0.067
$\sum_i \rho_i(\mathbf{x}, \phi_i)$	0.558 ± 0.217	0.643 ± 0.238	0.628 ± 0.137	0.719 ± 0.179
steps to convergence	76.3 ± 5.5	65.4 ± 19.2	73.4 ± 14.5	73.8 ± 12.2
steps to satisfaction	16.6 ± 15.4	8.6 ± 12.6	17.2 ± 21.6	9.1 ± 15.6

Table 1: Comparison of parameter optimization with different initialization strategies (*uniform* sampling from $\mathcal{P}$ and pre-optimization with Sobol sequences), and loss functions for the three best candidate systems. For each system, the optimization is run ten times.

Sobol initialization leads to robustly learning parameters that achieve high satisfaction with the STL specification and reach STL-satisfying parameters set with fewer gradient steps. While the adaptive loss does lead to a lower mean robustness, it leads to higher additive robustness of subspecifications. The steps necessary for reaching the convergence criteria are similar when considering the standard deviation intervals. Additionally, we observe that with the loss $\mathcal{L}_\rho$, it happens that a gradient step leads to leaving the satisfaction parameter space again, especially for larger learning rates.

7. Discussion

While the numerical example above validates the approach, further studies are needed to test broader applicability. In particular, the single initial state and noise-free input are a simplification of reality. The extension to state and input sets could be implemented by sampling a trajectory batch from input and state sets and applying gradient descent to optimize the ODE parameters on the batch. Another potential improvement is using advanced ODE solvers suited for stiff ODEs, e.g., Kverno5 (Kværnø, 2004), instead of the Euler method. The best network identified in our example does not have a path between the species on the left side of the implications to the species on the right side of the implications. If such behavior was intended, it could be forced by constraining the possible edge set $\mathcal{F}$.

We restricted the used ODEs to Hill equations for activation and inhibition, and degradation terms. Beyond these three reactions, biological systems commonly include reactions such as sequestration or logarithmic growth, which could be included as additional edge types in our graph representation. Still, this would result in an exponential increase of possible graphs, which could be challenging for the convergence of the genetic algorithm. Gradient-based methods for graph neural networks (Franceschi et al., 2019; Zheng et al., 2020) could potentially improve scalability, but they usually require many examples to learn. Additionally, larger networks and more complex specifications will significantly increase the difficulty of identifying even a single viable model candidate. This complexity could be addressed by decomposing the specifications, learning models for the components, and then learning the connections between components. Similar to the concept in (Ahmadi Daryakenari et al., 2024), the connections between components could be initially approximated by neural ODEs.

Our approach assumes that an expert translates qualitative information from experiments and publications into STL specification. While this is possible for many biological systems, a more

automated translation, e.g., based on large language models, will be necessary to investigate broad applicability. Additionally, for systems with limited qualitative information, a large set of candidate networks can comply with the given specifications due to underspecification (as in our example). When more information becomes available through experiments or expert feedback, the candidates could be refined so that eventually the real system model is inferred or the model that can be used for synthesizing a biological system in the lab is identified. Our approach is a flexible framework that can account for new information by refining the STL specification or by biasing the population generation with the new knowledge or with previously found candidates. When quantitative data becomes available, it can be directly incorporated by adding a loss term or using STL inference methods, such as those presented by Li et al. (2023) or Bombara and Belta (2021) to convert the quantitative data into refined STL statements.

8. Conclusion

We proposed a learning approach for biomolecular models from STL specifications of system behavior, which is suitable for inference and synthesis. We introduced a graph representation that is mapped to biologically-grounded ODEs so that our genetic algorithm can simultaneously optimize the topology of the graph and the parameters of the ODEs. We addressed the challenge of robust parameter initialization and choice of appropriate loss functions. On a numerical example, we observed that our approach identifies multiple system candidates that satisfy the specification. Thus, validating that our approach achieves learning of biomolecular models from STL specifications, which describe qualitative observations.

Acknowledgments

This work was funded in part by the Air Force Office of Scientific Research grant FA5590-23-1-0529, NSF GCR 2219101, and NIH R01 EB030946.

References

Nazanin Ahmadi Daryakenari, Mario De Florio, Khemraj Shukla, and George Em Karniadakis. AI-Aristotle: A physics-informed framework for systems biology gray-box identification. *PLOS Computational Biology*, 20(3):1–33, 2024.

Uri Alon. *An introduction to systems biology: Design principles of biological circuits*. Chapman and Hall/CRC, 2006.

Ezio Bartocci, Luca Bortolussi, and Laura Nenzi. A temporal logic approach to modular design of synthetic biological circuits. In *Computational Methods in Systems Biology*, pages 164–177, 2013.

Gilles Bernot and Jean-Paul Comet. On the use of temporal formal logic to model gene regulatory networks. In Francesco Masulli, Leif E. Peterson, and Roberto Tagliaferri, editors, *Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 112–138, 2010.

Giuseppe Bombara and Calin Belta. Offline and online learning of signal temporal logic formulae using decision trees. *ACM Transactions on Cyber-Physical Systems*, 5(3):1–23, 2021.

Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

Daniel M Busiello, Samir Suweis, Jorge Hidalgo, and Amos Maritan. Explorability and the origin of network sparsity in living systems. *Scientific reports*, 7(1), 2017.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Proc. of the Int. Conf. on Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

Fernando M. Delgado and Francisco Gómez-Vela. Computational methods for gene regulatory networks reconstruction and analysis: A review. *Artificial Intelligence in Medicine*, 95:133–145, 2019.

Alexandre Donzé and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 92–106, 2010.

Leah Edelstein-Keshet. *Mathematical Models in Biology*. SIAM, Philadelphia, 2005.

Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 1972–1982, 2019.

Judah Goldfeder and Hillel Kugler. Temporal logic based synthesis of experimentally constrained interaction networks. In *Molecular Logic and Computational Synthetic Biology*, pages 89–104, 2019.

Hanwen Huang, Andreas Handel, and Xiao Song. A bayesian approach to estimate parameters of ordinary differential equation. *Computational Statistics*, 35(3):1481–1499, 2020.

James Keener and James Sneyd. *Mathematical Physiology*. Springer, New York, NY, 2004.

Anne Kværnø. Singly diagonally implicit runge–kutta methods with an explicit first stage. *BIT Numerical Mathematics*, 44(3):489–502, 2004.

Karen Leung, Nikos Aréchiga, and Marco Pavone. Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods. *The International Journal of Robotics Research*, 42(6):356–370, 2023.

Danyang Li, Mingyu Cai, Cristian-Ioan Vasile, and Roberto Tron. Learning signal temporal logic through neural network for interpretable classification. In *Proc. of the American Control Conf. (ACC)*, pages 1907–1914, 2023.

Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 152–166, 2004.

Mariana E. Martinez-Sanchez, Leonor Huerta, Elena R. Alvarez-Buylla, and Carlos Villarreal Luján. Role of cytokine combinations on CD4+ T Cell differentiation, partial polarization, and plasticity: Continuous network modeling approach. *Frontiers in Physiology*, 9, 2018.

Mitsuhiro Matsuda, Makito Koga, Eisuke Nishida, and Miki Ebisuya. Synthetic signal propagation through direct cell-cell interaction. *Science Signaling*, 5(220):ra31–ra31, 2012.

Nicolae Moise and Avner Friedman. Rheumatoid arthritis - a mathematical model. *Journal of Theoretical Biology*, 461:17–33, 2019.

James D. Murray. *Mathematical Biology, I: An Introduction*. Springer, New York, third edition, 2001.

Alec A. K. Nielsen, Bryan S. Der, Jonghyeon Shin, Prashant Vaidyanathan, Vanya Paralanov, Elizabeth A. Strychalski, David Ross, Douglas Densmore, and Christopher A. Voigt. Genetic circuit design automation. *Science*, 352(6281):aac7341, 2016.

Christopher A. Penfold, Vicky Buchanan-Wollaston, Katherine J. Denby, and David L. Wild. Nonparametric Bayesian inference for perturbed and orthologous gene regulatory networks. *Bioinformatics*, 28(12):i233–i241, 2012.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19(25):1–24, 2018.

Nicoló Rossi, Ankit Gupta, and Mustafa Khammash. Synthevo: A gradient-guided evolutionary approach for synthetic circuit design. In *Proc. of the IEEE Conference on Decision and Control (CDC)*, pages 6298–6304, 2024.

Sadra Sadraddini and Calin Belta. Robust temporal logic model predictive control. In *In Proc. of the Annual Allerton Conference on Communication, Control, and Computing*, pages 772–779, 2015.

Peter C. St. John, Jonathan Strutz, Linda J. Broadbelt, Keith E. J. Tyo, and Yannick J. Bomble. Bayesian inference of metabolic kinetics from genome-scale multiomics data. *PLOS Computational Biology*, 15(11):1–23, 2019.

Wenkui Sun, Xiaoya Fan, Lijuan Jia, Tinyi Chu, Shing-Tung Yau, Rongling Wu, and Zhong Wang. Estimating unknown parameters in differential equations with a reinforcement learning based PSO method, 2024.

Nicolaas G. van Kampen. *Stochastic Processes in Physics and Chemistry*. North-Holland Personal Library. North-Holland, Amsterdam, 3rd edition, 2007. ISBN 978-0444529657. Revised and enlarged edition.

J. M. Varah. A spline least squares method for numerical parameter estimation in differential equations. *SIAM Journal on Scientific and Statistical Computing*, 3(1):28–46, 1982.

Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 11458–11468, 2020.