HAL
open science

Dynamic Influence Networks for Rule-Based Models

Angus G. Forbes, Andrew Burks, Kristine Lee, Xing Li, Pierre Boutillier, Jean Krivine, Walter Fontana

To cite this version:

Angus G. Forbes, Andrew Burks, Kristine Lee, Xing Li, Pierre Boutillier, et al.. Dynamic Influence Networks for Rule-Based Models. IEEE Transactions on Visualization and Computer Graphics, 2018, 24 (1), pp.184-194. 10.1109/TVCG.2017.2745280 . hal-02399102

HAL Id: hal-02399102

https://hal.science/hal-02399102v1

Submitted on 18 Dec 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:1711.00967v1 [cs.GR] 2 Nov 2017

Dynamic Influence Networks for Rule-based Models

Angus G. Forbes, Andrew Burks, Kristine Lee, Xing Li, Pierre Boutillier, Jean Krivine, and Walter Fontana

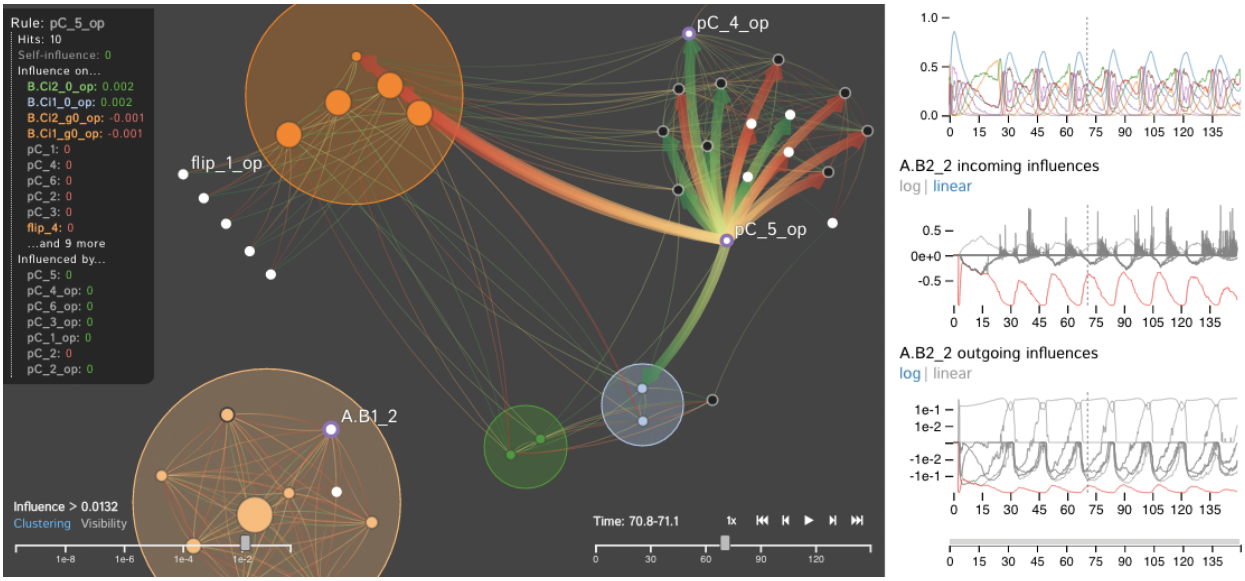

Fig. 1. A screenshot of the *DIN-Viz* application for analyzing the dynamics of a rule-based model of a protein-protein interaction network (i.e., a Dynamic Influence Network). Our approach emphasizes the influence rules have on each other and enables users to analyze the dynamics of these influences as they change over time. The left panel shows a network of interconnected rules at a specific time step. The right panel provides a global overview of the system as well as detailed information about selected rules.

Abstract—We introduce the Dynamic Influence Network (DIN), a novel visual analytics technique for representing and analyzing rule-based models of protein-protein interaction networks. Rule-based modeling has proved instrumental in developing biological models that are concise, comprehensible, easily extensible, and that mitigate the combinatorial complexity of multi-state and multi-component biological molecules. Our technique visualizes the dynamics of these rules as they evolve over time. Using the data produced by *KaSim*, an open source stochastic simulator of rule-based models written in the *Kappa* language, DINs provide a node-link diagram that represents the influence that each rule has on the other rules. That is, rather than representing individual biological components or types, we instead represent the rules about them (as nodes) and the current influence of these rules (as links). Using our interactive *DIN-Viz* software tool, researchers are able to query this dynamic network to find meaningful patterns about biological processes, and to identify salient aspects of complex rule-based models. To evaluate the effectiveness of our approach, we investigate a simulation of a circadian clock model that illustrates the oscillatory behavior of the KaiC protein phosphorylation cycle.

Index Terms—Dynamic networks, biological data visualization, rule-based modeling, protein-protein interaction networks.

1 Introduction

In an oft-cited overview of the goals of computational systems biology, Hiroaki Kitano explains the role of simulation in visualizing and understanding complex systems. Simulation, he writes, can "predict the dynamics of systems so that the validity of the underlying assumptions can be tested" [34]. Furthermore, he articulates the importance of dynamic diagrams that help to identify patterns in biological systems, to explain why they emerge, and to enable researchers to understand how they can be controlled [35]. In this paper, we introduce the Dynamic Influence Network, a novel representation that enables the effective visualization of the dynamics of a system of rules to describe complex biological processes.

A central challenge across complex systems is to understand how a multitude of heterogenous and interacting agents can give rise to coherent system behavior. The nature of the challenge is as much empirical as it is theoretical. It is empirical, because the relevant agents of a system, their interactions, and the global behavior they induce need to be discovered and characterized. It is theoretical, because the connections between agent-centric interactions and system-level behavior need to be grasped at a level beyond mere description, so as to separate contingency from necessity.

The technical challenge in modeling, even at a high level of abstraction, a complex system like a cell is that the molecular components and their manifold interactions often result in an overwhelming combinatorial complexity, frustrating an initial insight needed for guiding the construction of a model. The challenge is particularly acute for dynamical models of a mechanistic kind, which are preferred over statistical black boxes when a causal account of system behavior is sought in terms of interactions between molecular agents. The molecular level is privileged because of the fundamental role it plays in our understanding

- *Angus G. Forbes is with University of California, Santa Cruz. E-mail: angus@ucsc.edu. Website:* `https://creativecoding.soe.ucsc.edu/`
- *Andrew Burks, Kristine Lee, and Xing Li are with University of Illinois at Chicago. E-mail: {aburks3, khlee2, xli227}@uic.edu.*
- *Jean Krivine is with Université Paris Diderot. E-mail: jean.krivine@irif.fr.*
- *Pierre Boutillier and Walter Fontana are with Harvard Medical School. E-mail: {pierre_boutillier, walter}@hms.harvard.edu.*

of cellular processes during development, disease, and evolution.

A specific case in point are molecular signaling processes in which criss-crossing cascades of interactions between proteins are triggered in response to exogenous or endogenous molecular events. These cascades integrate a multitude of signals through which a cell homes in on an appropriate adaptive response, such as moving, dividing, differentiating, repairing, maintaining homeostasis, and possibly attacking itself or other cells.

Proteins are large molecules that mediate these signaling cascades by virtue of chemical tags attached to specific locations (sites) at their surface. These tags influence a protein's behavior. For example, a phosphorylation can change the conformation of a protein, thereby modulating its chemical activity or altering which other proteins it can transiently recognize and associate with. The set of tags, i.e. the state of a protein, is in turn modified through interaction with other proteins whose behavior is controlled by their own tagging state.

1.1 The *Kappa* Language for Rule-based Modeling

Information about the state conditions required for an interaction between specific proteins can be extracted or inferred from a rapidly growing experimental literature and curated databases. The *Kappa* language [19] is designed to formally express such interactions as rules that can be executed by machine. The approach follows the representational scheme of chemistry where graphs are used to express molecules as connected atoms and graph-rewrite rules capture local bond rearrangements that result in chemical reactions. There is a crucial distinction between a *mechanism*, in which parts of entities are responsible for a specific transformation, and a *reaction*, in which molecular species (the full entities) affected by the transformation are produced or consumed. A *Kappa* rule is a schema that does not require a full specification of the interacting entities, referring only to the parts that are necessary for the transformation. At the heart of rule-based modeling is the *agent abstraction*, which conceptualizes a protein as an agent defined by a name and an interface of sites that represent distinct interaction capabilities, e.g. binding and modification, without explicitly representing the underlying structural or chemical features that enable these capabilities [9, 10, 19].

KaSim is a stochastic simulator of rule-based models written in the *Kappa* language [8]. *Kappa* expresses only local rules of action, which are both "descriptions of mechanistic knowledge and executable instructions" [25]. Since the definitions are completely local and not based on any kind of enumeration, the simulation algorithm that executes these rules has a per event time cost which is independent of the size of the set of generable species, or element types, and independent of the total size of the elements or agents in the system. Even on a standard desktop computer, *KaSim* can easily generate simulations of a system with millions of agents and that is defined by hundreds of rules [18]. The output of a *KaSim* simulation, which is ingested into our *DIN-Viz* software, contains information about: a) a series of *observables*, the population of elements or groups of elements within the system; as well as b) the number of times individual rules fire over time, or alternately, the probability that they fire in different configurations. We explicitly define this *influence* metric for rule interactions in Sect. 3.1.

1.2 Visual Analysis for Understanding Rule-based Models

The rule-based approach to models proceeds without a preconceived notion of how the global behavior of interest arises in a system. In traditional models, e.g., reaction-based models, one often has a mechanism in mind and aims at proving that it indeed behaves as claimed. If such initial insight is not available, one may proceed by carefully expressing known facts about interactions between parts of entities. The result is a collection of local rules. The problem is that *even if the sought-after behavior is produced, we still need to work out why it behaves that way.* That is, a rule-based approach often ends up replacing a world we don't understand with a model we don't understand. In this regard, the modeler is in the same position as an experimentalist, but with the much easier task of dealing with a system about which everything at the micro-level is known, allowing one to focus on how the micro-level gives rise to the observed macro behavior. An advantage of rule-based models is the ease by which the assumptions underlying such models can be altered and the consequences explored.

Rule-based models are ideally suited for mechanistic causal analysis, that is, for identifying the chains of events (rule applications) that were necessary in obtaining a particular result. The formalization of a biologically meaningful notion of mechanistic causality that permits extraction of pathways, or workflows, from a simulation of a rule-based model is a challenging subject of ongoing research [12]. Systems biologists utilize simulation for a range of applications, but broadly speaking, high-level goals for using rule-based modeling to simulate complex systems include the following goals, related to accessibility, execution, and understanding [9, 17, 38]:

G1 Understand how a multitude of heterogenous and interacting agents can give rise to coherent system behavior;
G2 Utilize dynamical models for causal explanations of biochemical interactions;
G3 Provide effective visual analytics methods for investigating simulation outputs of rule-based models.

Protein-protein interaction networks with hundreds to hundreds of thousands of individual elements can be difficult to represent. However, visualizing the rules that govern these elements and their interactions with each other or with other elements provides a more concise way to represent the system. This highlights the salient aspects of the system, while at the same time accurately aggregating the lower-level elements.

Through our survey of the relevant literature in systems biology and information visualization (see Sect. 2), along with in depth conversations with domain experts in rule-based modeling, we articulate the main visual analytics tasks that an interactive visualization of dynamic rule influences should support:

T1 Identify influence patterns between rules and within groups of rules;
T2 Visualize the dynamics of rules as they change over time;
T3 Generate hypotheses about the relationship between local interactions and their participation in global behaviors;
T4 Annotate components of the dynamic system in order to highlight salient features and share them with others;

We introduce a new visual analytics technique, the Dynamic Influence Network, or DIN for short, to aid in the understanding of how coherent system behavior emerges from a set of rules. Additionally, we present a software tool, *DIN-Viz*, that provides systems biologists with the means to visually examine dynamic data collected during simulation.[1] The paper thus presents the following contributions:

- We introduce a dynamic network representation of rules and rule influence (Sect. 3.1);
- We provide a web-based software application, created in JavaScript using the D3.js library [7], for interactively exploring and analyzing Dynamic Influence Networks (Sect. 3.2);
- We present a thorough explication of a use case that clearly articulates the various strategies a systems biologist uses to make sense of a rule-based model for a complex dynamic system (Sect. 4).

Additionally, Sect. 2 discusses relevant related work from biological modeling, biological data visualization, causality visualization, and dynamic graph visualization, and Sect. 5 presents user feedback from domain experts in systems biology and molecular biology.

2 Related Work

Visual analytics often involves mitigating the complexity of datasets that describe systems that are too large or too complicated to reason about without external representation. This is true as well for visualization projects that aim to explore biological pathways. Instead of representing individual proteins or biological elements, which could number in the tens or hundreds of thousands for a single cell, or even

[1] *DIN-Viz* is open source software freely available at `https://github.com/CreativeCodingLab/DynamicInfluenceNetworks`.

the types of elements, which could also require a large number of nodes, our approach visualizes the relevant *rules* that govern the interactions between elements. Since most systems have many less rules than elements or element types, this decreases the amount of information that needs to be represented (although issues related to uncertainty, incompleteness, and complexity can still occur with rule-based approaches). Moreover, this information is potentially more useful for gaining insight into how a system works, and for validating useful models that can guide research into biological pathways.

2.1 Biological Modeling

While the majority of pathway visualization tools apply to reaction-based modeling, our project introduces a visual analysis technique for rule-based models. In addition to the *Kappa* language described above, the *BioNetGen* language [23, 31] is also widely used by systems biologists. While the differences between these languages are minimal, one benefit of *Kappa*, according to Wilson et al. [64], is that tools in the *Kappa* ecosystem make use of formal methods to aid in information discovery and in debugging models.

A modeling framework called *PySB* aims to make it easier to build mathematical models of biochemical systems as Python programs [42]. In their approach, models are not only created using programs, these models are already executable programs. *PySB* transforms the Python code into either *BioNetGen* or *Kappa* rules, and provides methods that make it easier to create macros that encode recurrent biochemical patterns and to define complex networks as reusable modules. Pedersen et al. [53] also introduce a modular extension to *Kappa* that provides a means for writing modular rule-based models.

While our technique was created using *Kappa* rules and *KaSim* outputs, it should be straightforward to extend it to other rule-based modeling languages or high-level modeling frameworks.

2.2 Dynamic Network Visualization

Efforts to effectively visualize graphs with nodes or edges that represent temporal data or that have a topology that evolves over time are cataloged by Beck et al. [5]. They survey the landscape of dynamic graph visualization, categorizing projects primarily in terms of how they represent time, that is, whether or not they use animation or a static timeline to show the evolution of networks. These categories are then further parcellated according to which layout strategies they utilize and how they address particular problems inherent in dynamic datasets. Inspired by Moody et al. [46], who investigate animated network "movies" for a range of sociological datasets, our tool features an animated node-link diagram whose layout is determined by clusters of influence (measured by how likely rules are to fire at the same time), either on a per-frame basis or within a user-selected time window. An interactive timeline is used to navigate through time, and more detailed information about selected nodes is presented for the currently selected time period.

Interesting recent approaches to visualizing dynamic data include Archambault and Purchase's work on dynamic attribute cascades [1], Mashima et al.'s *GMap* [45], and Bach et al.'s *GraphDiaries* [2] and *Matrix Cubes* [3] techniques. However, since a main goal of our visualization was to emphasize the relationship of rules to other rules, we elected to use a visual representation that made it easier to apply visual encodings to the links between nodes (see Sect. 3). Techniques by Ma et al. [43, 44], Purgato et al. [55], and Ye et al. [66] present multiple synchronized representations of a dynamic brain network to provide additional insight into the community dynamics within the network. Our tool also presents auxiliary representations to support the analysis of dynamic data, providing detail on demand for selected nodes.

Vehlow et al. [59] provide a thorough overview of different approaches to grouping data within graphs. A taxonomy of methods categorizes groups as juxtaposed, embedded, superimposed, or encoded using visual node attributes. Our technique utilizes superimposition, providing colored clusters as a way to show group membership of nodes that are similar, as well as visual node attributes, enabling a user to apply coloring to indicate a secondary grouping of nodes. Hadlak et al. [30] present a network layout in which each node represents a cluster, and contains a time plot providing an overview of the temporal trend of the cluster, as well as a secondary view that shows time series data describing changes to a selected cluster. Our software tool also allows the user to examine more detailed information about temporal trends within the network.

2.3 Causality Visualization

Elmqvist and Tsigas [22] introduce the Growing Squares and Growing Polygons techniques to explore causality, finding that they are significantly more useful than static graphs or Hesse diagrams for reasoning about systems. In these techniques, differently sized shapes are used to indicate information flow in a system of interacting processes, filling with different colors to represent the changing influence of the different processes.

Ware et al. [63] explore the use of visual causal vectors to indicate causal relationships between data elements, and Bartram and Yao [4] utilize animated causal overlays in order to highlight causal flows and to indicate the relative strength of the causal effect. Kadaba et al. [33] also find that the use of animation is superior in terms of both accuracy and speed in comparison to static representations for facilitating comprehension of complex causal relations.

More recently, Dang et al. [14] introduce *ReactionFlow* to highlight the inherent causality in biological pathways. In this technique, different reactions are presented in a column at the center of the screen, and a user can select any reaction, or any protein involved in a reaction, and play an animation that indicates which other reactions could occur downstream. Zhang et al. [67] introduce the *ReView* tool for visualizing traffic causality and to reason about the origin of network traffic anomalies that may be indicative of suspicious requests. Also intended to help determine root causes of events (although in an entirely different domain), Vigueras and Botia [61] introduce an algorithm for defining causality graphs for debugging multi-agent systems in order to track the causality of events produced by interactions among agents in a group. Forbes et al. [27] also investigate the use of an animation framework to represent simulations of dynamic systems. Our technique aims to provide systems biologists with the ability to more effectively reason about the causal mechanisms within protein-protein interaction networks.

2.4 Biological Network Visualization

Murray et al. [47] survey biological pathway visualization projects and introduce a taxonomy of visualization tasks for the analysis of biological pathway data. Relevant tasks are organized in a high-level categorization as *attribute* tasks, *relationship* tasks, and *modification* tasks. Interestingly, simulations of rule-based models are not explicitly discussed. However, this taxonomy does describe tasks that are relevant to visualizations of rule-based models, and our approach in particular, such as *grouping* [49, 50, 60], *annotation* [13, 15, 20, 28, 41], and *causality* [24, 39, 51].

A range of projects introduce techniques for visualizing rule-based models of protein-protein interaction networks. Danos et al. [16] discuss the use of *contact maps* to visually represent stochastic trajectories for user-defined observables in order to tell a "story" that summarizes how a given event type can be obtained. Kohn et al. [37] introduce *molecular interaction maps* that explicitly define the topology of rule-based networks, and that can be used for simulating the interaction of molecular rules. Inspired by these efforts, as well as by the earlier process diagrams of Kitano et al. [36], Chylek et al. [11] provide guidelines for visualizing and annotating rule-based models using interactive *extended contact maps* that represent a cell signaling system so that it is both visual and executable.

Smith et al.'s *RuleBender* [56] provides a framework for editing and exploring rule-based systems of intracellular biochemistry, such as *BioNetGen*. In this framework, the primary visualization provides an interactive contact map in which molecules are rendered as large gray nodes and domain states are positioned inside these nodes. Rules are represented as links between specific sites within the nodes, and nested isocontours are used to define a compartment hierarchy of elements related to particular components of the cell. A secondary visualization

shows the relations between the reaction rules that describe the behavior of a system. That is, similar to our approach, the influence graph shows if a rule activates or inhibits another rule. However, our approach further emphasizes how these influences can change over time, facilitating the analysis of the dynamics of the system.

Gostner et al. [29] provide an overview of graphical modeling software tools for representing or simulating reaction-based models, and detailing usability and perceptual issues that they can introduce. They find that creating graphical languages with many glyphs reduces ambiguity, but at the cost of introducing visual clutter and making the visual layout unappealing to users. They advocate for a minimalist approach that presents only the visual elements necessary for a particular analysis. Our tool also uses a smaller visual language for representing rules and clusters of rules.

Pedersen et al. introduce *Bio Simulators* [52], a web-based framework that uses *KaSim* to run simulations of *Kappa* rules. The visualization output is a simple chart that shows the population of predefined "observables" within the system, that is biological agents (e.g., proteins or protein complexes) that are affected by the rules. This visualization provides an overview of the system, but does not indicate specifically which rules are responsible for these changes in observables, nor provide insight into how the activity of rules affects other rules. Our *DIN-Viz* tool provides this type of visual output in a secondary panel, providing an alternative perspective of the system.

3 Dynamic Influence Networks

In this section, we introduce details of the Dynamic Influence Network and our visual analytics software tool, *DIN-Viz*. The *DIN-Viz* interface presents two interconnected view panels, the Network Panel and the Data Panel. The Network Panel provides an overview of the influence the rules have upon each other, and the Data Panel provides additional detail both about selected individual rules, as well as an overview of the entire system. These views can be temporally navigated using an interactive timeline. Interactive thresholding defines which nodes and links are visible, and pop-up option panels enable users to emphasize different characteristics of the network and to toggle between different clustering metrics.

By using *DIN-Viz*, researchers can engage in each of the visual analysis tasks defined in Sect. 1.2. Specifically, our interface aims to facilitate hypothesis generation and validation. We discuss this in detail in Sect. 4, where we present a walkthrough of an analysis of a complex biological system. Briefly, a user is provided with an overview of the dynamic system in which rules that are tightly coupled are grouped together in clusters (**T1**). A user can then create custom clusters of specific rules, arranging them in close spatial proximity in order to evince meaningful behaviors at given times (**T2**). For example, a user may be interested to learn if particular rules fire together, independently of whether or not they influence each other. A user can then temporally traverse the dynamic system in order to identify particular times where this hypothesis about synchronized rule firing is true or not true (**T3**). The user can apply annotations (labels and color cues) to identify relevant features of the system in order to present this information to others (**T4**).

DIN-Viz was created through a collaborative design process, in which we received expert feedback from our systems biology collaborators over the course of a six month development period. During this time, many iterations of the application and visual encodings were developed and refined. As the software became more adept at representing rule-based models, new datasets were provided that initially challenged our application, both in terms of the robustness of our implementation and the utility of our design choices, forcing us to rethink some aspects of our approach and to extend others. Through this process, *DIN-Viz* has become increasingly effective at representing large rule-based models in order to facilitate analysis tasks that utilize Dynamic Influence Networks.

3.1 Defining the Dynamic Influence Network

Before describing further details of our visualization technique, we introduce a formal definition of *influence* in relation to *Kappa* rules.

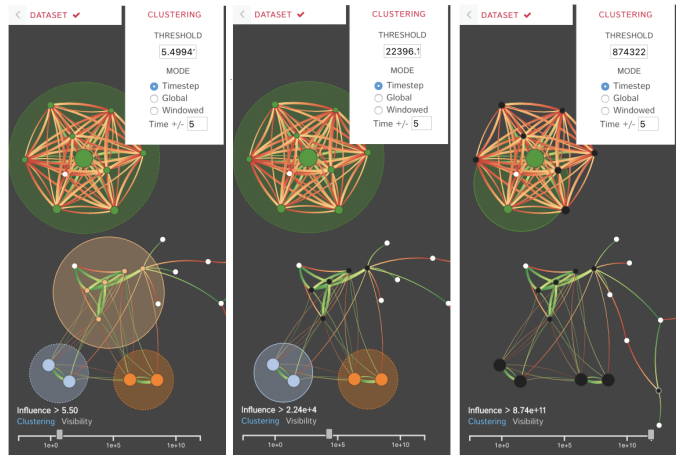

Fig. 2. These screen captures show views of the DIN at the same time step, but each using three different clustering thresholds. As the user increases the clustering threshold value (left to right) using the slider at the bottom-left of the Network Panel, fewer and fewer clusters are created. In the pop-up menu, a user can enter a precise clustering threshold and choose the clustering window options.

Let s be a rule

$$s: \quad \mathscr{L}_s \longrightarrow \mathscr{R}_s \qquad @\ \gamma_s \tag{1}$$

with $\mathscr{L}_s$ and $\mathscr{R}_s$ *Kappa* patterns, and γ_s a rate constant. The activity α_s of s indicates the propensity of s to induce the next state transition in a mixture of fully specified molecular species $\mathscr{M}$ (the system). The probability that s induces the next transition is simply the relative propensity, $\alpha_s / \sum_r^N \alpha_r$. Moreover,

$$\alpha_s = \gamma_s \cdot |\{\mathscr{L}_s; \mathscr{M}\}| \cdot \frac{1}{\sigma_s} \tag{2}$$

where $|\{\mathscr{L}_s; \mathscr{M}\}|$ denotes the number of matchings of $\mathscr{L}_s$ in the mixture $\mathscr{M}$ and σ_s the number of symmetries preserved by s. The term $|\{\mathscr{L}_s; \mathscr{M}\}|/\sigma_s$ is the number of physical configurations that are distinguishable with respect to the mechanism expressed in s. The activity α_s of a rule corresponds to the conventional chemical kinetics notion of flux (velocity) through a reaction.

The Dynamic Influence Network (DIN) is a directed graph whose nodes are rules. A directed edge from rule r to rule s has a weight that reports the contribution of events due to rule r (r-events) on the activity of rule s over some fixed time interval $[t, t+\tau]$. Rule r affects the activity of s by either generating or destroying matchings of $\mathscr{L}_s$ in $\mathscr{M}$. Note that in a system of reactions (rather than rules), any firing of the reaction $\mathrm{A} \longrightarrow \mathrm{B}$ has an immediate impact on the activity of the reaction $\mathrm{B} + \mathrm{C} \longrightarrow \mathrm{D}$, if the system already contains nonzero amounts of C. In a rule-based system, however, a firing of rule $r: \mathrm{A}(\mathrm{x}_u) \longrightarrow \mathrm{A}(\mathrm{x}_p)$ need not have an immediate impact on the activity of $s: \mathrm{A}(\mathrm{x}_p, \mathrm{y}_p) \longrightarrow \mathrm{A}(\mathrm{x}_p, \mathrm{y}_u)$. When r fires on an instance $\mathrm{A}(\mathrm{x}_u, \mathrm{y}_u)$ in the mixture $\mathscr{M}$, it has no immediate impact on the activity of s; it does, however, when it fires on $\mathrm{A}(\mathrm{x}_u, \mathrm{y}_p)$.

Let i index the sequence of events that occur in $[t, t+\tau]$. The ith event causes a transition of the system from state M_i (before the event) to state M_{i+1} (after the event). If the ith event is due to an application of r, the contribution of r to the *relative change in activity* of rule s is given by

$$\Delta_i(r \rightsquigarrow s) = \begin{cases} \dfrac{\alpha_s(i+1) - \alpha_s(i)}{\alpha_s(i)} & \text{if } i \text{ is due to } r \text{ and } \alpha_s(i) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

An alternative is to replace the activities α_s with firing probabilities $p_s = \alpha_s / \lambda$ with $\lambda = \sum_q \alpha_q$ the system activity:

$$\Delta_i(r \rightsquigarrow s) = \begin{cases} p_s(i+1) - p_s(i) & \text{if event } i \text{ is due to rule } r \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

We aggregate these contributions during the interval $[t, t+\tau]$ and divide by the number $\#r$ of events that were due to r to obtain the average influence of r on s during that interval:

$$\langle \Delta_t(r \rightsquigarrow s) \rangle_\tau = \frac{1}{\#r} \sum_i \Delta_i(r \rightsquigarrow s) \text{ with } i \in \{ j \,|\, \text{time}(j) \in [t, t+\tau] \}, \quad (5)$$

where $\text{time}(i)$ is the time stamp of the ith event. We shall refer to the matrix or graph defined by (5) and using (3) as the activity-DIN, or aDIN, and to (5) using (4) as the probability-DIN, or pDIN.

An edge in the aDIN is easy to interpret: if at time t the firing of r produces patterns to which s is applicable, the a-influence is positive; if it alters (and thus removes) patterns to which s could have applied, it is negative. The interpretation of the pDIN is more subtle, since an r-event may affect the firing probability of s not only by producing or destroying matchings of the pattern $\mathscr{L}_s$, but indirectly by affecting the system activity $\lambda = \sum_q \alpha_q$ through influencing the activities of rules other than s.

The definition (5) is, informally, reminiscent of a derivative $d\alpha_s/dt$ in a continuum setting, if only we could take the limit $\tau \to 0$. In general, one cannot write a system of differential equations for the evolution of the rule activities in terms of only the $\mathscr{L}$ and $\mathscr{R}$ patterns of the rules. One might therefore think of the DIN as a purely observational version of the elusive formal system of interdependencies between rule activities. We believe however that the DIN is useful in an "empirical" approach towards understanding a rule-based model.

3.2 The *DIN-Viz* Software Application

DIN-Viz features two main view panels: the Network Panel, which presents an interactive dynamic network of rules (nodes) and the amount of influence between the rules (edges); and the Data Panel, which presents detailed information about selected rules. Fig. 1 shows an overview image of *DIN-Viz*, with the Network Panel on the left and the Data Panel on the right.

In the Network Panel, the DIN is presented as a node-link influence diagram, leveraging a force-directed layout [21] to position related nodes and clusters near to each other. The nodes and links directly represent the rules and the influence of each of the rules comprising the DIN, and the visualization enables the topological analysis of the *KaSim* simulation data at the time step scale. By mapping the rule-to-rule influence to the link strength, highly mutually influential rules are closely grouped together visually. This provides insight to the systems biologist as to which rules are likely to be indicated in the formation of biological pathways. In order to further facilitate the analysis of these highly related rules, we perform a clustering operation on the network based on the influence between rules and generate groups with a high inner influence, based on a user-selected threshold.

In the Data Panel, we provide additional views that provide a temporal context for the graph representation. The upper time-based chart is the phenotype line graph, which charts *KaSim* "observables" over the course of the simulation. The lower time-based graphs chart the incoming and outgoing influence of user-selected nodes.

3.2.1 Network Panel

To convey the density of information necessary for the accurate analysis of the data, we encode both data that is produced directly by the simulation and data derived from that simulation. The derived data include rule clusters, averaged influence values (when clustering globally or within a time window), and interpolated values during animation.

To visualize the system and its data, we use a force-directed layout of a node-link diagram, where nodes encode the rules, and links encode the influence between two rules. The absolute value of influence between rules, whether defined by activity or probability (see Sect. 3.1), determines the attractive link forces in the network. *DIN-Viz* maps the number of hits of a rule to the node size, and the influence of one rule on another through the link width. We indicate the directionality of a link and the sign of the influence by using two directional gradients: yellow-to-red for negative, and yellow-to-green for positive. Alternatively,

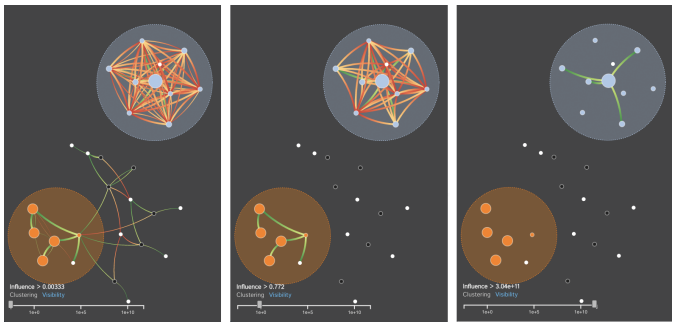

Fig. 3. Links may be interactively filtered on demand using the visibility slider (bottom-left of each image). With all links visible (left), the light blue cluster is very highly interconnected. Increasing the visibility threshold (middle) causes the free nodes to have no visible links, showing that they do not have a strong relationship any nodes at this time step. At the highest visibility threshold (right), only 4 links remain, all connected to the node in the center of the light blue cluster. These important relationships are difficult to see when more links are visible (left, middle).

a colorblind safe mode is available that replaces the yellow-to-green positive colormap with white-to-blue.

Details-on-demand for the links show the source and target, as well as the exact influence value, and for the rules show the rule name, self-influence, and the top incoming and outgoing influences to the rule. The user may also choose to make visible the names of the rules as labels, either for all rules, or for interactively selected rules.

By default, we use curved lines in order to represent edges. We rely on clustering for immediate spatial analysis and data-on-demand for more in-depth quantitative information, introducing curvature only as a minimal intervention to help differentiate multiple edges sharing source and target nodes. The use of curves does not appear to detract from the intensity gradient used to encode direction [32]. However, the literature on graph perception indicates that curved edges can hinder readability [65], so we provide an option for users to toggle between curved or straight edges.

Rule Clustering We derive cluster assignments for the rules based on their influence upon one another. To form these clusters, we extract all links from the network. If a link is above the user-determined clustering threshold, its source and target rules are considered to be clustered together. If nether is currently associated with a cluster, they are placed together in a newly created cluster. If one is in a cluster and the other is not, then the unclustered node joins the cluster. Lastly, if both nodes currently belong to different clusters, these two clusters are joined together.

Onto these derived clusters we map a categorical color scheme to distinguish between clusters. We double encode the cluster assignment through the node color as well as underlaying clustered nodes with a bounding circle of the cluster color. Double encoding is used to emphasize the relationship between nodes; in our iterative design process, we discovered that simply changing the color of nodes did not sufficiently highlight them, especially inside of dense clusters.

Clustering is helpful for grouping rules together and aids the user in understanding how rules operate together. However, these simulations can have influence values spanning orders of magnitude. To handle diverse simulations, the user can fine-tune the clustering threshold based on the nature of the system, as well as the interactions they would like to highlight. In medium-sized or smaller networks (with less than 500 nodes), the clustering computation can be performed at real-time rates on a consumer laptop. The responsiveness of the clustering method provides strong interactivity with the visual analytics tool.

The initial clustering mode is per time step, where the clusters will be computed using links from the current time step. This operation yields very precise groupings of rules based on their current states. The clusters can be very volatile for certain systems, and a node could potentially switch between clusters at every time step. To provide an alternative to this, we include two additional clustering modes: global

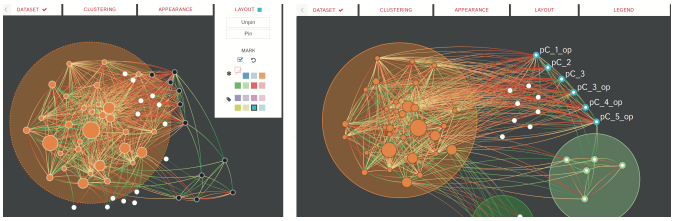

Fig. 4. Influence networks can become very complicated (left). Through pinning and marking, the user may apply custom spatial and categorical groupings that serve to unclutter as well as annotate the rules. After reorganization (right), rules pC_* are marked and labeled, as positioned near to one another. The white nodes below and to the right of the orange cluster (left) have been pinned outside of the orange cluster with two new marked clusters being created, green and light-green (right). A pop-up menu provides options for the pinning and marking features.

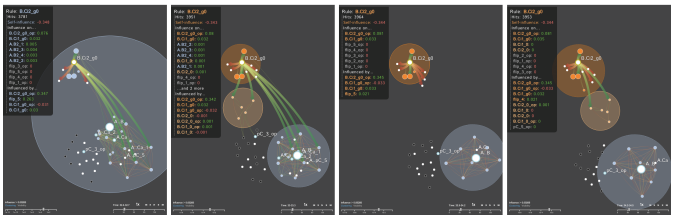

Fig. 5. These four screen captures show the changes in rule influence as the user steps through the simulation in the Network Panel. Here, the clustering of rules changes as each rule's influence increases or decreases, and the incoming and outgoing influences of a rule chosen by the user (here the rule*B.Ci2_g0* is selected) are indicated both by the highlighted edges and the pop-up panel in the upper-left corner of the visualization. Additionally, a dashed indicator marks the corresponding time in the line graphs (See Fig. 1), linking network and time-based contexts together. An additional controller below the graphs allows the user to zoom into a smaller time window for a more detailed view.

and windowed. These modes can smooth the clustering operation when moving backward or forward in time, and can help the user to highlight trends across time.

Global clustering averages the influence that all nodes have on one another over all time steps and transforms these average influence values into the links used to cluster nodes. Globally clustering the rules groups them by their overall behavior, not just at one specific instant. A downside to this approach would be a link which fluctuates perfectly from a value x to $-x$ over the course of the simulation, resulting in an average influence of 0. However, the clusters fail to capture the more granular behavior may when attempting to group rules globally. As a middle ground, we implement a time-window clustering that smooths volatility while preserving detail. The user can specify the number of time steps that the clustering will use. For example, if the user selects a window of 10, the influences are averaged over 21 time steps $[t-10, t+10]$, including the current time step. This method preserves the local behaviors while smoothing the cluster assignments over time.

By changing the clustering window, the user can analyze the data at different time scales. Fig. 2 shows a view of the DIN at the same time step, but using three different clustering thresholds. As the user increases the clustering threshold value (using the slider at the bottom-left of the Network Panel), fewer and fewer clusters are created. In this figure, the user has selected a time window of 5, so that clusters are determined through averaging influences over the 5 previous and 5 subsequent time steps.

Interactive Filtering We also allow for the interactive filtering of links by their influence value to help handle larger datasets. By removing the edges below a user defined influence value, the graph is less visually cluttered, and the edges with higher influence values become prominent, whereas they could otherwise be obscured. Fig. 3 shows an example where a user increases the visibility threshold. As the user increases the visibility threshold value (again, using the slider at the bottom-left of the Network Panel), more and more edges are removed from the visualization. For example, in the rightmost frame, the four positive influences are clearly seen within the larger gray cluster once the negative edges with less influence are removed.

Marking and Pinning While force-directed layouts help to mitigate visual clutter in node-link diagrams, dense networks can still be difficult to make sense of— an issue that is exacerbated when representing large datasets. *DIN-Viz* includes an option whereby the user can manually creates a layout of nodes or entire clusters through relocating and "pinning" them to specified locations in the Network Panel. This reorganization reduces clutter, but also helps users to distribute the rules and clusters in a way that is cohesive with their thought process during exploratory analysis. When pinned, the spatial positioning and grouping of selected rules and clusters is preserved over the course of the entire simulation, overriding the normal layout behavior.

While the mathematically defined clusters help capture groups of rules with a strong influence on one another, there may be rules which are related through their behavior but lack a strong influence with one another. To solve this problem, we implement a "painting," or marking, interaction. The user can mark a set of nodes with the same color to give them a categorical grouping, inserting them into an existing or newly created painted cluster. Alternatively, nodes can be painted and labeled, grouping these rules visually, but only with color, not placed within a cluster. These categorical groupings, similar to the spatial groupings achieved through pinning, aid in the logical organization of rules during analysis by the user.

Through pinning and marking actions, users are aided in the exploration of the dataset, leading to the formation and evaluation of hypotheses. Additionally, pinning and marking, along with displaying the labels of rules, serve as tools to annotate the data and prepare it for presentation. Grouping rules both spatially and categorically can help to emphasize specific interactions and biological pathways. Fig. 4 shows an example of how the use of marking and pinning can be used to effectively reorganize a dense influence network to highlight specific sets of rules. The operations of labeling, cluster windowing, cluster thresholding, visibility thresholding, pinning, and marking, can all be used in combination to articulate components of the network that are relevant to a particular analysis session.

Animation To represent the dynamism of the system, animation is used to update the influence between the rules, which in turn updates the edge weights, node sizes, and cluster definitions. A time slider controls the current time step, enabling the user to move through time or to jump to a particular time step. We also include standard playback controls that animate the simulation so that the user can observe changes in the DIN over time. The user can play and pause the animation, as well as speed up or slow down the animation. The *KaSim* simulation data provides "snapshots" at particular time intervals. Playback across longer intervals could cause confusing jumps where the network suddenly leaps to a completely new configuration. While this may be important to visualize (and encourage the systems biologist to re-run the simulation at a greater temporal resolution), even less dramatic shifts can obscure the dynamics of the system. To improve the temporal coherence as we animate between frames, we provide an option where we perform a linear interpolation of influence values between two sequential time steps [6, 62]. Fig. 5 demonstrates the evolution of a DIN over time, depicting how the clusters of rules emerge and disappear as the rules' influence increases and decreases.

3.2.2 Data Panel

Although the use of animation helps to facilitate insight into the dynamics of the system, it can make it difficult to keep track of detailed differences between time-steps, hindering comparative analysis and increasing cognitive load [5, 57]. To address this problem, we supplement the Network Panel with the Data Panel.

The Data Panel provides the explicit temporal context missing from the Network Panel, enabling a thorough multi-level analysis of the data. The Data Panel consists of 3 line charts, graphing information dependent on time. The Phenotype Chart graphs observables— populations of particular biological elements— from the simulation over time. This view assists in the analysis of global trends in the simulation for the entire system. The additional Rule Influence Charts display the incoming and outgoing influences of a user-selected node. The Rule Influence Charts show global trends in the simulation for a single rule, rather than the entire system. Self-influences are denoted by a red line, while influences to and from other rules are denoted by gray lines. Hovering over these lines shows the corresponding time step and influence values while highlighting the line itself. By using the three charts in the Data Panel, system trends over time may be analyzed at both a global and local level simultaneously. Located at the bottom of the Data Panel is the time selection slider, which enables the user to zoom the time axes of the Phenotype and Rule Influence Charts. The right side of Fig. 1 shows the Data Panel.

4 Case Study

To evaluate the DIN technique, we present a detailed use case in which a systems biologist examines the dynamics of a complex protein-protein interaction network. The basic usage practice consists in: (i) Visualizing the dynamics of the influence network and identifying clusters of rules, determined by mutual strength of influence, regardless of whether influence has a positive or negative sign; (ii) Identifying temporal sequences of activation and inhibition between rules or rule clusters; and (iii) Using these temporal "story lines" to connect the mechanisms laid down in the rules with a specific macroscopic behavior of the system. For our use case, this practice occurred in iterative steps, as, for instance, an identification of an interesting sequence (ii) could lead to the need to re-conceptualize which rules were most relevant (i), or a hypothesis about the meaning of a story line (iii) could require a change to the pinning or thresholding of rule clusters in order to reveal potentially meaningful dynamics (ii).

4.1 The KaiABC Oscillator

Many organisms control physiological processes in a pattern that anticipates daily changes in light levels. Such circadian rhythms are maintained by molecular networks that exhibit autonomous oscillations that directly involve or are coupled to gene regulation. A widely studied system is the circadian oscillator of the cyanobacterium *Synechococcus elongatus* [48]. This system is remarkable because its key components are just three proteins— KaiA, KaiB, and KaiC— which have been isolated and reconstituted in the test tube, where they maintain the oscillating behavior under consumption of chemical energy (ATP) [58]. Though this rule-based system is relatively small, it exhibits many of the complexities found in much larger models.

1. The protein KaiC is a complex of six identical units each of which can be reversibly phosphorylated (marked with a phosphate group) independently of the others, with dephosphorylation (unmarking) occurring more readily than phosphorylation (marking). The hexamer as a whole (henceforth simply KaiC) has therefore 6 sites and $2^6 = 64$ possible phosphorylation states, but we distinguish only 7 overall phosphorylation levels (p-levels) $p \in \{0, 1, \ldots, 6\}$.
2. KaiC is assumed to switch between two conformational states, termed "active" (A) and "inactive" (I). Importantly, the probability (i.e. the rate constant) of a conformational flip from A to I increases with the p-level of KaiC.
3. KaiA binds KaiC in the A-form, but the interaction strength (more specifically: the average time that KaiA stays bound to KaiC) decreases rapidly with increasing p-level. When bound, KaiA prevents conformational flipping and promotes the phosphorylation of KaiC. In the model, this is done by upping the phosphorylation rate and preventing dephosphorylation. In sum, KaiA sticks to KaiC in the A-form at low p-levels, cranks up the p-level, and by doing so kicks itself off KaiC. When KaiA unbinds from KaiC at high p-levels, KaiC flips back into the I-form, which KaiA cannot bind. At this point the dephosphorylation tendency takes over, leading to a decrease in the p-level.

At low numbers of KaiA, (1)-(3) induce individual KaiC molecules to move up and down p-levels rather randomly and without coordination among each other. As a result, the population of KaiC molecules settles on a steady level of average overall phosphorylation.

4. Coordination between KaiC proteins is achieved by a third agent, KaiB, which reversibly binds KaiC in the I-form. When bound, KaiB locks KaiC in the I-form, thereby allowing dephosphorylation to proceed. Importantly, bound KaiB also strongly (but reversibly) binds KaiA. The probability that KaiA binds KaiB, when the latter is bound to KaiC, is maximal at intermediate p-levels (but the time it stays bound is independent of p-level). While bound in this way, KaiA can no longer interact with any KaiC. Hence, KaiB promotes dephosphorylation of the KaiC it is bound to and holds back phosphorylation of other KaiC molecules by sequestering KaiA. These interactions statistically synchronize the phosphorylation and dephosphorylation cycles on individual KaiC molecules, resulting in collective p-level oscillations of the KaiC population.

When translating these mechanistic descriptions (some of which are empirical findings and some of which are hypotheses [58]) into *Kappa*, we obtain 57 rules, whose dynamical action is simulated using *KaSim*. If the same model were based on a description in terms of molecular species instead of patterns, it would require 707 species and 13090 reactions. In the present case, KaiC is represented as an agent with 6 phosphorylatable sites, because *Kappa* is (deliberately) not designed to express the idea of equivalent sites or the notion of a phosphorylation "level." However, we can choose to automatically detect such equivalences (and hence implicitly the notion of a phosphorylation level). In so doing, the number of species shrinks to 59 and the number of reactions they participate in turns out to be exactly 1000. Even in relatively simple cases as this one, the rule-based approach provides a hugely compressed description that is lossless with regard to dynamics.

4.2 Using DINs to Understand the KaiABC Clock

After spending an initial period of time becoming familiar with the overall dynamics of the system, we then begin a more focused visual analysis to identify important features, or "story lines," of the KaiABC clock that are made apparent using *DIN-Viz*. Fig. 6 depicts a series of time steps from this analysis session and is referred to throughout this section.

We proceed first by turning off visual clustering, working only with the default structure imparted by the force-directed layout. We also turn off all edges, leaving only unmarked nodes. Running the DIN animation in this terse format allows quick identification of which nodes (rules) are potential drivers by paying attention to (a) the evolution of node size, which reflects the number of firings in the corresponding time interval (displayed on the lower-right), and (b) identifying nodes that move abruptly or rapidly, as such nodes appear to signal changes in the influence pattern. A case in point is the node `A.Ca` in the timeframes $[64.6 - 64.7]$ to $[64.7 - 64.8]$. Around $t = 64.6$, `A.Ca` had no firings but received a high positive influence from rule `A..B`. At $t = 64.7$, `A.Ca` starts firing and suddenly connects with a host of other nodes, as can be seen when turning on the edge display. Using the full graph display is, of course, useful too, in particular when keeping an eye on the trajectory of the systemic property shown in the right panel of the browser. It is also informative to turn on and off all inhibitory edges (red) or all activating edges (green).

The next step consists in marking and pinning nodes that appeared salient in the first pass. It proves useful to rerun the animation before pinning in order to determine any cluster structure among the salient nodes, which would suggest a division of the screen area into regions dedicated to these clusters. It is important to keep in mind that cluster structure is influence-based and therefore likely transient. The purpose of pinning is to fix a "coordinate" system of salient nodes against which to assess the tug of war of influence over the remaining mobile nodes.

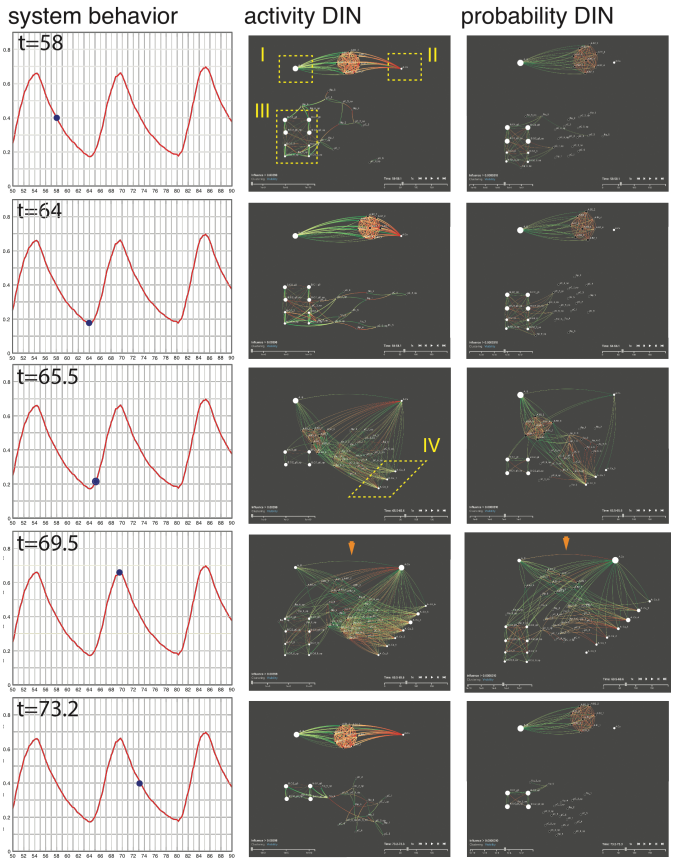

Fig. 6. This figure shows a series of snapshots taken during an analysis session of the KaiABC clock, discussed in Sect. 4.2. The left column shows the current time step and the global phosphorylation in the system (the y-axis) across the simulation session, with a point indicating the particular time step (the t-axis). The middle column presents snapshots of the DIN where influence is defined using the relative change in activity (the *aDIN*). The right column presents snapshots of the DIN where influence is defined based on the firing probabilities of rules (the *pDIN*).

If the pinned nodes are indeed salient, they should engage in distinct patterns of activity that organize the mobile nodes. Tuning the visual clustering (by setting the cluster threshold) can be useful to determine cohesive structure among the mobile nodes. Note that here pinned nodes are excluded from such visual clustering.

Fig. 6 exhibits our layout for the KaiABC clock. In this model several rules have a natural family structure. For example the `A..Ca_x` rules represent the dissociation of A from an active C at a rate constant that depends on the p-level of the C-agent. These rules are all expressing the same basic transformation (unbinding of A from C) but their refinement according to p-level is a critical part of the differential affinity mechanism [58]. Often, but not always, do such rules cluster naturally in the DIN. A modeler should therefore make use of any information he or she has about the nature of the rules to group them using *DIN-Viz*.

Once parts of the DIN are fixed, the state of the DIN needs to be put into correspondence with the macroscopic behavior of the system. For this purpose we supply a file with a time series of the property of interest (i.e., a Phenotype Chart). In our case, the systemic property is the overall fractional phosphorylation Y:

$$Y = \frac{\sum_i^n i|\mathrm{C(p{=}i)}|}{n|\mathrm{C()}|} \tag{6}$$

where $|\mathrm{C(p{=}i)}|$ is the number of C-agents with i phosphorylated sites, $|\mathrm{C()}|$ is the total number of C-agents, and n the number of phosphorylatable sites—in our case $n = 6$. The quantity Y can be viewed as the probability that any of these sites is phosphorylated. $Y = 1$ means that all C-agents are fully phosphorylated, and $Y = 0$ means that none has even a single phosphorylated site. The left-hand column of Fig. 6 shows a few oscillations of $Y(t)$ with a dot marking the position in the cycle corresponding to the aDIN and pDIN snapshots on the right.

The top row ($t = 58$) shows 3 of the 4 anchor clusters of nodes we select by observing the animation as described above. While the initial confidence in these selections is supported by detailed knowledge of the model, we believe that someone without knowledge of the model would have discovered the same selection. First, the selected nodes are conspicuous in the temporal evolution of their firing rate and, second, the mobile nodes are seen to be meaningfully organized by the influence behavior of these anchors. The latter is an after-the-fact property that may require some iteration, which this interactive approach to DIN visualization is designed to facilitate. Anchors I and II (Fig. 6, middle) are the single rules `A..B` and `A.Ca`, respectively. Rule `A.Ca` is central to the processes described in item 3 of Sect. 4.1, while rule `A..B` and the anchor cluster III are central to those described in item 4. In the down-leg of the system cycle, many of the C-agents are in a state where they are being dephosphorylated, as they are locked into the inactive conformation by the binding of B (anchor III).

Interacting with anchor III is a cloud of mobile nodes containing rules named `pc_x_op`, which are dephosphorylation rules. (The `x` denotes the site on which the rule operates.) In addition, the cloud contains `flip_x` rules, which flip a C-agent from the active to the inactive state, promoting dephosphorylation. (The `x` in the rule name denotes the p-level of the agent, not a specific site.) Finally, the cloud also contains `pC_x` rules, which are phosphorylation rules. The animation makes sense of all this: One can see a "catching" motion where sites on C-agents are dephosphorylated, ultimately leading to a p-level of zero, turning on the `B.Cix_0_op` nodes, which effectively cause agents of type B to dissociate from agents of type C, initializing the phosphorylation phase. At the same time, `pC_x` rules fire and activate `flip_x` rules, indicating that some C are still in the previous up-leg of the cycle where they are reaching the peak and get switched into the inactive state by binding to B (rules `B.Cix_g0`, which fire at higher p-levels). This analysis of the "catching" motion upon animation (at any edge visibility threshold) simply indicates the flow along the down-leg of the cycle: A majority of the C population has entered the downward leg and is dephosphorylating, thus promoting eventual detachment from B, which locks the inactive state. However, another fraction of the population is lagging behind, still entering phosphorylation peak territory. The microscopic picture around anchor III enables these conclusions, although the systems level data simply indicate an overall declining p-level of the population.

According to the DIN, the processes that dominate at the midpoint of the down-leg have not changed much their interrelations when the system-level trajectory has reached the valley floor at $t = 64$. Consider now what is going on, in parallel, between anchors I and II at $t = 58$ or $t = 64$. As indicated in item 4 of Sect. 4.1, the phosphorylation of active C-agents is promoted by their binding to agents of type A, rule `A.Ca`, which is our anchor II. However, agents of type A are sequestered by agents of type B when the latter are bound to highly phosphorylated C. The point of this is to allow dephosphorylation to proceed by silencing the A-agents (which promote phosphorylation). But once C-agents have reached maximal dephosphorylation levels, they should enter the up-leg, for which the A-agents need to be awake. This is accomplished by releasing (unbinding) the A-agents through rule `A..B`, anchor I. The hairball in the middle is a cluster of mobile nodes that are about the binding of A to B, the opposite of what anchor I accomplishes. By definition of influence (Sect. 3.1), the firing of `A..B` has a positive effect on the binding rules `A.Bx` because it makes A available. In the downward leg of the cycle, `A..B` fires much more frequently (the large disk) than `A.Bx`, but the latter is still firing with appreciable frequency in this snapshot—presumably because of the laggards that are now reaching the peak and ready to bind B, lock in the inactive state and sequester A.

Rules `A.Ca` and `A.Bx` compete for free A, rendered by the red arrow from `A.Bx` to `A.Ca`. Yet, at $t = 64$ the negative influence is not

reciprocated, because `A.Ca` doesn't fire at all, and this despite the availability of activated C (or `A.Ca` could not be influenced). When stepping through the DIN at $t = 65.5$ and beyond, one notices that `A.Ca` never reciprocates the negative influence, even when it begins to fire. How can that be, if, from a static analysis point of view, `A.Ca` removes a free A from the system that could have also participated in `A.Bx`? The reason is that the binding rate constant of `A.Bx` is so high (both in the real system and the model) that once a B binds to a C, creating the condition for an A to bind that B, the event happens very quickly. There rarely is just a B bound to a C in the system. In other words: the rule `A.Bx` fires to immediately consume any pattern instance to which it is applicable. This creates the seemingly paradoxical situation of a rule that fires often and has almost always activity zero. A rule with activity zero cannot be negatively influenced; it can only be influenced positively, on the supply side, i.e. by `A..B`. The DIN shows that, at this point in the cycle, whenever `A..B` feeds the system with free A (green arrows oriented towards `A.Bx`), `A.Bx` makes sure nothing of that reaches `A.Ca` (red arrows from `A.Bx` to `A.Ca`).

The DIN panel at $t = 58$ shows two parallel mechanisms at work in the downward leg of the Kai oscillator. (i) The binding of B to C locks C into its inactive state and allows dephosphorylation to proceed more effectively. This mechanism is at work in the lower half of the DIN panels. (ii) The high affinity of A for B bound to inactive C puts the breaks on the front-runners, i.e. those C-agents that have already reached the valley floor and switched into the active state, ready for the upward (phosphorylation) leg of the cycle. Those agents need rule `A.Ca` to fire and bind them to A, locking their active state and promoting phosphorylation. But despite these very C-agents having released B, which in turn releases A (high firing rate of `A..B`), A gets immediately captured by B-agents bound to laggard C-agents that have not yet completed their dephosphorylation leg. This holds back the front-runners ready from initiating the upward leg while ensuring that the laggards complete their downward leg. The upper part of the DIN reproduces precisely the statistical synchronization mechanism proposed by van Zon et al. [58], but seen through the lens of a rule-based model. The fact that group IV is already active at $t = 65.5$, a time when the system-wide p-level just passed its minimum, indicates how spread out the distribution of individual p-levels must be, as some are evidently already nearing the top. The basic structure of the DIN then persists throughout the upward phase and beyond, before it breaks down again at $t = 73.2$ and the previous organization reappears.

In sum, the use case illustrates a system in which KaiC proteins go through a phosphorylation cycle. They are incrementally phosphorylated at 6 positions in the molecule when they switch to a dephosphorylation process that removes the phosphates. Yet, even if each protein goes through such a cycle, this would not lead to synchronization across the whole population of proteins. In the model, a series of interactions causes a large fraction of individuals to synchronize and this causes oscillations at the level of the whole population. Using the DIN, the systems biologist can begin to investigate subtleties of the dynamics of these interactions in order to better understand this population-level synchronization.

5 User Feedback

While the detailed case study presented above provides a real-world example of how a biologist can analyze a rule-based model using DINs, we also gathered initial feedback regarding the *DIN-Viz* tool, both through unstructured interviews and written responses to questions about how *DIN-Viz* might be used to support relevant analysis tasks. Feedback was provided by professors and postdoctoral researchers in systems biology or molecular biology from Harvard Medical School and University of Arizona. Each of the six researchers who provided feedback are actively engaged in investigating protein-protein interaction networks related to cancer biology and interested in using bioinformatics tools for rule-based modeling of biological networks.

User response was, for the most part, overwhelmingly positive. For example, one user was initially skeptical since the KaiC use case merely "enables recovery of what we already know." However, after exploring the model in more detail, the user expressed more interest, telling us he was impressed by the ability to reason about macro and micro level behaviors simultaneously: "What I find spectacular is this juxtaposition of the crazy moving hairball on the left, which is the microscopic picture, and the quiet calm coordinated oscillations at the systems level on the right. That is really something emotional to see the two together." One of our systems biology collaborators told us that they were very intrigued by being able to look alternately at the aDIN and pDIN versions of the simulation: "I think I now see the subtle footprint of a critical synchronization mechanism at work in the Kai clock. Seeing is an active act, which is sometimes in the way; the DIN visualization switches "seeing" into a receptive mode where you can get rid of the bias that comes from the active version of seeing." Another user was excited about the potential pedagogical utility of *DIN-Viz*: "The coloring and pinning of the nodes is terrific! A very useful way of interacting with the tool. I could learn a great deal about this system by working with this visualization. I think its going to be very useful in the classroom." However, we were also cautioned that our use case provided only an initial confirmation of the success of our approach: "You need to make a model of a system that's not yet understood and show how the visualization enables a bona fide explanation."

While encouraged by this feedback, in future work we will more rigorously evaluate the use of DINs to help make sense of a wide range of biological processes, both in scientific and educational contexts. In general, there is a lack of evaluation about the effectiveness of rule-based approaches, and *DIN-Viz* could be used to help understand when it is most appropriate to use them to make sense of complex systems.

6 Conclusions

In this paper, we introduced the Dynamic Influence Network, a novel visual analytics technique for investigating the dynamics of how rules influence other rules in rule-based models of complex systems. We demonstrated that our web-based application, *DIN-Viz*, supports a range of visual analysis tasks for making sense of dynamic protein-protein interaction networks.

While our approach has shown to be useful for the manual exploration of these types of rule-based modeling systems, our initial evaluation leads us to believe that some of these manual explorations could be automated. That is, just as a researcher observes that particular rules fire at similar rates, an automated system might highlight a range of interesting temporal patterns, allowing the user to investigate whether or not these patterns are scientifically meaningful.

The goal of the DIN is precisely to focus on rules rather than individual elements or types. However, it could be useful to be able to "drill down" from a selected rule or set of rules to another representation, such as a contact map or biological pathway, that provided additional context for the rule dynamics. Determining how to navigate effectively between these levels of abstraction is a significant visualization challenge that we plan to investigate.

The DIN technique was designed to support the *Kappa* language, and specifically to ingest *KaSim* simulation data, but we expect that it should be relatively straightforward to adapt our technique to support other rule-based modeling languages. More broadly, DINs could be used to represent a wide range of dynamic systems explored in other domains, such as economics [40], ecology [26], or social network analysis [54], among others.

Acknowledgments

This work is funded by the DARPA Big Mechanism Program under ARO contracts W911NF-14-1-0367 and W911NF-14-1-0395.

References

[1] D. Archambault and H. C. Purchase. On the effective visualisation of dynamic attribute cascades. *Information Visualization*, 15(1):51–63, 2016.

[2] B. Bach, E. Pietriga, and J.-D. Fekete. GraphDiaries: Animated transitions and temporal navigation for dynamic networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(5):740–754, 2014.

[3] B. Bach, E. Pietriga, and J.-D. Fekete. Visualizing dynamic networks with Matrix Cubes. In *Proc. CHI*, pp. 877–886. ACM, 2014.

[4] L. Bartram and M. Yao. Animating causal overlays. *Computer Graphics Forum*, 27(3):751–758, 2008.

[5] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. A taxonomy and survey of dynamic graph visualization. *Computer Graphics Forum*, 36(1):133–159, 2017.

[6] P. Bénard, A. Bousseau, and J. Thollot. State-of-the-art report on temporal coherence for stylized animations. In *Computer Graphics Forum*, vol. 30, pp. 2367–2386, 2011.

[7] M. Bostock, V. Ogievetsky, and J. Heer. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[8] P. Boutillier, J. Feret, J. Krivine, and L. K. Quyen. KaSim development homepage. `http://dev.executableknowledge.org`, accessed March 26th, 2017.

[9] L. A. Chylek, L. A. Harris, J. R. Faeder, and W. S. Hlavacek. Modeling for (physical) biologists: An introduction to the rule-based approach. *Physical Biology*, 12(4):045007, 2015.

[10] L. A. Chylek, L. A. Harris, C.-S. Tung, J. R. Faeder, C. F. Lopez, and W. S. Hlavacek. Rule-based modeling: A computational approach for studying biomolecular site dynamics in cell signaling systems. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 6(1):13–36, 2014.

[11] L. A. Chylek, B. Hu, M. L. Blinov, T. Emonet, J. R. Faeder, B. Goldstein, R. N. Gutenkunst, J. M. Haugh, T. Lipniacki, R. G. Posner, J. Yang, and W. S. Hlavacek. Guidelines for visualizing and annotating rule-based models. *Molecular BioSystems*, 7(10):2779–2795, 2011.

[12] P. R. Cohen. DARPA's Big Mechanism program. *Physical Biology*, 12(4):045008, 2015.

[13] T. Dang and A. G. Forbes. BioLinker: Bottom-up exploration of protein interaction networks. In *Proc. PacificVis*. IEEE, 2017.

[14] T. Dang, P. Murray, J. Aurisano, and A. G. Forbes. ReactionFlow: An interactive visualization tool for causality analysis in biological pathways. *BMC Proceedings*, 9(6):S6, 2015.

[15] T. Dang, N. Pendar, and A. G. Forbes. TimeArcs: Visualizing fluctuations in dynamic networks. *Computer Graphics Forum*, 35(3):61–69, 2016.

[16] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling of cellular signalling. In *International Conference on Concurrency Theory*, pp. 17–41. Springer, 2007.

[17] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling and model perturbation. In *Transactions on Computational Systems Biology XI*, pp. 116–137. Springer, 2009.

[18] V. Danos, J. Feret, W. Fontana, and J. Krivine. Scalable simulation of cellular signaling networks. In *Asian Symposium on Programming Languages and Systems*, pp. 139–157. Springer, 2007.

[19] V. Danos and C. Laneve. Formal molecular biology. *Theoretical Computer Science*, 325(1):69–110, 2004.

[20] K. Dinkla, M. El-Kebir, C.-I. Bucur, M. Siderius, M. J. Smit, M. A. Westenberg, and G. W. Klau. eXamine: Exploring annotated modules in networks. *BMC Bioinformatics*, 15(1):201, 2014.

[21] T. Dwyer. Scalable, versatile and simple constrained graph layout. *Computer Graphics Forum*, 28(3):991–998, 2009.

[22] N. Elmqvist and P. Tsigas. Animated visualization of causal relations through growing 2D geometry. *Information Visualization*, 3(3):154–172, 2004.

[23] J. R. Faeder, M. L. Blinov, and W. S. Hlavacek. Rule-based modeling of biochemical systems with BioNetGen. In I. V. Maly, ed., *Systems Biology*. Springer, 2009.

[24] R. M. Felciano, S. Bavari, D. R. Richards, J.-n. Billaud, T. Warren, R. Panchal, and A. Krämer. Predictive systems biology approach to broad-spectrum, host-directed drug target discovery in infectious diseases. In *Proc. Pacific Symposium on Biocomputing*, pp. 17–28, 2013.

[25] J. Feret, V. Danos, J. Krivine, R. Harmer, and W. Fontana. Internal coarse-graining of molecular systems. *Proceedings of the National Academy of Sciences*, 106(16):6453–6458, 2009.

[26] T. Filatova, P. H. Verburg, D. C. Parker, and C. A. Stannard. Spatial agent-based models for socio-ecological systems: Challenges and prospects. *Environmental Modelling & Software*, 45:1–7, 2013.

[27] A. G. Forbes, T. Hollerer, and G. Legrady. Behaviorism: A framework for dynamic data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1164–1171, 2010.

[28] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano. CellDesigner 3.5: A versatile modeling tool for biochemical networks. *Proceedings of the IEEE*, 96(8):1254–1265, 2008.

[29] R. Gostner, B. Baldacci, M. J. Morine, and C. Priami. Graphical modeling tools for systems biology. *ACM Computing Surveys*, 47(2):16–1–21, 2014.

[30] S. Hadlak, H. Schumann, C. H. Cap, and T. Wollenberg. Supporting the visual analysis of dynamic networks by clustering associated temporal attributes. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2267–2276, 2013.

[31] L. A. Harris, J. S. Hogg, J.-J. Tapia, J. A. Sekar, S. Gupta, I. Korsunsky, A. Arora, D. Barua, R. P. Sheehan, and J. R. Faeder. BioNetGen 2.2: Advances in rule-based modeling. *Bioinformatics*, 32(21):3366–3368, 2016.

[32] D. Holten and J. J. van Wijk. A user study on visualizing directed edges in graphs. In *Proc. CHI*, pp. 2299–2308. ACM, 2009.

[33] N. Kadaba, P. Irani, and J. Leboe. Visualizing causal semantics using animations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1254–1261, 2007.

[34] H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, 2002.

[35] H. Kitano. Systems biology: A brief overview. *Science*, 295(5560):1662–1664, 2002.

[36] H. Kitano, A. Funahashi, Y. Matsuoka, and K. Oda. Using process diagrams for the graphical representation of biological networks. *Nature Biotechnology*, 23(8):961–966, 2005.

[37] K. W. Kohn, M. I. Aladjem, S. Kim, J. N. Weinstein, and Y. Pommier. Depicting combinatorial complexity with the molecular interaction map notation. *Molecular Systems Biology*, 2(1):51, 2006.

[38] D. C. Krakauer, J. P. Collins, D. Erwin, J. C. Flack, W. Fontana, M. D. Laubichler, S. J. Prohaska, G. B. West, and P. F. Stadler. The challenges and scope of theoretical biology. *Journal of Theoretical Biology*, 276(1):269–276, 2011.

[39] A. Krämer, J. Green, J. Pollard, and S. Tugendreich. Causal analysis approaches in Ingenuity Pathway Analysis (IPA). *Bioinformatics*, 30(4):523–30, 2014.

[40] C. Kühn and K. Hillmann. Rule-based modeling of labor market dynamics: an introduction. *Journal of Economic Interaction and Coordination*, 11(1):57–76, 2016.

[41] A. Lex, C. Partl, D. Kalkofen, M. Streit, S. Gratzl, A. M. Wassermann, D. Schmalstieg, and H. Pfister. Entourage: Visualizing relationships between biological pathways using contextual subsets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2536–2545, 2013.

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

[43] C. Ma, A. G. Forbes, D. A. Llano, T. Berger-Wolf, and R. V. Kenyon. SwordPlots: Exploring neuron behavior within dynamic communities of brain networks. *Journal of Imaging Science and Technology*, 60(1):10405–1–13, 2016.

[44] C. Ma, R. V. Kenyon, A. G. Forbes, T. Berger-Wolf, B. J. Slater, and D. A. Llano. Visualizing dynamic brain networks using an animated dual-representation. In *Proc. EuroVis*, pp. 73–77. Eurographics, 2015.

[45] D. Mashima, S. Kobourov, and Y. Hu. Visualizing dynamic data with maps. *IEEE Transactions on Visualization and Computer Graphics*, 18(9):1424–1437, 2012.

[46] J. Moody, D. McFarland, and S. Bender-deMoll. Dynamic network visualization. *American Journal of Sociology*, 110(4):1206–1241, 2005.

[47] P. Murray, F. McGee, and A. G. Forbes. A taxonomy of visualization tasks for the analysis of biological pathway data. *BMC Bioinformatics*, 18(2):21–1–13, 2017.

[48] M. Nakajima, K. Imai, H. Ito, T. Nishiwaki, Y. Murayama, H. Iwasaki, T. Oyama, and T. Kondo. Reconstitution of circadian oscillation of cyanobacterial KaiC phosphorylation in vitro. *Science*, 308(5720):414–415, 2005.

[49] F. Paduano, R. Etemadpour, and A. G. Forbes. BranchingSets: Interactively visualizing categories on node-link diagrams. In *Proc. VINCI*, pp. 9–16. ACM, 2016.

[50] F. Paduano and A. G. Forbes. Extended LineSets: A visualization technique for the interactive inspection of biological pathways. *BMC Proceedings*, 9(6):S4, 2015.

[51] C. Partl, A. Lex, M. Streit, D. Kalkofen, K. Kashofer, and D. Schmalstieg. enRoute: Dynamic path extraction from biological pathway maps for exploring heterogeneous experimental datasets. *BMC Bioinformatics*, 14(19):1–16, 2013.

[52] M. Pedersen, N. Oury, C. Gravill, and A. Phillips. Bio Simulators: A web UI for biological simulation. *Bioinformatics*, 30(10):1491–1492, 2014.

[53] M. Pedersen, A. Phillips, and G. D. Plotkin. A high-level language for rule-based modelling. *PLOS ONE*, 10(6):e0114296, 2015.

[54] A. Prskawetz. The role of social interactions in demography: An agent-based modelling approach. In *Agent-Based Modelling in Population Studies*, pp. 53–72. Springer, 2017.

[55] A. Purgato, M. Santambrogio, T. Berger-Wolf, and A. G. Forbes. Interactive visualization for brain spatio-temporal networks. In *Proc. BHI*, pp. 21–24. IEEE, 2017.

[56] A. M. Smith, W. Xu, Y. Sun, J. R. Faeder, and G. E. Marai. RuleBender: Integrated modeling, simulation and visualization for rule-based intracellular biochemistry. *BMC Bioinformatics*, 13(8):S3, 2012.

[57] B. Tversky, J. B. Morrison, and M. Betrancourt. Animation: Can it facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002.

[58] J. S. van Zon, D. K. Lubensky, P. R. Altena, and P. R. ten Wolde. An allosteric model of circadian KaiC phosphorylation. *Proceedings of the National Academy of Sciences*, 104(18):7420–7425, 2007.

[59] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing group structures in graphs: A survey. In *Computer Graphics Forum*, 2016. doi: 10.1111/cgf.12872

[60] C. Vehlow, D. P. Kao, M. R. Bristow, L. E. Hunter, D. Weiskopf, and C. Görg. Visual analysis of biological data-knowledge networks. *BMC Bioinformatics*, 16(1):1–15, 2015.

[61] G. Vigueras and J. A. Botia. Tracking causality by visualization of multi-agent interactions using causality graphs. In *International Workshop on Programming Multi-Agent Systems*, pp. 190–204. Springer, 2007.

[62] J. Villegas, R. Etemadpour, and A. G. Forbes. Evaluating the perception of different matching strategies for time-coherent animations. In *Human Vision and Electronic Imaging XX (HVEI)*, vol. 9394 of *Proc. Electronic Imaging*, pp. 939412–1–13. SPIE-IS&T, 2015.

[63] C. Ware, E. Neufeld, and L. Bartram. Visualizing causal relations. In *Proc. Information Visualization*, pp. 39–42. IEEE, 1999.

[64] J. Wilson-Kanamori, V. Danos, T. Thomson, and R. Honorato-Zimmer. Kappa rule-based modeling in synthetic biology. In M. A. Marchisio, ed., *Computational Methods in Synthetic Biology*, pp. 105–135. Springer, 2015.

[65] K. Xu, C. Rooney, P. Passmore, D.-H. Ham, and P. H. Nguyen. A user study on curved edges in graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2449–2456, 2012.

[66] A. Q. Ye, O. A. Ajilore, G. Conte, J. GadElkarim, G. Thomas-Ramos, L. Zhan, S. Yang, A. Kumar, R. Magin, A. G. Forbes, and A. D. Leow. The intrinsic geometry of the human brain connectome. *Brain Informatics*, 2(4):197–210, 2015.

[67] H. Zhang, M. Sun, D. D. Yao, and C. North. Visualizing traffic causality for analyzing network anomalies. In *Proc. International Workshop on Security and Privacy Analytics*, pp. 37–42. ACM, 2015.