
Subject Section

ACCORDION: Clustering and Selecting Relevant Data for Guided Network Extension and Query Answering

Yasmine Ahmed^{1,*}, Cheryl Telmer² and Natasa Miskov-Zivanov^{1,3,*}

¹Electrical and Computer Engineering Department, ²Bioengineering, Computational and Systems Biology, University of Pittsburgh, ³Department of Biological Sciences, Carnegie Mellon University, Pittsburgh, PA

*To whom correspondence should be addressed.

Associate Editor: XXXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Querying new information from knowledge sources, in general, and published literature, in particular, aims to provide precise and quick answers to questions raised about a system under study. In this paper, we present ACCORDION (Automated Clustering Conditional On Relating Data of Interactions to a Network), a novel tool and a methodology to enable efficient answering of biological questions by automatically assembling new or expanding existing models using published literature. Our approach integrates information extraction and clustering with simulation and formal analysis to allow for automated iterative process that includes assembling, testing and selecting most relevant models, given a set of desired system properties. We demonstrate our methodology on a model of the circuitry that controls T cell differentiation. To evaluate our approach, we compare the model that we obtained, using our automated model extension approach, with the previously published manually extended T cell model. Besides being able to automatically and rapidly reconstruct the manually extended model, ACCORDION can provide multiple viable extended model versions. As such, it replaces large number of tedious or even impractical manual experiments and guides alternative interventions in real biological systems.

Contact: {yaa38, nmzivanov}@pitt.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

While modeling helps explain complex systems, guides data collection and generates new challenges and questions [1], it is still largely dependent on human intervention. For example, in biology, model creation requires reading hundreds of papers, extracting useful information manually, incorporating background and common-sense knowledge of domain experts, and conducting wet-lab experiments. These time-consuming steps make the creation and the development of models a slow, laborious and error-prone process. In addition to that, as the amount of biological data in the public domain is constantly growing, the growth further augments the issues of data inconsistency and fragmentation [2]. Therefore, the automation of model building, and even more, of model extension, when new information becomes available, or when the domain knowledge advances, is a critical next step for computational modeling. Such automation will not only lead to more efficient modeling due to

reducing the amount of slow human interventions, but will also allow for more consistent, comprehensive and robust modeling process.

In the last few decades, computer models have been used to explain how biomolecular signaling pathways regulate cell functions. Usually, modelers start with a few seed components and their interactions, a baseline model, which is most often found in curated public model databases such as Reactome[3], STRING[4], KEGG[5], or in published literature. Depending on the questions to be answered with modeling, the baseline model is usually further extended with the information extracted from published literature or obtained from domain experts [6]. The literature information extraction starts with a formal search query, which is defined according to a question posed about the modeled system. The search query guides automated selection of articles that contain relevant information from published literature databases. As the biomedical literature mining tools are becoming essential for the high throughput extraction of knowledge from scientific papers, we use in our work

existing machine reading engines. We then use the extracted information to extend or assemble models and answer questions about the studied system [7].

In [8], the authors proposed a method that starts with a baseline model and selects interactions automatically extracted from published work. The goal of [8] was to build a model that satisfies pre-defined requirements or to identify new therapeutic targets, formally expressed as existing or desired system properties. As results in [8] demonstrate, automatic model extension is a promising approach for accelerating modeling, and consequently, disease treatment design. The authors in [8] organize the information extracted from literature into layers, based on their proximity to the baseline model. Recently, another extension method that uses Genetic Algorithm (GA) was proposed in [9]. The GA-based approach was able to extract a set of extensions that led to the desired behavior of the final expanded model. The disadvantages of the GA-based approach include non-determinism, as the solution may vary across multiple algorithm executions on the same inputs, as well as issues with scalability.

In this work, we propose ACCORDION (Automated Clustering Conditional On Relating Data of Interactions to a Network), a tool that automatically and efficiently assembles the information extracted from available literature into models, tests the newly assembled models, and selects the most suitable model to address user questions. In contrast to [8], our approach focuses on identifying clusters of strongly connected elements in the newly extracted information, which is necessary for these additions to the model to have a measurable impact. Once the interactions extracted from literature are clustered, we score their performance on a selected set of system properties, using statistical model checking [10] and stochastic simulation methods [11]. The scoring helps determine which clusters to add to the baseline model. Therefore, ACCORDION takes at most a few hours to execute thousands of experiments *in silico*, which would take days, or months, or would be impractical to conduct *in vivo* or *in vitro*.

While ACCORDION is not limited to any particular model, and it can be used for extending many different models, in order to demonstrate its accuracy, efficiency, and utility, we used a computational model of T cell differentiation, published in [12]. Our main goal with this case study is to demonstrate that ACCORDION can automatically expand an existing published model into another published model, using new elements and new interactions automatically extracted from published literature. As the final golden model, we used the T cell model published in [13] and the set of desired system properties discussed in [12][13]. The golden model and the properties are used to evaluate the ACCORDION output. To this end, the contributions of this work include: (i) a new method to extend models by combining clustering and path finding that is more efficient than existing methods; (ii) an evaluation of the effect of published literature and machine reading on automatically reproducing a manually built model; (iii) several new candidate models of the circuitry controlling naïve T cell differentiation, assembled automatically, satisfying the same set of desired properties as existing manually built models, and thus, enabling exploration of redundancies or discovering alternative pathways of regulation.

2 Background

We provide in this section an overview of several tools and background concepts that are used by ACCORDION. We start with the description of the tools that we have used to automatically find and read published papers relevant for user queries (Section 2.1). Next, in order to use the extracted information in models, while retaining all the useful information, a suitable representation format for model components is critical (Section 2.2). Additionally, we also provide here a brief overview of the model

analysis techniques. These existing techniques and tools are used by ACCORDION to evaluate the models that are expanded with the new extracted information, and to select the best model to address user questions [11][14] (Section 2.3).

2.1 Information extraction from literature

We rewrite user questions in the form of logical expressions. These formally written queries (**Figure 1(a)(left)**) are used to search public literature databases (e.g., PubMed [15]) as illustrated in **Figure 1(a)(middle)**. Once the relevant papers are selected, they are sent to machine reading engines for automated extraction of information (**Figure 1(a)(middle)**).

The machine reading approaches and tools are usually categorized into Information Retrieval (IR) tools, with the main goal of finding papers that pertain to a certain topic, and Entity Recognition (ER) tools, used to identify the biological entities within a text [7][16][17]. The state-of-the-art automated reading engines [18][19] are capable of finding hundreds of thousands of events in cellular signaling pathways from thousands of papers, in a few hours. In the context of biomedical literature, entities represent elements of a biochemical reaction, which can be of various types, such as proteins, chemicals, genes or even biological processes. For each extracted element, reading engines provide its name, the database where it is characterized, and the database identifier for the element. Events in the machine reading output represent the interactions between biochemical elements, which can also be of different types, such as post-translational modifications (e.g., binding, phosphorylation, ubiquitination, etc.), transcription, translation, translocation, and increase or decrease of amount or activity. Machine reading also collects the evidence, usually a sentence from which the information was extracted. In this work, we use an open-source reading engine, REACH [19], to quickly obtain information from literature. In **Figure 1(a)(right)**, we show two example sentences. The REACH reading engine extracts events into an interactions-based format shown in the table in **Figure 1(b)**. In the rest of the paper, we will refer to the list of interactions retrieved from literature, in this format as *reading output*.

2.2 Model representation and executable models

The three rows listed in the table in **Figure 1(b)** can be automatically translated into the element-based BioRECIPES format [20], which is then used as input to the executable model generation (see Section 2.3). The BioRECIPES tabular model representation format is illustrated in **Figure 1(c)** with several examples of molecules and interactions in T cells [12]. In the examples, PTEN is positively regulated by Foxp3, and negatively regulated by TCR. Ras has one positive regulator, TCR, and no negative regulators. IL-2 has both positive and negative regulators, Ras and Foxp3, respectively, while TCR is an input to the model without any regulators.

The BioRECIPES representation format includes, for each model element: (i) name, (ii) type (protein, gene or a chemical), (iii) identifier from a database (e.g., UniProt [21]), (iv) variable that represents state, and (v) set of regulators. While the BioRECIPES format is a sufficient representation for all the relevant element and interaction information, all interactions in a model can also be represented as a directed graph $G(V, E)$, with a set of nodes V and a set of edges E . Each node $v \in V$ corresponds to one model element, and each edge $e(v_i, v_j) \in E$ represents an interaction in which element v_i regulates element v_j . The graphical representation of all model interactions is often referred to as an influence map, and it is especially useful for the methods that are used in ACCORDION, as will be discussed in Section 3.2. Next to the table in **Figure 1(c)**, we show a graph with element interactions listed in the table. As can be seen in the graph, we include the information about the polarity of the interaction in the form of arrow type, that is, a pointed arrow represents positive regulation, while a blunt arrow represents negative regulation.

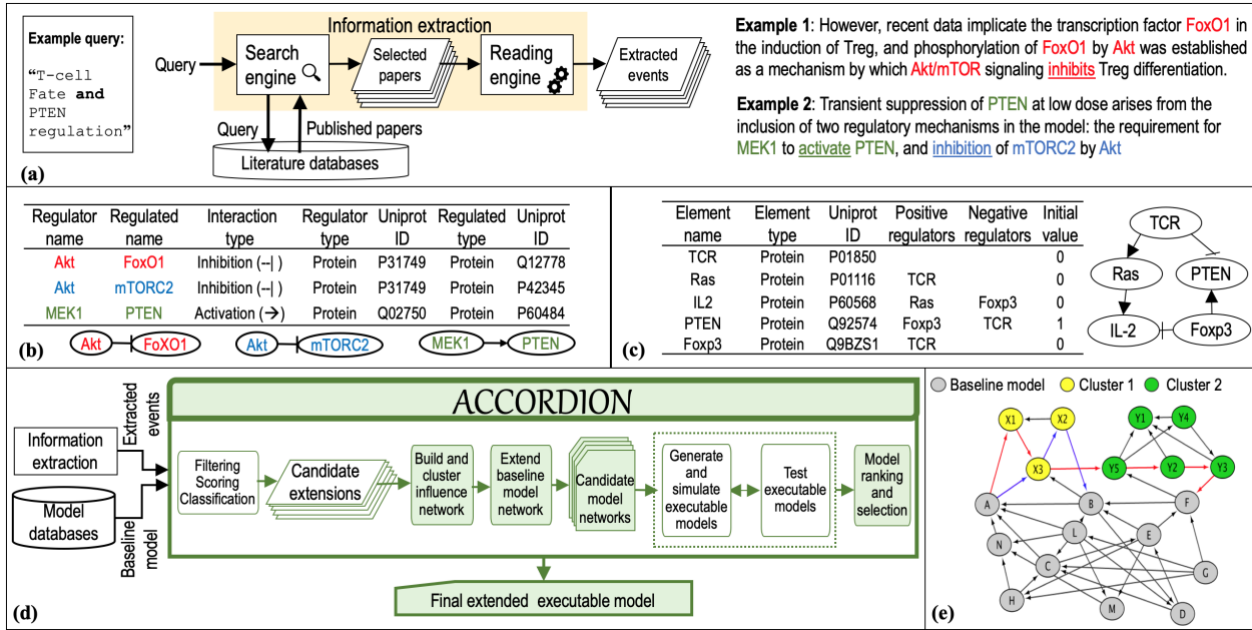


Figure 1. ACCORDION inputs and methodology overview: (a) **Left:** example query used to select relevant papers. **Middle:** main components of information extraction from relevant papers. **Right:** two example sentences with highlighted entities and events that are extracted by machine readers. (b) **Top:** tabular outputs from REACH engine when reading example sentences from (a). **Bottom:** graphical representation of REACH outputs. (c) **Left:** tabular representation of several elements and their influence sets (positive and negative regulators) in BioRECIPES format. **Right:** graphical representation of elements and influence sets. (d) The flow diagram of the ACCORDION processing steps, inputs, and outputs. (e) Toy example of a baseline model and connected clusters: blue edges highlight a return path within one cluster, and red edges show a return path connecting two clusters.

We will refer to the set of regulators of an element as its influence set, distinguishing between positive and negative regulators. Additionally, we can define a vector of all variables representing states of model elements as $x = (x_1, \dots, x_N)$, where N is the total number of model elements. If we use Boolean variables, then $x_i \in \{0, 1\}$, where $i=1..N$. Next, we can assign a state transition function to any model element, which defines a state change of the element, given the states of its regulators. We will refer to these functions as element update rules and to the model with update rules as an executable model. In the case of Boolean variables representing element states, the basic operations are AND (*), OR (+) and NOT (!). For example, one version of update rules for the small graph in **Figure 1(c)** can be: $PTEN = Foxp3 * !TCR$, $Ras = TCR$, and $IL-2 = Ras * !Foxp3$. The choice between AND and OR operation depends on the available information about interactions and element regulations. For example, for an element to be “activated”, all necessary regulators are combined with an AND operation, and all sufficient regulators with an OR operation.

2.1 Model analysis

Here, we describe two methods, simulation and formal analysis, that we will use to evaluate the models that were assembled or extended with the newly obtained information and data.

2.3.1 Stochastic simulation. We use DiSH simulator [11] to observe dynamic behavior of the baseline model and the extended models. DiSH can simulate networks with multi-valued elements in both deterministic and stochastic manner, and we utilize both these features in our analysis, as shown later in Section 5. Each simulation run starts with a specified initial model state, where initial values are assigned to all model elements to represent a particular system state (e.g., naïve cell, regulatory T cell, etc.). Next, we use element update rules to determine element state transitions. We track element changes for a pre-defined number of simulations steps, or until a steady state is reached [11].

A trajectory of values is obtained for each element in a single simulation run, that is, if we assume that a run has M steps, we define the trajectory of element x_i in the k th run as $T_k(x_i) = (x_{i0k}, x_{i1k}, \dots, x_{iMk})$. When the simulator is in the stochastic mode, in each simulation step, only one element is randomly chosen, and its new value is computed according to its update rule. Depending on the information available, the rates at which elements are updated can be different across model elements; when there is limited information about elements in the BioRECIPES model file, we choose to use the same rate for all elements. In either case, due to the randomness in element update order, multiple runs that start with the same initial state may result in different trajectories. DiSH simulations output a file that includes all the simulated trajectories for all model elements, in other words, for K runs, for each model element x_i , we obtain its simulated trajectories $T(x_i) = (T_1(x_i), T_2(x_i), \dots, T_K(x_i))$. Additionally, we compute average trajectories for model elements (by averaging element values in each step across all trajectories) and use these averaged trajectories to plot and visualize element behavior over time.

2.3.2 Statistical model checking. In this work, we use statistical model checking [10][14] to test our models against formally defined properties. Model checking, in general, has been used to verify whether a model of a system, or a system design, satisfies a set of properties describing expected behavior of the system. Each property is encoded into Bounded Linear Temporal Logic (BLTL) [14]. Here, we choose statistical model checking since the state transitions are not necessarily deterministic, and we follow the simulation approach described in Section 2.3.1. The simulation approach that we use is similar to the discrete-time Markov chain approach [22], thus the verification problem can be mapped to computing the probability of whether a given temporal logic formula is satisfied by the system. We can presumably use numerical methods to compute the exact probability; however, this straightforward implementation suffers from the state explosion problem [23]. Statistical model checking, on the other hand, provides a probability estimate using simulation and avoids a full

state space search. The input to the model checker is a system property expressed as a BLTL formula and the output is a probability estimate of the model satisfying that property. Statistical model checking uses randomized sampling to generate simulation trajectories from the system model, and then performs statistical analysis on those trajectories. For instance, let us assume that we would like to test a property that, at any point within the first s_1 time steps, element v_0 becomes 1 and element v_1 becomes 0, and they both keep those values for at least s_2 time steps. We would then write the formula: $F^{s_1} G^{s_2}(v_0 = 1 \wedge v_1 = 0)$, where F^{s_1} stands for “any time in the future s_1 steps”, and G^{s_2} stands for “globally for s_2 steps”.

3 Proposed methodology

The steps and components within ACCORDION are outlined in **Figure 1(d)**. The first step of our proposed methodology is creating an input for ACCORDION, which includes extracting new event information from literature by machine reading engines, followed by filtering, scoring and classifying these events. Once the new input is created, the three main steps within ACCORDION are performed, and they include (1) clustering of new events, (2) assembly of the clustered event data into models, and (3) selection of the most suitable and useful events. In the following subsections, we discuss each of these steps in detail.

3.1 Extraction and classification of new event information

To query for new information from paper databases, we write questions as search terms in the form of logical expressions, which can be used by literature search engines, either internet based tool (e.g., Google), or Medline search tools (e.g., PubMed [15], Ovid [24]). The search engines return a list of papers most relevant to these search terms. The selected papers are then used as an input for machine reading engines, which extract entities and events from the papers. Once the event data is extracted, it is forwarded to the event classification tool, to identify potential extensions for the existing model. However, it is important to note that the output of machine reading engines is often inconsistent and even inaccurate.

Therefore, extracted event information needs to be filtered before it can be used in models. First, we select from the reading output only the protein-protein interactions and remove any biological processes. The rationale behind this is the lack of the context a biological process has been mentioned in, which will affect the interpretation of a given interaction if one of its members contains a biological process. The extracted interactions are further filtered using public protein interaction databases [21][4][25], which increases the confidence in the interactions that will be used as potential extensions for models. To classify the remaining interactions, we use an interaction classification tool. As described in Section 1, we assume that, in order to answer a query, we would most often start from an existing baseline model, and thus, the extracted interactions are classified according to their relationship with the baseline model. We classify interactions into three groups: (a) *corroborations* – when the interaction from the reading output matches an interaction that already exists in the model; (b) *contradictions* – when the interaction from the reading output represents a contradicting regulatory mechanism from the one that exists between the same elements in the model; (c) *extensions* – when the interaction from the reading output is not in the model.

As corroborations confirm what is already in the baseline model, we do not use them in extending the baseline model. In our future work, we plan to include a confidence measure for the interactions in the model, and the corroborations found in literature would contribute to computing the confidence. Additionally, ACCORDION currently does not examine and utilize the information within the extracted contradictions, although they

may hold useful information about the modeled system. In some cases, contradictions could be even considered as model extensions. For example, in the reading output, we often come across interactions stated as “A positively regulates B” or just “A regulates B”, while the model includes interaction “A inhibits B” or “B inhibits A” or “B regulates A”. Given that extracted contradictions can be further explored and the information within contradictions can sometimes lead to model improvements, we will explore them more carefully in our future work. To extend the baseline model, only the interactions that are classified as extensions form an input for ACCORDION, and in the rest of the paper, we will refer to these interactions as *Candidate Extension Interactions (CEIs)*.

3.2 Clustering of new extracted interactions

The method used to identify clusters of extracted, filtered and classified CEIs is formally outlined in Algorithm 1 (see Supplementary material) and described in detail here.

The set of CEIs can be represented as a set of candidate extension edges E_{ext} , and the source and target nodes of these edges that are not already in the baseline model graph, $G_{BM}(V_{BM}, E_{BM})$, will be members of the set of candidate extension nodes V_{ext} . We then create a new graph $G_{new}(V_{new}, E_{new})$, where $V_{new} = V_{BM} \cup V_{ext}$, and $E_{new} = E_{BM} \cup E_{ext}$. **Figure 1(e)** shows a toy example graph G_{new} , where grey nodes belong to the baseline model, while yellow and green nodes belong to the CEIs obtained from machine reading. We further classify the edges $e(vs, vt)$ from the set E_{ext} , where vs is the source node and vt is the target node, into the following categories: (a) edges in which both the source node vs and the target node vt belong to the baseline model: $\{vs, vt\} \in V_{BM}$; (b) edges in which either a source node or a target node belongs to the baseline model: $(vs \in V_{BM} \text{ and } vt \notin V_{BM})$ or $(vs \notin V_{BM} \text{ and } vt \in V_{BM})$; (c) edges in which neither the source node nor the target node belongs to the baseline model: $\{vs, vt\} \notin V_{BM}$.

Adding the CEIs to the baseline model all at once usually does not result in a useful and accurate model. Alternatively, we can add one interaction at a time and test each model version, which is time consuming, or even impractical, given that the number of models increases exponentially with the number of CEIs. Moreover, adding individual interactions does not have an effect on the model when the interaction belongs to category (iii), and most often when it belongs to category (ii). It proves much more useful to add paths of connected interactions, which are at the same time connected to the baseline model in at least two elements. Therefore, our approach for finding the most useful subset of the CEIs includes finding connected interactions, that is, a set of edges in the graph G_{new} that form a return path. Formally, we say that a path of connected edges $\{e_{i1}(vs_1, vt_1), e_{i2}(vs_2=vt_1, vt_2), e_{i3}(vs_3=vt_2, vt_3), \dots, e_{ij}(vs_j=vt_{j-1}, vt_j)\}$ is a return path, if $\{vs_1, vt_j\} \in V_{BM}$. In **Figure 1(e)**, we highlight one such return path in blue. To find these return paths formed by CEIs, we conduct clustering of the whole graph G_{new} that includes both the baseline model and the CEIs. We use Markov Clustering algorithm (MCL) [26] to cluster the CEIs. MCL is an unsupervised graph clustering algorithm, commonly used in bioinformatics. For example, MCL has been applied on protein-protein interaction networks [27][28]. In [29], the authors showed that the MCL algorithm is tolerant to noise, while identifying meaningful clusters. Additionally, in [29], MCL is compared with another clustering algorithm, Affinity Propagation (AP) algorithm, proposed in [30], and it is demonstrated that the MCL algorithm outperforms the AP algorithm. Moreover, the analysis in [27] supported the superiority of MCL over other clustering techniques [31][32][33] in identifying protein complexes from interaction networks.

MCL simulates random walks on an underlying interaction network, by alternating two operations, expansion and inflation. First, self-loops are

added to the input graph representing biological interactions (e.g., network of extensions, or joint network of baseline model and CEIs), and this graph is then translated into a stochastic Markov matrix [34]. This matrix represents the transition probabilities between all pairs of the graph nodes, and the probability of a random walk of length p between any two nodes can be calculated by raising this matrix to the exponent p , a process called expansion. As longer paths are more common between nodes within the same cluster than between nodes across different clusters, the transition probabilities between nodes in the same cluster will typically be higher in these newly obtained expanded matrices. MCL further amplifies this effect by computing entry wise exponents of the expanded matrix, a process called inflation [26], which raises each element of the matrix to the power r , called inflation parameter (IP). Clusters are determined by alternating expansion and inflation, until the graph is partitioned into subsets such that there are no paths between these subsets.

3.3 Assembly of new interaction data into models

After generating clusters, the next step is to add them to the model. Similar to the discussion about individual extensions in Section 3.2, we can add clusters one at a time, or in groups. The more cluster or cluster groups we generate, the more models we need to assemble and test. Moreover, the number of possible cluster combinations grows with the total number of generated clusters, and the number of clusters depends on the inflation parameter r , as it directly influences cluster granularity [26]. To alleviate the problem of the large number of cluster combinations, we propose a method for combining the clusters found by the MCL algorithm. Formally, if the clusters we generated in the previous step are C_1, \dots, C_n , and we find a subset of clusters C_{i_1}, \dots, C_{i_j} , where $j > 1$, for which at least one return path exists that goes through all the clusters, then we merge these clusters into a single cluster that will be added to the model. An example of a multi-cluster path is highlighted in red in **Figure 1(e)**, starting at the Baseline model, connecting to Cluster 1, then connecting to Cluster 2, and from Cluster 2 connecting back to the Baseline model. Therefore, we extend the baseline model with multiple clusters simultaneously, based on how clusters are connected to the model. The cluster merging procedure is outlined in Algorithm 2 (see Supplementary material). Finally, we rank and score the final list of candidate clusters, based on the existence of return path, in order to choose the ones that will be incorporated in extension.

Next, we can select one or more clusters from the set of ranked and scored clusters to generate multiple *Candidate Executable Models (CEMs)*. Each CEM contains elements from both the baseline model and the selected cluster(s). Both procedures, the assembly of interaction lists, and the generation of executable models are fully automated. However, element update rules are not necessarily unique, as previously discussed in [8]. For example, if the original rule is “ $A = B + C$ ”, and the candidate extension states that “ D positively regulates A ”, then the new update rule for A can be either “ $A = (B + C) * D$ ”, or “ $A = B + C + D$ ”. We will investigate the effect of adding a new regulatory element in both cases, when AND ($*$) operation is used, and when OR ($+$) operation is used.

3.4 Selection of final extended model

In order to find which of the CEMs is most suitable for answering user questions, we can test all the CEMs on a set of known or desired system properties. We use both simulation and formal analysis to evaluate the CEMs. In order to simulate a model, all model elements need to be assigned a starting state (i.e., initial value). The initial values for the baseline model elements (nodes in the set V_{BM}) are typically already known, however, the newly added elements (nodes in the set V_{ext}) need to

be assigned initial values as well. Unfortunately, machine reading does not usually provide this information. In this work, we assume that all elements within the same cluster have the same initial value. In Section 6, we will compare models with different initializations of the newly added elements to evaluate the effect of initialization on the behavior of the CEMs.

To obtain dynamic traces of the baseline model and the CEMs, we use the DiSH simulator (Section 2.3.1). We test each candidate model using the statistical model checking approach (Section 2.3.2), that is, for each candidate model, we compute a probability of satisfying a set of system properties written as BLTL formulas. As discussed in Section 2.3.2, the statistical model checker calls the simulator in order to obtain element trajectories for a defined number of steps. Finally, we select the model that has the highest probability of satisfying the selected properties as our final extended executable model. The procedure for selecting this final extended model is summarized in Algorithm 3 (see Supplementary material).

4 Case study: T cell differentiation

Naïve peripheral T cells are stimulated via antigen presentation to T cell receptor (TCR) and with co-stimulation at CD28 receptor. This stimulation results in the activation of several downstream pathways, feedback and feedforward loops between pathway elements, which then lead to the differentiation of naïve T cells into helper (Th) or regulatory (Treg) phenotypes. The distribution between Th and Treg cells within the T cell population depends on antigen dose; for instance, high antigen dose results in prevalence of Th cells, while low antigen dose leads to a mixed population of Th and Treg cells. The key markers that are commonly used to measure the outcomes of the naïve T cell differentiation into Th and Treg cells are IL2 and Foxp3, respectively. In other words, Th cells are characterized by high expression of IL-2 and low expression of Foxp3, and Treg cells are characterized by high expression of Foxp3 and low expression of IL-2. To demonstrate our model extension procedure, we use two existing models of T cell differentiation, from [12] [13].

4.1 Baseline model and golden model

In [12], the authors proposed a model where most of the elements are assumed to have two main levels of activity, and are therefore represented with Boolean variables, and their update rules are logic functions. Additionally, the stimulation through TCR is assumed to have three different levels, no stimulation (0), low dose (1), and high dose (2), and it is implemented using two Boolean variables. We used the model from [12] to create the *baseline model* for our case study. The interaction map of this model is provided in [12] (also included in Supplementary material).

In [13], the authors have proposed an extension of the original T cell model from [12], a new model that improved the behavior of the original model. Specifically, in the new model in [13], Foxp3 is present in almost 70% of the differentiated population after the stimulation with low antigen dose, while there is a brief transient induction of Foxp3 after the stimulation with high antigen dose. These results recapitulate experimental observations closer than the results in [12]. We will refer to this model from [13] as the *golden model*. For the baseline, we used the original model from [12], without several interactions overlapping with the golden model from [13] (TCR activates PIP3, PIP3 activates Akt, Akt activates mTORC2 and mTORC2 inhibits Akt). While the model from [12] satisfied a large number of system properties, except for a few that are satisfied by the model in [13] only, the baseline model in its reduced shape does not satisfy a larger set of system properties. Our aim is to use ACCORDION to automatically expand this baseline model in order to

recapitulate the behavior of the golden model, that is, fulfill all the system properties.

4.2 Derivation of properties

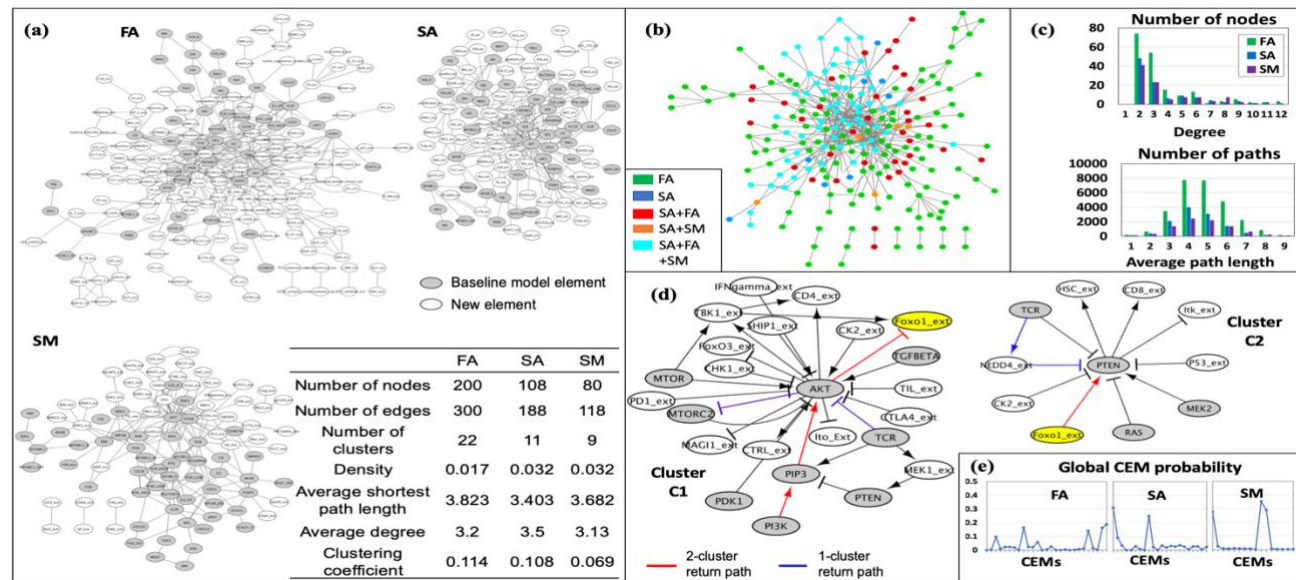


Figure 2. T cell model extension results. (a) Networks obtained when combining baseline model with the CEI set for each of the three cases, FA, SA, and SM. Gray nodes are the baseline model nodes and white nodes are the new nodes that are part of the CEIs. Table: Common graph features measured for the CEI sets in the FA, SA, and SM cases. (b) The network graph for all three sets of CEIs (FA, SA, and SM), highlighting the common nodes. (c) Top: different degree D values and corresponding number of nodes; Bottom: different average path length APL values and the corresponding number of paths. (d) Two clusters that form a return path with the baseline T cell model, shown as directed graphs (yellow node is a common node for both clusters). (e) Overall probability of satisfying desired system properties of each candidate extension model (CEM) that ACCORDION assembled for each of the three reading output sets, FA, SA, and SM (27, 22, and 16 CEMs, respectively).

From the golden model in [13] and the results of its studies, we define a set of properties that our final automatically extended model needs to satisfy. The properties describe the observed responses of the system components to three scenarios: (1) no stimulation ($TCR=0$), (2) stimulation with low antigen dose ($TCR=1$), and (3) stimulations with high antigen dose ($TCR=2$). The properties measure the behavior of the key system components, *Foxp3*, *IL-2*, *PTEN*, *CD25*, *STAT5*, *AKT*, *mTOR*, *mTORC2* and *FoxO1*. The complete list of 27 properties is shown in Figure S4 in the Supplementary material.

5 Results

We conducted two sets of experiments. First, we evaluated our approach for automating model extension on the T cell model example, with data and information from published literature databases. Second, we compared our method of clustering and selecting the information and data extracted from literature with the method that was previously proposed and described in [8].

5.1 Experimental setup

All models that we use or create are written in the BioRECIPES representation format, which was briefly discussed in Section 2.2 and outlined in [20]. From this format, the executable models are generated automatically with the DiSH simulator, which is described in detail in [11] and publicly available at [35].

In the experiments discussed here, we used the PubMed database [15]. The PubMed search was conducted using Entrez [36], an integrated database retrieval system that allows access to a diverse set of databases at the National Center for Biotechnology Information (NCBI) [37]

website. The published articles that were obtained through search of PubMed are read using the REACH engine [19], which extracted a list of events and the corresponding information (see Section 2.1). The REACH reading engine is available online and can be run through the Integrated

Network and Dynamical Reasoning Assembler (INDRA) [38]. We conducted our analysis on three different sets of event data, which were obtained using varying levels of automation and manual intervention. For each set, the list of events, with associated entities, is automatically translated from the reading output into BioRECIPES tabular format.

In the *fully automated* (FA) approach, both the PubMed database search for relevant articles and the extraction of structured event data from the selected articles were automated. Specifically, in the FA experiment, we used search query “T-cell and (PTEN or AKT or FOXO)” and selected top 11 from the best matched papers, by the PubMed search engine. In the *semi-automated* (SA) approach, we selected papers that are cited by [13] and used the event information that REACH extracted from those papers. Finally, in the *semi-manual* approach (SM), we rely the most on human intervention, we manually excluded from the SA reading output those interactions that violate any assumptions made by the authors originally in [12]. For instance, the authors in [12] consider element TCR to be an input to the network, and therefore, TCR should not have any regulators in the T cell model. Therefore, if REACH retrieves an interaction in which TCR is a regulated element, we manually remove these interactions and keep only the interactions having TCR as a regulator.

Model extension algorithms are written in Python. The statistical model checker is written in C++ and it was used to test all candidate models on a set of properties listed in Figure S4 in the Supplementary material. The properties are written as BLTL formulas. The overall iterative model extension tool is written in Python, and it was run on a 3.3 GHz Intel Core i5 processor. For the clustering algorithm, we used the MCL package from [26], and for the visualization of the graphs we used Cytoscape [35].

5.2 Effect of selected literature and machine reading output

The events we obtained in all three cases (FA, SA, and SM) include new elements that are not in the baseline model, as well as the baseline model elements. In **Figure 2(a)**, we show the network of events (undirected interaction map, for easier visualization) extracted from literature using the FA, SM, and SA approaches. The white nodes are the new elements obtained from automated reading (denoted with suffix “_ext”) and the grey nodes are the baseline model elements. Following our Algorithm 1, we form a joint network that includes both the interactions from the baseline model and all the interactions extracted from literature, and then cluster these networks. We obtained 22 clusters using the FA set, 11 clusters using the SA set, and 9 clusters using the SM set.

We were interested in further exploring the structure and emerging properties of the three sets (FA, SA, and SM) of extracted events. This will provide a prior knowledge about the main characteristics of the network constructed from the reading output in order to facilitate mining such networks. Moreover, when applying our proposed method on any new case study, modelers will have a clear overview about the network that should be generated. Therefore, for each network, we computed three main graph parameters as follows.

Average path length (*APL*) [40] is defined as the average number of steps along the shortest paths for all possible pairs of nodes:

$$APL = \frac{1}{n \cdot (n-1)} \cdot \sum_{i \neq j} d(v_i, v_j)$$

where n is the number of nodes in the graph. The clustering coefficient (*Coeff*) [40] is computed for each node in a directed graph as:

$$Coeff = \frac{T(u)}{\frac{degree^{tot}(u)(degree^{tot}(u)-1) - 2degree^{out}(u)}{2}}$$

where $T(u)$ is the number of triangles in the graph that contain node u , $degree^{tot}$ is the sum of the in degree (the number of incoming edges) and the out degree (the number of outgoing edges) of u , and $degree^{out}(u)$ is the reciprocal of $degree^{tot}$ of u . *Coeff* is a number between 0 and 1, and when *Coeff* approaches 0, the graph is more likely to contain stars, while the *Coeff* approaching 1 means the graph is a clique. The graph density D [40] is defined for a directed graph as:

$$D = \frac{E}{|V|(|V|-1)}$$

where E is the number of edges and V is the number of nodes. A graph is considered to be dense if the number of edges is close to the maximum number of possible edges, therefore, the graph density is close to 1 for a dense graph and close to 0 for a sparse graph.

We list in the table in **Figure 2(a)** the main graph parameters for the FA, SA, and SM networks. As can be seen from the table, the FA network has the largest number of nodes and edges, and it results in the largest number of clusters. On the other hand, SA and SM have smaller number of edges and nodes. Moreover, SM is considered a subset of SA, after the removal of many edges and nodes of specific types (Section 5.1). In **Figure 2(b)**, we highlight the difference between the three networks: FA is highlighted in green, SA in blue, and SM, which is a subset of SA, in orange. In addition, we show the overlapping nodes between the three networks in cyan.

Interestingly, it was observed that despite network diversity, FA, SA and SM share prominent structural features: they have small *APL*, small *Coeff*, and small D , and thus, large degrees are unlikely. This similarity is even better illustrated in **Figure 2(c)**, showing the degree histogram for the nodes in each network that follows a power law, and the distribution of the average path length centered around a value of approximately 4. As can be noticed, both network parameters have similar patterns but with different count numbers for each reading output set in proportion to the size of its network. Moreover, the density graph suggests that the networks constructed from the information extracted by machine readers are less

dense, and the average path length is small, even with varying network size. These results also suggest that the difference in size of reading output sets did not affect the characteristics of networks constructed from CEIs and the baseline model of our case study. Specifically, the inspection of obtained clusters shows that they are less dense and star-like networks (two examples shown in **Figure 2(d)**). Moreover, computing these network parameters will predict whether our method will work properly or not. For instance, if the *APL* is large, we will expect to extract a fewer number of return paths from the constructed network, and therefore, in our analysis we will lack the connectivity of the CEIs to the baseline model. Additionally, the less dense graphs will reduce the computation time, and computing this parameter helps determine in advance the expected execution time of our algorithm.

5.3 Return path and best candidate model

To extend the baseline model, we first test the connectivity of each cluster to the model by searching for a return path (starts and ends in the baseline model) between an individual cluster and the model. In **Figure 2(d)**, we highlight in blue a return path that exists between cluster C1 and the baseline model (TCR → AKT → MTORC2), and a return path that exists between cluster C2 and the baseline model (TCR → NEDD4_ext → PTEN), where clusters C1 and C2 are two of the nine clusters generated from the SM set. Furthermore, we explored multiple cluster connectivity with respect to return paths and created 5, 11, and 7 additional CEMs by merging two clusters together from the clusters obtained using FA, SA, and SM sets, respectively. Therefore, the total number of CEMs resulting from the FM set is 27, from the SA set is 22, and from the SM set 16. We also highlight in red in **Figure 2(d)** a return path that exists between the baseline model, and clusters C1 and C2 (PI3K → PIP3 → AKT → Foxo1_ext → PTEN).

The final list of CEMs includes the baseline model extended by one or two of the clusters generated by MCL for each network. For several candidate models, and for all the 27 properties, we show in **Figure 2(e)** the global probability estimates for all candidate models, in the FA, SA, and SM cases. The highest peak per graph represents the best candidate model when compared to the golden model. Assuming independence, we computed the global probability estimate for each candidate model by multiplying all the probability estimates for all properties. From the morphology of the generated network, we expect a cluster to affect the behavior of the model if it contains the key elements included in system properties. As a consequence, the probability estimates for satisfying properties will vary for those clusters. However, clusters lacking those key elements will most probably not affect the behavior of the model, and thus, larger number of properties will not be satisfied. In our case study, we found that the CEMs that include two clusters with key elements satisfy a larger number of properties. Merging clusters helped increase the probabilities, however, the 70% steady-state level of Foxp3 in the low-dose scenario observed in [13] is not achieved. The best performance is obtained for the model that combines two clusters C1 and C2, (**Figure 2(d)**) obtained from the SM set, which satisfies almost all properties (24 out of 27), (**Figure 2(e)**). Additionally, these two clusters together restored all the missing interactions removed from the golden model (Section 4.1). The network of the best candidate model for the SM reading output is shown in the supplementary material (Figure S5).

5.4 Comparison with existing model

We tested the effectiveness of the previously proposed model extension method from [8] when applied to our case study. This is achieved by replacing our model extension method by the method introduced in [8],

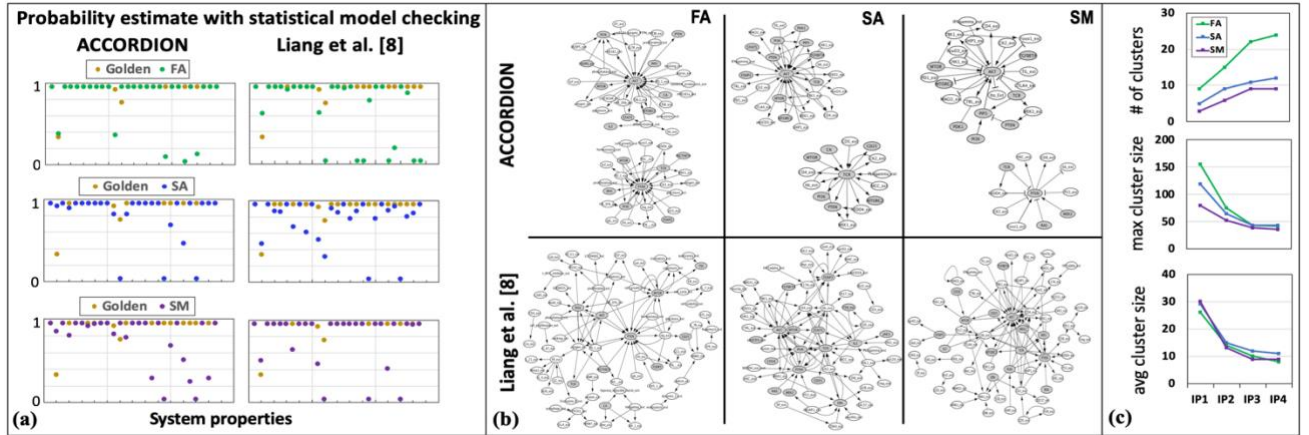


Figure 3. (a) Probability estimate for each tested property (27 properties overall), for the golden model, and for the best models obtained from the three reading outputs (FA, SA, SM) for ACCORDION and the method from [8]. (b) CEIs that were included in the final model for each FA, SA, and SM output, for ACCORDION and [8]. (c) Several cluster characteristics measured as functions of inflation parameter (IP), for FA, SA, and SM reading outputs (IP1=0.5, IP2=2, IP3=4, IP4=6).

using the same baseline T cell model (described in Section 4.1) and the three reading output sets (FA, SA, and SM). In [8], the authors described an automated extension method that considers only the extensions that are related to the baseline model. They first identify a set of baseline model elements of interest, and then, they add the extensions based on their proximity to the elements of interest in single or multiple steps. Next, they introduce several extension configurations depending on the approach that the user is interested in. For example, the focus of model extension can be including the regulation of a certain element or a set of elements, regardless of the number of extension layers this would require. Another approach discussed in [8] focuses on reducing the number of layers while tracking the effect of adding new extensions to the baseline model. In this work, we focus on studying the effect of adding new extensions to the baseline model, therefore, we used the latter approach described above, which is less time consuming, and we applied it on our case study.

Figure 3(a) highlights the difference between the results of our method and the method from [8], when tested using statistical model checking. We compared the probability estimate for each property and each candidate model using our method and the method from [8]. As can be observed, our method outperforms the method from [8] in the case of the FA and SA reading outputs. However, in the SM case, the method from [8] shows slightly better results. These results indicate that the layer-based approach is less effective when used on a large set of CEIs and without any human intervention. The visualization of the topology of the sets of extensions extracted by each method is shown in Figure 3(b). Our method provides concise groups of connected CEIs, that are at the same time connected to the baseline model through return paths. On the other hand, the networks generated by the method from [8], show disconnected components having several interactions not connected to the model (Figure 3(b)). Thus, the comparisons we conducted suggest that the Liang et al. method [8] has two major drawbacks: it is subjective and prone to human judgment variation in selecting the number of elements of interest and the number of layers, and it becomes impractical with the large number of layers.

5.5 Guided extension of executable models

A model created automatically with ACCORDION, using the information from the papers available in public databases, which satisfies most of the desired properties, may not necessarily be the same as the golden model. The differences can be found in both network structure and element update functions. Using our extension methodology, we sometimes obtain multiple models that satisfy the same number of properties. This variety

helps us examine redundancies or discover alternative pathways regulating the same target element.

Adding new elements to the baseline model is a threefold challenge. First, when using MCL to cluster our directed networks, the principal handle for changing cluster granularity is the inflation parameter described in Section 3.2. An increase in the inflation parameter causes an increase in the cluster granularity. Therefore, there should be a reasonable way to choose this parameter in order to obtain a meaningful set of clusters for each reading output network. In [26], the authors determined a good set to choose from (e.g., 1.1 to 10.0), however, the range of suitable values will certainly depend on the input graph. For our case study and the different reading output sets, we found that 1.1 will be too low, and 6.0 or above is too high. We have therefore chosen 4 as the inflation parameter in our studies and conducted experiments based on this value (Figure 3(c)).

Moreover, several issues may result from the fact that we use directed networks, while MCL works better with undirected networks. If there is an edge from node v_i to node v_j , with similarity $s(i,j)$, for a directed graph, this implies $s(j,i) = 0$. Therefore, to solve this issue, we make $s(j,i)$ equal to $s(i,j)$, which seems to ignore an important information (directionality). However, the information that has been ignored is not informative with respect to the existence of cluster structure [26]. Finally, the time required by the extension is proportional to the number of properties that we need to test against. In other words, if we have M clusters and P properties, the time required for the extension algorithm is at the order of $O(M*P)$. However, the time complexity can be reduced to $O(M)$ if testing for all the clusters is carried out in parallel.

The second challenge is in deciding which operation to use (e.g., AND or OR), when adding new regulators through CEIs. We investigated element update rules, and the difference between our final model and the golden model. The update rules of some elements will be removed, and the goal is to restore the correct rules from the clusters. For instance, the PTEN update rule from the reduced model in [13] is $PTEN = (FOXO1 * MEK1)$ and $(!CK2 + !NEDD4)$. We can remove the interaction $(!CK2 + !NEDD4)$, and the rule becomes $PTEN = (FOXO1 * MEK1)$, which means the removed interaction should be found in the set of CEIs. In the process of adding new element regulators, if the information about the regulation type or the importance of the regulator (e.g., necessary vs. sufficient) is available, it will guide the choice of the operation.

Finally, the third challenge is in deciding elements' initial values (e.g., 0 or 1 in the Boolean models) when simulating CEMs and testing their behavior against the desired system properties. We found that assigning

different initial values to source and target elements of CEIs that were not in the original baseline model have quite similar results. This emphasizes the robustness of the baseline model and that the final extended model is influenced by the values of elements in the baseline model. These findings are mostly in agreement with what has been shown in [8]. It is likely that these results are influenced by our choice of the case study, and the fact that we defined system properties (Figure S4 in Supplementary material) in terms of the steady states of the key model elements. As our next steps, we will further explore effects of initialization, as well as expand system properties to incorporate more complicated temporal relationships between elements.,

6 Conclusion and future work

In this paper, we have described a novel methodology and a tool, ACCORDION, that can be used to automatically assemble the data extracted from literature into models. Our proposed approach combines machine reading with clustering, simulation, and model checking, to create an automated framework for rapid model assembly and testing. Furthermore, by automatically extending models with the information published in literature, our methodology allows for efficient collection of the existing information in a consistent and comprehensive way, while also facilitating information reuse and data reproducibility, and replacing hundreds or thousands of manual experiments, thereby reducing the time needed for the advancement of knowledge. As our future work, we will apply ACCORDION on large scale case studies, including more complex systems and larger reading output sets.

Acknowledgements

Funding

References

- [1] J. Epstein, "Why Model?," *Cybern. Syst.*, vol. 35, no. 2–3, pp. 117–128, 2008.
- [2] M. A. Valenzuela-Escárcega, G. Hahn-Powell, T. Hicks, and M. Surdeanu, "A Domain-independent Rule-based Framework for Event Extraction," *Proc. ACL-IJCNLP 2015 Syst. Demonstr.*, pp. 127–132, 2015.
- [3] A. Fabregat *et al.*, "The Reactome Pathway Knowledgebase," vol. 46, no. November 2017, pp. 649–655, 2018.
- [4] C. Von Mering *et al.*, "STRING: known and predicted protein – protein associations, integrated and transferred across organisms," vol. 33, pp. 433–437, 2005.
- [5] K. Encyclopedia, "Using the KEGG Database Resource," pp. 1–54, 2005.
- [6] N. Miskov-Zivanov, "Automation of Biological Model Learning, Design and Analysis," *Proc. 25th Ed. Gt. Lakes Symp. VLSI - GLSVLSI '15*, pp. 327–329, 2015.
- [7] O. Etzioni, M. Banko, and M. J. Cafarella, "Machine Reading," *Ameri*, pp. 1517–1519, 2006.
- [8] K. Liang, Q. Wang, C. Telmer, and D. Ravichandran, "Methods to Expand Cell Signaling Models using Automated Reading and Model Checking," pp. 1–15.
- [9] K. Sayed, K. N. Bocan, and N. Miskov-Zivanov, "Automated Extension of Cell Signaling Models with Genetic Algorithm," *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. EMBS*, pp. 289–305, 2018.
- [10] Q. Wang, N. Miskov-Zivanov, B. Liu, J. R. Faeder, M. Lotze, and E. M. Clarke, "Formal modeling and analysis of pancreatic cancer microenvironment," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9859 LNCS, pp. 289–305, 2016.
- [11] K. Sayed, Y. H. Kuo, A. Kulkarni, and N. Miskov-Zivanov, "DiSH simulator: Capturing dynamics of cellular signaling with heterogeneous knowledge," *Proc. - Winter Simul. Conf.*, pp. 896–907, 2018.
- [12] N. Miskov-Zivanov, M. S. Turner, L. P. Kane, P. A. Morel, and J. R. Faeder, "The duration of T cell stimulation is a critical determinant of cell fate and plasticity," *Sci. Signal.*, vol. 6, no. 300, pp. 1–16, 2013.
- [13] W. F. Hawse *et al.*, "Cutting Edge: Differential Regulation of PTEN by TCR, Akt, and FoxO1 Controls CD4 + T Cell Fate Decisions," *J. Immunol.*, vol. 194, no. 10, pp. 4615–4619, 2015.
- [14] S. Kumar-Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A Bayesian Approach to Model Checking Biological Systems," *Cmsb*, no. 2005, pp. 218–234, 2009.
- [15] R. J. Roberts, "PubMed Central : The GenBank of the published literature," vol. 98, no. 2, pp. 381–382, 2001.
- [16] S. Ananiadou, S. Pyysalo, J. Tsujii, and D. B. Kell, "Event extraction for systems biology by text mining the literature," *Trends Biotechnol.*, vol. 28, no. 7, pp. 381–390, 2010.
- [17] R. B. Altman *et al.*, "Text mining for biology - The way forward: Opinions from leading scientists," *Genome Biol.*, vol. 9, no. SUPPL. 2, 2008.
- [18] G. A. P. C. Burns, P. Dasigi, A. de Waard, and E. H. Hovy, "Automated detection of discourse segment and experimental types from the text of cancer pathway results sections," *Database (Oxford)*, vol. 2016, pp. 1–12, 2016.
- [19] M. A. Valenzuela-escárcega *et al.*, "Large-scale Automated Reading of Scientific Cancer Literature Discovers New Cancer Driving Mechanisms," pp. 3–5, 2017.
- [20] K. Sayed, C. A. Telmer, A. A. Butchy, and N. Miskov-Zivanov, "Recipes for translating big data machine reading to executable cellular signaling models," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10710 LNCS, pp. 1–15, 2018.
- [21] A. Bateman *et al.*, "UniProt: The universal protein knowledgebase," *Nucleic Acids Res.*, vol. 45, no. D1, pp. D158–D169, 2017.
- [22] M. Y. Vardi, "Automatic Verification of Probabilistic Concurrent Finite-State Systems," *26th Annu. Symp. Found. Comput. Sci. (FOCS '85)*, pp. 327–338, 1985.
- [23] R. G. Simmons, "Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling."
- [24] "https://www.ovid.com." [Online]. Available: <https://www.ovid.com>.
- [25] T. Gene and O. Consortium, "Gene Ontology : tool for the," vol. 25, no. may, pp. 25–29, 2000.
- [26] S. Van Dongen, "Graph clustering by flow simulation," *Comput. Sci. Rev.*, vol. 1, no. september 1969, pp. 27–64, 2007.
- [27] S. Brohée and J. Van Helden, "Evaluation of clustering algorithms for protein-protein interaction networks," *BMC Bioinformatics*, vol. 19, 2006.
- [28] X. Lei, F. Wang, F. X. Wu, A. Zhang, and W. Pedrycz, "Protein complex identification through Markov clustering with firefly algorithm on dynamic protein-protein interaction networks," *Inf. Sci. (Ny)*, vol. 329, pp. 303–316, 2016.
- [29] J. Vlasblom and S. J. Wodak, "Markov clustering versus affinity propagation for the partitioning of protein interaction graphs," *BMC Bioinformatics*, vol. 10, pp. 1–14, 2009.
- [30] F. B. J and D. D., "Clustering by passing messages between data points," *Science (80-.)*, vol. 315, no. February, pp. 972–976, 2007.
- [31] A. D. King, N. Pržulj, and I. Jurisica, "Protein complex prediction via cost-based clustering," vol. 20, no. 17, pp. 3013–3020, 2004.
- [32] M. Blatt, S. Wiseman, and E. Domany, "Superparamagnetic Clustering of Data," 1996.
- [33] G. D. Bader and C. W. V Hogue, "An automated method for finding molecular complexes in large protein interaction networks," *BMC Bioinformatics*, vol. 27, pp. 1–27, 2003.
- [34] P. A. Gagniac, *Markov Chains: From Theory to Implementation and Experimentation*.
- [35] "https://bitbucket.org/biodesignlab/ ramework/Simulation/Simulator_Python."
- [36] K. J. Schuler GD, Epstein JA, Ohkawa H, "Entrez," *Methods Enzym.*, vol. 266, pp. 141–162, 1996.
- [37] L. Y. Geer *et al.*, "The NCBI BioSystems database," vol. 38, no. October 2009, pp. 492–496, 2010.
- [38] B. M. Gyori, J. A. Bachman, P. K. Sorger, K. Subramanian, and J. L. Muhlich, "From word models to executable models of signaling networks using automated assembly," pp. 1–26, 2017.
- [39] P. Shannon *et al.*, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome Res.*, no. 13, pp. 2498–2504, 2003.
- [40] M. E. J. Newman, "The Structure and Function of Complex Networks *," vol. 45, no. 2, pp. 167–256, 2003.