Dynamic System Explanation

DySE, a framework that evolves to reason about complex systems – lessons learned

Cheryl A. Telmer
Molecular Biosensor and Imaging Center
Carnegie Mellon University
Pittsburgh, PA, USA
ctelmer@cmu.edu

Khaled Sayed
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA, USA
k.sayed@pitt.edu

Adam Butchy
Department of Bioengineering
University of Pittsburgh
Pittsburgh, PA, USA
aab133@pitt.edu

Kara N. Bocan
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA, USA
knb12@pitt.edu

Emilee Holtzapple
Computational and Systems Biology
University of Pittsburgh
Pittsburgh, PA, USA
ERH87@pitt.edu

Casey E. Hansen
Department of Bioengineering
University of Pittsburgh
Pittsburgh, PA, USA
CEH92@pitt.edu

Gaoxiang Zhou
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA, USA
GAZ11@pitt.edu

Yasmine Ahmed
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA, USA
YAA38@pitt.edu

Natasa Miskov-Zivanov [†]
Electrical and Computer Engineering, Bioengineering, Computational and Systems Biology
University of Pittsburgh
Pittsburgh, PA, USA
nmzivanov@pitt.edu

ABSTRACT

The large amount of knowledge contained in the scientific literature can be mined using natural language processing and utilized to automatically construct models of complex networks in order to obtain a greater understanding of complex systems. In this paper, we describe the Dynamic System Explanation (DySE) framework, which configures hybrid models and executes simulations over time, relying on a granular computing approach and a range of different element update functions. A standardized tabular format assembles the collected knowledge into networks for parameterization. The Discrete Stochastic Heterogeneous (DiSH) simulator outputs trajectories of state changes for all model elements, thus providing a means for running thousands of *in silico* scenarios in seconds. Trajectories are also analyzed using statistical model checking to verify against known or desired system properties, determined from text or numerical data. DySE can automatically extend models when additional knowledge is available, and model extension is integrated with model checking to test the validity of additional interactions and dynamics, and enable iterative model updating and improvements. Here, we focus in particular on the difficulties and complications that arise when attempting to automate and integrate information extraction with model assembly and analysis. We discuss how these obstacles are addressed in DySE, the lessons learned from developing and using DySE, and plans for future developments.

CCS CONCEPTS

• Modeling and simulation • Model development and analysis • Computing methodologies

KEYWORDS

Granular computing, Hybrid element-based models, Automated model extension, Statistical model checking

ACM Reference format:
Cheryl A. Telmer, Khaled Sayed, Adam Butchy, Kara N. Bocan, Emilee Holtzapple, Casey E. Hansen, Gaoxiang Zhou, Yasmine Ahmed and Natasa Miskov-Zivanov. 2019. Dynamic System Explanation, DySE, a framework that evolves to reason about complex systems – lessons learned. In *Proceedings of Artificial Intelligence and Data Reuse conference (AIDR 2019). Pittsburgh, PA, USA, 10 pages.* https://doi.org/10.1145/3359115.3359123

1 Introduction

"Survival machines that can simulate the future are one jump ahead of survival machines who can only learn on the basis of overt trial

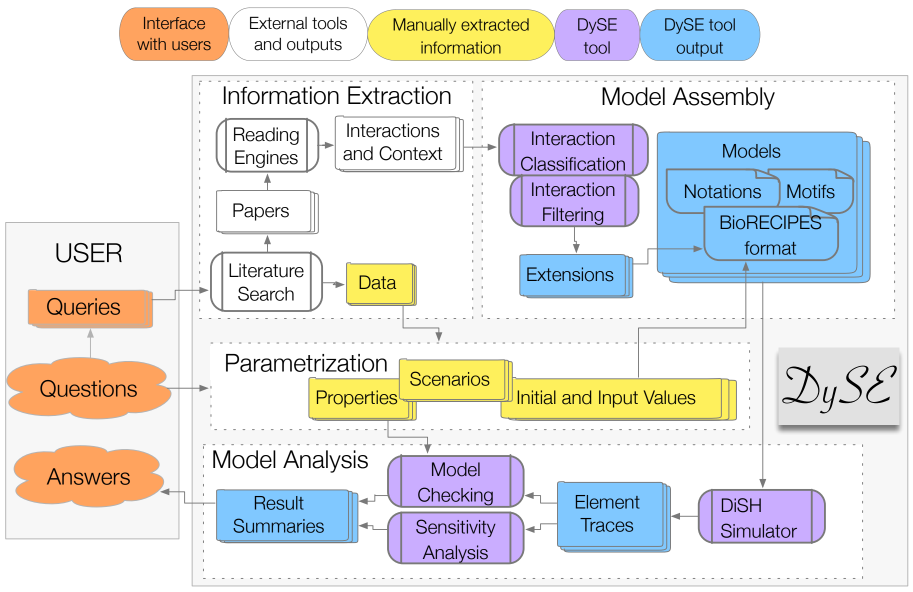


Figure 1. The DySE framework overview showing major tasks initiated by questions from a user.

and error. The trouble with overt trial is that it takes time and energy. The trouble with overt error is that it is often fatal. Simulation is both safer and faster. The evolution of the capacity to simulate seems to have culminated in subjective consciousness." [1]. Simulation is a very useful method that can help us to understand complex systems. Computational modeling has been employed extensively for simulations of engineered systems [2] and has the potential to impact our understanding of complex natural systems including cell signaling, infectious disease biology, epidemiology, and agriculture. Decades of research articles contain content for models, but due to the vast number of publications and databases, comprehensive collection and assembly of this knowledge requires automation. Additionally, even with large volumes of information, all of the components and their relationship rules are often not available for natural systems. Therefore, a modeling approach that can accommodate missing information and abstract relationships is needed in order to study large systems. The Dynamic System Explanation (DySE) framework takes a granular computing approach with hybrid element-based models, where elements change states over time as determined by their regulation functions [3]. This modeling approach has been used to study the duration of T-cell stimulation as it relates to cell fate [4].

The elements and relationships between elements can be provided to the DySE framework from multiple sources: natural language processing (NLP) engines, such as REACH [5], RUBICON [6] and DRUM [7] that extract directed causal interactions from text; inputs from databases; previous models; and directly from user knowledge. In order to use information from these diverse sources, it is necessary to standardize the knowledge representation.

A previously developed tabular format [8], simple for both humans and machines to read and interact with, is compatible with the modeling approach used in DySE.

Curation of the collected information is necessary to remove automated reading errors prior to assembly of the network, and decisions of abstraction level are implemented to decrease the number of elements so that larger systems can be investigated. To capture the timing of events, models incorporate feedback and feedforward loops and common motifs.

Parameterization in DySE is done by assigning a vector of discrete values to each model input, as well as initial values to all model elements. DySE has also incorporated a rich notation for relationships between elements. We study the dynamics of the system through model simulations run with the Discrete Stochastic Heterogeneous, DiSH, simulator [9], and in combination with statistical model checking [10, 11]. DySE automatically updates and extends models using several clustering algorithms and model selection methods [12, 13].

This paper describes the DySE framework as a set of interdependent tools that currently exist, and suggests future development directions. The functionality of DySE is demonstrated with examples related to pancreatic cancer. In Figure 1, we outline the major components and tasks within DySE. In the following sections, we focus in particular on lessons learned from the development and preliminary use of DySE, a framework that integrates information extraction, model assembly, and model analysis. These lessons include:

1. Automated reading output has errors, and therefore, alternative methods to filter and score, including human curation, are necessary.
2. The tabular format is both suitable and easy to process for machines and ideal for including human in-the-loop.
3. A hybrid element rule-based modeling approach is compatible with machine reading output and the tabular format, and executable models can be rapidly generated from this format.
4. Assembling models from machine-extracted interactions is straightforward but having the model achieve expected behavior is more difficult.
5. Automated iterative model assembly and testing is more rapid and comprehensive than manual methods when large models are built.
6. Infrastructure for many possibilities of element influences has been developed.

2 Modeling approach

To study a large biological system like pancreatic cancer cells and their environment, discrete element-based models can offer an advantage over other model types such as Ordinary Differential Equation (ODE) models [14], or reaction rule-based models (RBMs) [15]. The ODE-based approaches require all biochemical reactions to be explicitly defined, which becomes impractical for hundreds or thousands of different reactions [15]. To avoid this, the RBMs introduced reaction rules, which can represent not one but many similar reactions [15]. However, even with fast ODE solvers, or the RBM approach, the exact element interaction mechanisms are often unknown or are very generally defined, and therefore, a more abstract approach can be used.

Defining models at a higher level of abstraction, by using a finite set of discrete value levels for model elements, and omitting some of the precise mechanism details, allows for capturing both direct and indirect interactions, as well as important feedback and feed-forward loops within a large system network, while avoiding significant (or impractical) increase in model analysis runtime [3, 16]. In general, the static structure of these networks can be illustrated as a directed graph, with the system components (i.e., model elements) as nodes, and the influences between components as directed edges [17]. Figure 2 is an example of one such graph, outlining the high-level influence map of cancer cells, involving the Warburg effect, DNA damaging agents, traumatic injury and M1 and M2 macrophages in the tumor microenvironment.

To study the system dynamics, we use *executable* hybrid element rule-based models developed using a granular computing approach [9]. In these models, we apply value granulation by defining a set of discrete values to represent each element's activity levels, and we assign an update rule to each element that is a function of the element's regulators [8, 9]. The hybrid element rule-based models can capture large systems at different levels of abstraction, and they are especially suitable for the analysis of dynamic behavior arising from the complexity of the system's interaction network [4, 18].

Furthermore, the hybrid element rule-based models are a natural fit for the outputs of Natural Language Processing (NLP) systems (e.g., REACH [5] and TRIPS [19]) due to the following: (i) the information extracted from published articles can be automatically assembled into executable hybrid element rule-based models, which can then be simulated to produce quantitative data on the state of the system, and (ii) the hybrid element rule-based models can handle information resolution varying from very dense to sparse.

Currently, the only other tool that we are aware of, that automatically creates models from NLP output is INDRA [20]. However, INDRA assembles reaction rule-based models, which, in contrast to the hybrid element rule-based models, are often not the best fit for machine reading output. Moreover, INDRA creates new executable models from NLP output only, which can be problematic, since the variability of language and representations can lead to incomplete or incorrectly extracted interactions. The executable hybrid element-rules based models are, on the other hand, well suited for (1) large system networks, (2) when the details about the system being modeled are not all available or known, and for (3) automated extension of models using the information extracted from literature.

3 Model assembly

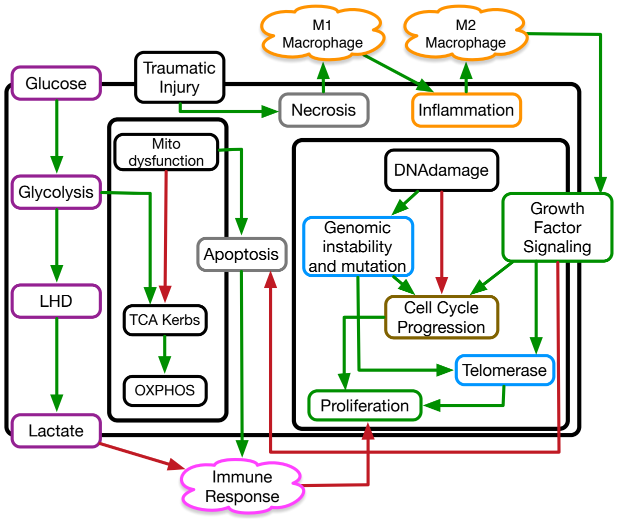


Figure 2. High level overview of the cancer network, including metabolic processes and components (purple boxes), the cellular compartments, plasma membrane, mitochondria and nucleus, and element regulations, positive activating the element (green arrows), and negative inhibiting the element (red arrows).

Correlations in data and new laboratory discoveries initiate questions of "Why?" and "How did this observation occur?" Investigators query the literature with search terms related to such questions in order to collect knowledge and assemble a structured influence map. If machines are reading, then the output must be in a standardized machine-readable format that is robust enough to represent knowledge extracted from articles. The automatically collected interactions are then translated from the reading format to the modeling format to allow for the simulations of executable models. Automated reading has difficulty including common sense connections that may not be stated explicitly in the literature, and therefore, it is important to have a human readable format.

After evaluating several alternative representations for the information that will be used to generate executable models, the tabular format described in [8] was selected as the most compact, readable by both humans and machines, with relevant information displayed for evaluating context and correctness. This format is flexible, columns can be added or removed as the study requires, and to keep records of associated identifiers, spatial and temporal information.

3.1 Standardized knowledge representation format

At the center of the DySE framework is the standardized BioRECIPES format for models [8]. The tabular format has model elements listed in the rows of the spreadsheet with element attributes relevant to the study included in the columns. If an element has regulators, then its positive and negative regulators are listed in the columns designated for the element's *influence set*.

A positive regulator can have an increasing, activating, stimulating or other positive influence on the behavior of the element. A negative regulator can have a decreasing, inhibiting or other negative influence on the behavior of the element. Therefore, the tabular format captures the *direction* (which element is the regulator) and the *sign* of the regulation (does the regulator have a positive or negative influence on the element).

The output of reading engines is very similar to the BioRECIPES format with the columns for element names, ID, positive regulators, negative regulators, and location being easily translated. Source literature ID and the evidence sentences can be retrieved but are not translated into the model format due to redundancies in the reading output. For example, the automated readers will extract "EGF positively regulates EGFR" from many papers and itemizing all of the evidence would decrease readability of the model spreadsheet.

As previously discussed [8], reverse phase protein arrays discovered that GAB2, AKT, β-catenin and PAI-1 were overexpressed in pancreatic cancer [21]. In order to find articles with information about the proteins under investigation, queries are formulated with protein synonyms and aliases using logical notation. Initial testing can reveal other common biological interactions that are not an appropriate context, and these are excluded using AND NOT. The following query was used to search for PMC (PubMed Central [22]) papers:

```
GAB2 AND (phosphatidylinositol OR proliferation OR
SHC1 OR PI-3 kinase OR PI3K OR PIK3 OR GRB2 OR PTPN11
OR 14-3-3 OR SFN OR YWHAH OR HCK OR AKT OR beta-catenin
OR Calcineurin OR SERPINE1) AND NOT (Fc-epsilon
receptor OR osteoclast OR mast cell)
```

Table 1. Simplified table showing the format of the automated reader output. The evidence sentences for the extracted interactions are provided below the table.

El. name	El. type	ID source	ID	Pos. reg.	Neg. reg.	Paper ID	Ev. #
RhoA	Protein	UniProt	P61586		Gab2	PMC3016968	1
Vav2	Protein	UniProt	P52735	Gab2		PMC3016968	2
RhoA	Protein	UniProt	P61586	Gab2		PMC3016968	3
Gab2	Protein	UniProt	Q9UQC2	RhoA		PMC3016968	4

1. "These data are consistent with a model in which Gab2 signals via p190A to suppress RhoA activation and downstream signaling by this GTPase. ++++ In addition, they suggest that the increased formation of cell protrusions by MCF-10A and Gab2 cells (XREF_FIG), which occurs without changes in Rac activation (XREF_FIG), reflects localized down-regulation of RhoA by Gab2 at the cell periphery."
2. "However, in the MCF-10A model, Gab2 overexpression, while decreasing RhoA activation, increased Vav2 phosphorylation."
3. "Consequently, Gab2 must signal upstream of RhoA to regulate cytoskeletal organization and cell migration."
4. "Instead, our finding that RhoA activation in MCF-10A and Gab2 cells was significantly enhanced early during the spreading process likely explains this effect . ++++ However, our detection of significantly reduced RhoA activation in MCF-10A and Gab2 cells identifies an additional mechanism whereby Gab2 enhances cell motility."

This query returned 249 papers, from which REACH and RUBICON reading engines extracted 4800 and 2450 events, respectively. The reading output provides the direction of the interaction, for example, GAB2 is a regulator of RhoA and Vav2, and it provides the sign of the regulation, GAB2 inhibits RhoA and GAB2 activates Vav2 (Table 1). Errors of direction and sign will be discussed further in Section 3.2.

The key components of the machine reading of biological articles are shown in Table 1. The comprehensive list of column headings is Element Name, Element Type, Database Name, Element Identifier, Location, Location Identifier, Cell Line, Cell Type, Organism, Positive Regulator Name, Positive Regulator Type, Positive Regulator ID, Positive Regulator Location, Negative Regulator Name, Negative Regulator Type, Negative Regulator ID, Negative Regulator Location, Interaction Direct or Indirect, Mechanism Type or Direction, Paper ID, Evidence.

Every model is an abstraction, and therefore, it is necessary to specify the level of abstraction and have methods to detail the *identity* of the element within the context of the desired model. For models of signaling pathways in biological systems, there are databases such as UniProt [23] that have collected gene and protein names and assigned unique IDs to these elements.

The tabular format allows for multiple IDs to be listed for protein families and protein complexes. For example, many enzymes in humans are the result of gene duplications to produce proteins that are part of a family and have similar function, such as AKT1 and AKT2 which both phosphorylate and inhibit ASK1. Where the two isoforms exhibit similar activities, the tabular format allows for the entry of multiple IDs to be considered for the same element. However, there is also the case where different proteins of the same family have different functions, as with ASK1 phosphorylating p38MAPK but ASK2 and ASK3 do not. This information is incorporated into the tabular format by listing the "NOT" IDs as separate elements, if encountered in papers, and specifying that their IDs are not to be included as a regulator.

Other NOT IDs are for proteins that are expressed in a tissue specific manner, for example, when a particular isoform of a protein is only found in brain tissue and not in pancreas. The IDs for these forms would be itemized to be excluded if encountered in the reading, for example, Q9GZT5, Wnt10A, is involved in development of ectoderm, tooth, tongue, sweat ducts and hair follicles so those relationships should be excluded from the model. This concern is minimized if the papers are preselected for tissue or disease context, for example, those related to pancreas development, or cancer, or for cancers that have a similar mutation profile.

It is also common in biology to have several proteins assemble into protein complexes for function. Often reading will extract elements and assign an ID, however, the elements actually form a protein complex, and therefore, a list of the IDs of the proteins included in the complex is necessary to specify the identity of the element. For example, in a model of signaling pathways in pancreatic cancer, the element MRN would represent UniProt IDs P49959 (MRE11), Q92878 (RAD50), O60934 (Nibrin). If the automated reading encounters one of the three IDs, then it can be represented by the MRN element. Another example, mTOR, the protein, is part of mTORC1 and mTORC2 so that mTOR is part of both complexes and Raptor is one of the proteins of mTORC1 and Rictor is part of mTORC2 (other element IDs listed also). Therefore, if mTOR is in the reading output, it cannot be assigned to a node without further information. If the reading output states that AMPK inhibits Raptor [24], then the interaction can be included as AMPK being a negative regulator of mTORC1. The tabular format keeps records of the identity of components and allows for a more parsimonious model.

It is difficult for investigators who are not deeply familiar with the biology to sort through these identities, however initiatives such as STRING [25] are collecting manually curated observations and can be automatically probed for validating the interactions extracted by automated reading and cross-referenced with UniProt IDs. The reading is still necessary to provide direction and sign of the interaction. This tool is currently being developed for use within the DySE framework.

The interactions that are assembled into a directed network are then ready for human curation to validate interactions prior to parameterization and simulation.

3.2 Common errors

Initial attempts to assemble models directly from reading produced models that were not usable to answer questions about the system. Scientific articles can be difficult for humans to read and understand, and therefore, it is even more difficult for machines to read and extract relationships accurately. The commonly occurring errors include errors of *omission*, *grounding*, *direction* and *sign*.

Scientific articles vary in the amount of information about interaction details, and therefore, the reading engines can extract anywhere from 0 to over 100 interactions per article. However, even though many interactions in the text are extracted, there are also many that are missed due to the language used to describe the interaction. The different readers use different algorithms to evaluate the text and therefore the errors of omission are different. Using multiple readers can improve coverage [26].

As experimental science progresses and more knowledge is generated from diverse fields such as biochemistry, molecular biology, yeast genetics, cell biology and cancer biology, there are often multiple names for the same protein (grounding or identity errors). For instance, AP-1 is Activator Protein 1 and is a transcription factor composed of c-fos and c-jun, or it is the adaptor protein 1 complex involved in clathrin recruitment and sorting. These two different areas of research, cell cycle and trafficking, selected short names for the complexes not aware that it was the same. There is often confusion that arises because the proteins and genes were discovered independently and named differently. For example, Cdk2 is also called Cdkn2 and is a cyclin-dependent kinase that phosphorylates cyclin B to proceed past the G1-S checkpoint, and Cdn2A is called Cdkn2A and inhibits the interaction of CDK2 and CDK4 with cyclin D to block progression through the G1-S checkpoint.

Errors of direction occur when there are sentences that are difficult to parse due to negative or double negative statements, or complex biological interactions and assumptions of experimental knowledge (i.e., knockout, siRNA knockdown, mutant, overexpression). In such cases, it is difficult for machines to correctly extract the meaning of text describing experimental results (Table 1, example 4). Machines can identify these errors if interactions for both directions are present. Machine readers also make errors of sign so that the regulator is positive instead of negative (Table 1, example 3). This is observed often when phosphorylation causes inactivation rather than activation. These errors can also be identified automatically, and the tabular format provides a compact view of the interactions and evidence for evaluation so that if common errors are detected they can be corrected. With filtering and scoring methods for determining the true and false interactions extracted by the automated readers, it will be possible to rapidly process the 4800 events (extracted by REACH machine reading of 249 papers related to GAB2) that would otherwise take days or weeks to assess manually.

The Filter For Understanding True Extractions (FLUTE) tool is being developed to use curated databases of protein-protein interactions to select those that have the most supporting evidence [27]. Experimental evidence, database annotations, and text mining results all contribute to determining how established an interaction is. Varying the level of evidence required for FLUTE to select an interaction can change the size and confidence of the selected interactions. While FLUTE is successful at lowering the number of errors in the final selected output, directionality and sign errors can be difficult to detect. Depending on the evidence threshold selected, the output has a 30-50% reduction in incorrect interactions.

If a model already exists for a system, then only new interactions are required to update the model. For this purpose, an automated method for comparing new information from reading with the existing model is useful. A tool to classify corroborations, contradictions and extensions is in development. New automated methods to increase the correctness (i.e., the percent of correct interactions) of the extension set will improve the quality of information being added to models. However, human intervention will always be required for assessing the truthfulness of the interactions and for making judgements about the relationships of multiple regulators.

4 Parametrization

Following assembly of the binary interactions extracted by machine reading, several parameters are considered including *regulator notation*, *motifs*, *input values* and *initial values*. The DySE tabular schema has a defined notation for representing various relationships between element's regulators, and numerical data can be used (where available) to further parametrize the relationship rules. Additionally, in this tabular schema, users can adjust relationships between elements that machines may have difficulty recognizing. These parameters, together with the positive or negative sign of the interaction, and the set of discrete values that each element is assigned, impact the functions that are used to update elements.

4.1 Notation for regulators

If there are multiple regulators of an element, then their influence can take multiple forms, and this is designated in the regulator notation. For example, regulators can act independently of each other (OR relationship in the logical functions), or regulators need to act together to have an effect (AND relationship in logical functions). More complex relationships may also exist between elements; for example, a notation has been developed to indicate that one regulator is necessary, and its regulatory effect is influenced by the presence of another [8].

For instance, A OR B may positively regulate C. Maybe A OR B negatively regulate C? It could be that A AND B are necessary to regulate C, or maybe A regulates C, and B enhances that relationship? In DySE, the combined influence of all element's regulators is considered to create element update functions, and a standardized notation is used to express the regulator relationships. In Table 2, we show in each row an example model element, and how the notation is used for positive regulators (**Pos. reg.**) and negative regulators (**Neg. reg.**). We also add a column that indicates the regulatory relationships (**Reg. rel.**), that is, leading operations in the corresponding element update rules. These rules are automatically generated from the tabular schema as logical or algebraic functions, and they are used to determine the next state of the element given the current state of the element and its regulators.

Table 2. Examples of the use of regulator notation in DySE.

El. name	Pos. reg.	Neg. reg.	Reg. rel.	Motif
Vav2	Gab2		A->B	
Cdc42		Shp2	A-|B	
RhoA		Gab2,Shp2	OR	
RhoA		(Gab2,Shp2)	AND	
Gab2gene	E2F1			Transcription - gene
Gab2rna	Gab2gene			Transcription - rna
Gab2	{HER2} [Gab2rna]			Transcription - protein

4.2 Motifs for timing

In biological signaling networks, the time necessary to change from one level of concentration or activity to another can vary even orders of magnitude (seconds to minutes to hours) among network components. When these networks are modeled as biochemical reactions, the timing of changes is usually indicated by reaction rates. However, the reactions and their rates are often not known, and only observations of indirect regulations are available. In such cases, a discrete element rule-based approach is suitable, modeling causal relationships between elements, and abstracting from the level of detailed biochemical reactions.

In DySE, when the mechanisms are known, we can use motifs to implement common interactions such as receptor activation, protein translocation and gene transcription, which enable incorporating into models the difference between slow and fast interactions. The column **Motif** in Table 2 lists motifs for several element regulation examples. For example, gene regulation takes much longer than posttranslational modifications such as phosphorylation, and the longer timing of gene regulation is implemented with the transcription and translation motifs. On the other hand, in the more abstract, logical and discrete models, receptor regulation is usually simplified such that external ligands are directly regulating internal pathways [18]. When the information about the receptor is known, the receptor activation motif allows for modeling this more accurately within the abstract model. Several motifs and their effect on the model of the circuitry that controls T cell differentiation are discussed in detail in [28]. Methods to further address the lack of information about timing in literature, in order to assemble useful models from extracted interactions are under development.

4.3 Inputs and initialization

To test, validate, and optimize models, we are often interested in exploring model responses to different *scenarios*, defined to mimic various experimental conditions. Input and initial value parametrizations are an integral part of scenario definitions.

All elements that do not have regulators are called *inputs*, and we can either fix their values during the studied time interval (i.e., the duration of the simulation, as will be described in Section 5), or alter their values during this interval (at specified simulation steps).

Scientific articles of human development and disease usually present either analytical data (human tissues analyzed by instrumentation), or experimental data collected from the use of experimental methods applied to a model system, such as a mouse model, cell line model, or coculture system. Information from these experimental models can be incorporated into the computational model via input value patterns and initial state values for elements. Inputs can be chemicals, processes, molecules, such as receptor ligands including cytokines or growth factors. Additional methods include knockouts to eliminate completely a protein, siRNA to decrease amounts of a specific RNA and protein, and inhibitor chemicals to decrease the activity of a protein upon addition. Overexpression experiments examine the effects of increasing the amount and activity of a protein, while mutant versions of proteins can be activating or inhibiting, depending on the specific mutation and protein. To model these experimental methods, additional variables are created and added as positive or negative regulators to the proteins that are targeted by the method. These variables include mutations (MUTPROTEIN), knockouts (KOPROTEIN), siRNAs (SIRNA), inhibitor compounds (INHPROTEIN) and overexpression (OEXPROTEIN). Furthermore, timing (see section 4) and weights can be included in expressions according to experimental methods. For example, 5*KOTP53 would be added as a regulator to TP53 and therefore none of the other regulators would influence TP53.

To study the dynamics of the system, we start with a specified system state, which we call the *initial state* for that particular study. Initial states are often defined to represent the healthy state if studying disease progression, or an early form in the development of a cell type or an organism, or the cell line prior to a drug study – it depends on the system being modeled and the experiments being conducted. The initial values can be derived from published data in articles, raw data in supplemental files, and values from databases, or determined from other sources of knowledge about the system. If knowledge is unavailable, initial values are set to a default value. Table 3 lists initial values for several elements of the pancreatic cancer cell microenvironment model outlined in Figure 2, for eight different scenarios (discussed in Section 5).

When data is available, it can be mapped to a set of several discrete values, and it can be used to determine initial values for model elements. While we have already developed methods to define input values and initial element values, and to test the effects of different initialization, we are currently working on fully automating this process.

Table 3. Pancreatic cancer analysis scenarios: the <u>initial values</u> of the inputs for the scenarios designed to show the development of pancreatic cancer.

	BL	S1	S2	S3	S4	S5	S6	S7
DNA damage	0	1	0	1	0	1	0	1
Traumatic injury	0	0	2	2	0	0	2	2
Tumor microenvironment	0	2	0	0	2	2	2	2

5 Simulation

Once the model is assembled and parametrized, and the scenarios are defined, the new executable model can be simulated using the DiSH simulator [9]. Data is also used for simulations, in particular, DySE utilizes data to describe the *properties* that emerge as the system progresses through time until a specified time point, or to a steady state. These properties are defined to also reflect initial questions for which the model was assembled.

5.1 The DiSH simulator and simulation outputs

The DiSH simulator [9] allows for model execution following several different schemes, created to recapitulate various trends in state changes observed in biological systems. Two main types of schemes include simultaneous scheme, where all elements change state at the same time, and sequential scheme, where the time of individual element state change can be random and thus, different from other elements. Each of these two types include several sub-types, that have different simulation parameters, which help more precisely define order of element state change or the frequency of changes. More details about the DiSH simulation schemes and the comparison between the schemes, illustrated using biological models, can be found in [9].

The model is simulated following the scenario specifications, that is, starting in the scenario-defined initial state and assuming the scenario-defined input value pattern. The output from simulation includes a trajectory (sequence) of values for each model element, from the given initial state, until the end of a specified time interval, or until a steady state is reached. In the case of stochastic simulation (i.e., sequential scheme), an average trajectory is computed for each element, for a given scenario.

5.2 Model validation and system explanation scenarios

Scenarios are used to represent well-known or common experimental conditions and outcomes, to calibrate and test the behavior of the assembled and parametrized executable model. In addition to the normal or default state of the system, we can also define a number of other scenarios to further explore system functionality, and to explore various experimental conditions with our *in silico* models. In the following, we describe how scenarios are used to set up simulations of the pancreatic cancer model.

First, following the procedure described in Sections 3 and 4, we assemble the model, by listing the elements involved in the studied system. Here, we define a *baseline scenario*, which assumes that all elements are initialized to 0, except glucose, which, being an input, is assigned a vector of values.

The baseline scenario represents healthy pancreas, the normal state of pancreas where there is no injury or DNA damage, and the increase in glucose input will result in oxidative phosphorylation (Table 3). Scenarios for tissue culture cell lines, inflammation and cancer have various inputs for DNA damage, traumatic injury and the tumor microenvironment (Table 3). We designed these different scenarios to explore the response of the model when activating individual inputs versus the combinations of inputs. Notice that when multiple inputs are active, element responses in time are more complex, due to the stochasticity in timing of element interactions (Table 4, also indicating when transient behavior is observed in simulations).

The scenarios 1, 2, and 3 simulate cell lines where the macrophages of the tumor microenvironment are not present and show the outcomes of DNA damage, traumatic injury and the two combined stresses. DNA damage (Scenario 1) causes apoptosis, cell cycle progression, genomic instability and high lactate production due to mitochondrial dysfunction. Injury causes no change in cell lines without the macrophages (Scenario 2).

When macrophages are included in the model with DNA damage (Scenario 5) the outcomes are similar to Scenario 1, however, when macrophages are present with injury (Scenario 6) there is cell cycle progression and inflammation but not sustained proliferation. Only when DNA damage and injury occur in the presence of macrophages (Scenario 7) proliferation is sustained, and therefore, cancer is observed This is illustrated in Figure 3a, with average simulation trajectories for Apoptosis, OXPHOS and Proliferation. Cancer requires cell cycle progression AND telomerase to maintain sustained proliferation. We call Scenario 7 the *cancer scenario*.

To test treatments (interventions) in DySE, we often further extend both the model and the scenarios that were used to study the model. We can extend the model by adding new elements and interactions, and we can extend scenarios by defining new initializations that represent drug treatments. For example, if our goal is to test inhibitors, then this would be an additional scenario, and not an extension of the model structure.

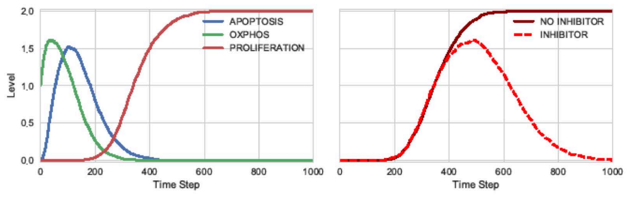
Table 4. Pancreatic cancer analysis scenarios: the <u>end values</u> of the simulations of the scenarios and transients (e.g. 0.5->0 means that there was an increase to 0.5 but then the end value was 0).

	BL	S1	S2	S3	S4	S5	S6	S7
Apoptosis	0	2	0	2	0	2	0	1.5->0
Cell cycle progression	0	2	0	2	0	2	2	2
Genomic instability	0	2	0	2	0	2	0	2
Immune	0	0	0	0	0	0.5->0	0	0.5->0
Inflammation	0	2	0	2	0.5->0	2	0.5->0	2
Lactate	2	0	2	0	2	0	2	0
OXPHOS	0	2	0	0	2	0.5->0	2	0
Sustained proliferation	0	0	0	0	0	0	0	2

5.3 Intervention scenarios

Once the model has been tested and sufficiently validated under all scenarios of interest (here, baseline and cancer scenarios), then the model can be used to test interventions such as inhibitors. Scenario 8 is an intervention scenario with an inhibitor of inflammation (implemented as a negative regulator of inflammation), INHinflammation, is introduced during simulation at a time step corresponding to the assumed start of the increase in inflammation (time step 200 in simulations). The plots in Figure 3b show that cancer cell proliferation decreases under this scenario, which can be interpreted as cancer reduction.

The model described here and outlined in Figure 2 is very simple and does not include the details of the signaling pathways involved in cancer. Attempts to manually assemble models of greater than 200 nodes have failed to produce robust models perhaps due to the difficulties of biased knowledge and level of abstraction. Automation can test many more extensions and scenarios than can be performed manually, therefore, automation is needed to construct models of large complex systems. The originally built model can now be used as a *baseline model* for iteratively creating a more detailed model of cancer signaling pathways. In DySE, this is done through automated model extension and, in the case of the pancreatic cancer model, by extending the model with the pathway modules involved in pancreatic cancer.



(a) (b)

Figure 3. Simulation results, element trajectories for (a) Apoptosis, OXPHOS and Proliferation under Scenario 7, and for (b) Proliferation without an inhibitor of inflammation (No inhibitor), and with the inflammation inhibitor added at time step 200 (Inhibitor), Scenarios 7 and 8, respectively. The x-axis represents simulated time steps, and the y-axis represents average element level; in this model, each element can have one of the three values 0 (no activity), 1 (low activity), and 2 (high activity).

6 Iterative model analysis and improvement

Automated methods are required to update and extend the models with newly obtained or published information. In the case of pancreatic cancer, this could include more details of cancer signaling pathways, such as RAS/ERK, TNFa/NFkB, JAK/STAT, TGFb/SMAD, AMPK/mTORC1, PI3K/AKT, ASK/JNK, lactate metabolism, ROS signaling and DNA damage response, additional tumor microenvironment molecules and immune cell types. As mentioned previously, it is preferable to iteratively validate clusters of additions. In DySE, iterative model validation and extension is integrated with simulation and model checking to enable fast and reliable model selection [12]. One of the methods used for model extension includes combination of Markov Clustering algorithm (MCL) and pathfinding strategies to identify clusters within the network of all available or relevant element interactions. Breadth First Addition (BFA), and Depth First Addition (DFA), based on graph search algorithms and model performance metric, are used to directly choose interactions to add to the model. Finally, the Genetic Algorithm-based method [13] is a combined clustering and addition method that uses the best clusters (again, based on a model performance metric) to form the next set of clusters. Statistical model checking is used to judge the performance of the extended models. This iterative assembly of new model versions allows for the slow growth of the model with selected knowledge (see the next section for a description of the selection criteria).

6.1 Statistical model checking

The clusters of new interactions are used to extend (and update) the baseline model. All extended model versions are then judged with the statistical model checking method [11], which investigates model simulation trajectories, and therefore, provides a metric for selecting the best model(s). More specifically, the performance metric is computed with respect to the probabilities of satisfying defined system properties.

The model checking tools use properties written in a format readable by machines [12]. For example, if an element `EL` is expected to assume value 1 within the first 2000 time steps, and if it needs to hold this value for at least 3000 steps, then we would write this formally as:

```
F[2000](G[3000](EL == 1))
```

In the case of pancreatic cancer model, and the scenario outcomes listed in Table 4 (only OXPHOS high), the corresponding properties are shown below:

```
F[400](G[600](apoptosis_0 == 0 & apoptosis_1 == 0 &
apoptosis_2 == 0))
F[400](G[600](cellcycleprogression_0 == 0 &
cellcycleprogression_1 == 0 & cellcycleprogression_2
== 0))
F[400](G[600](genomicstability_0 == 0 &
genomicstability_1 == 0 & genomicstability_2 == 0))
F[400](G[600](immune_0 == 0 & immune_1 == 0 &
immune_2 == 0))
F[400](G[600](inflammation_0 == 0 & inflammation_1 ==
0 & inflammation_2 == 0))
F[400](G[600](lactate_0 == 0 & lactate_1 == 0 &
lactate_2 == 0))
F[400](G[600](OXPHOS_0 == 0 & OXPHOS_1 == 2 &
OXPHOS_2 ==2))
F[400](G[600](proliferation_0 == 0 & proliferation_1
== 0 & proliferation_2 == 0))
```

Similarly, these eight properties would be written for all seven scenarios for the baseline model, as well as for all extended models analyzed in all scenarios.

Using the baseline model, scenarios, properties, and a list of potential extensions obtained from automated reading or other source, it is possible to select automatically those extensions to the model that will provide a model version that satisfies the properties. This iterative process tests different clusters of extensions to find those closely connected to the model and relevant to original user queries. The model versions built from the baseline model and clusters of newly obtained extensions that satisfy properties are analyzed further to discover explanations to questions. Fully automating the overall process is an ongoing direction of development.

6.2 Sensitivity analysis

When models become large, it is necessary to automate methods for identifying important paths and the most influential elements within the model network. Sensitivity analysis can be used not only to study the influence from immediate, directly connected regulators, but also to explore global influences from indirectly connected elements. A comprehensive sensitivity analysis approach has been proposed in [29], where both static and dynamic sensitivity analysis are performed. The static analysis is based only on element update rules, and thus, offers insights into the topological structure of the model, while the dynamic analysis uses simulation results to incorporate the context-based bias.

Our sensitivity analysis approach enables the extraction of globally important pathways, which is critical in studying biological systems. Once we identify these pathways, we can design control strategies and interventions to tune system inputs or alter element values during the transient process. Through the construction of a weighted directed graph for the model, we have reduced the problem of extracting important paths to a minimum-cost graph search problem. We have also validated our sensitivity analysis results using the model checking approach and demonstrated that element influence (especially in dynamic analysis), outperforming many other network attributes, is a useful measure in pathway extraction [29].

7 Evolution, learning and reasoning

The DySE framework creates models that evolve, that develop through iterations as new information and knowledge (traits) are tested (selected) and those that increase performance (fitness) of the model are maintained in the new model versions (offspring). This modeling approach is able to account for feedforward and feedback of information as it influences elements and includes the effects of timing within the system. With correct information, computational models and simulations represent the understanding of interactions and relationships of entities and the changes that occur over time. With a validated model, prediction and intervention testing provide a greater understanding of health and disease and improve the practice of medicine. These methods are not domain specific and can be implemented for any complex system including other natural systems of development, disease, agriculture, environment, climate and social systems such as geopolitical, economic or military or other world systems. Obtaining the knowledge required for assembling directed causal interactions into models is difficult, however this modeling approach can provide insights for the design of critical experiments to determine unknown relationships of cause and effect. The benefits of having causal models are enormous, the propagation of influence from a seemingly minor intervention, the feedback from a faulty decision, or feedforward to a synergistic improvement, these are all solvable with this modeling framework.

8 Conclusions

The DySE framework is a powerful platform for creating models to incorporate data or explain correlations in data with causative mechanisms. The directed nature of the modeled interactions, ability to incorporate feedforward and feedback loops, and iterative selection to grow the model, form a framework that learns and reasons about data to provide explanations rooted in knowledge.

Automated reading and incorporation of data, a standardized notation in an interactive tabular schema, coupled with iterative growth and testing of models and scenarios, results in an intelligent system for learning and reasoning from text and data.

ACKNOWLEDGMENTS

This work is supported by DARPA grants W911NF-17-1-0135 and W911NF-18-1-0017-(72314-NS-DRP) awarded to N. Miskov-Zivanov. We would also like to thank Dr. Peter Spirtes and Dr. Michael Lotze for their feedback and expert insights while developing DySE and applying it to the cancer microenvironment domain.

REFERENCES

[1] Richard Dawkins. 1978. *The selfish gene*. Oxford University Press, Oxford, United Kingdom.
[2] Malgorzata Stojek and Jacek Pietraszek. 2015. Simulation-Based Engineering Science Challenges of the 21st Century. *Applied Mechanics and Materials*, 712 (2015), 3-8.
[3] Rui-Sheng Wang, Assieh Saadatpour and Reka Albert. 2012. Boolean modeling in systems biology: an overview of methodology and applications. *Physical biology*, 9, 5 (2012), 055001.
[4] Natasa Miskov-Zivanov, Michael S Turner, Lawrence P Kane, Penelope A Morel and James R Faeder. 2013. The duration of T cell stimulation is a critical determinant of cell fate and plasticity. *Sci. Signal.*, 6, 300 (2013), ra97-ra97.
[5] Marco A Valenzuela-Escárcega, Gus Hahn-Powell, Mihai Surdeanu and Thomas Hicks. 2015. A domain-independent rule-based framework for event extraction. *Proceedings of ACL-IJCNLP 2015 System Demonstrations* (2015), 127-132.
[6] Gully APC Burns, Pradeep Dasigi, Anita de Waard and Eduard H Hovy. 2016. Automated detection of discourse segment and experimental types from the text of cancer pathway results sections. *Database* (2016).
[7] James Allen, Will de Beaumont, Lucian Galescu and Choh M Teng. *Complex event extraction using DRUM*. Florida Institute for Human and Machine Cognition Pensacola United States, 2015.
[8] Khaled Sayed, Cheryl A Telmer, Adam A Butchy and Natasa Miskov-Zivanov. 2017. Recipes for translating big data machine reading to executable cellular signaling models. In *Proceedings of the International Workshop on Machine Learning, Optimization, and Big Data*, Springer, Tuscan, Italy, 1-15.
[9] Khaled Sayed, Yu-Hsin Kuo, Anuva Kulkarni and Natasa Miskov-Zivanov. 2017. DiSH simulator: Capturing dynamics of cellular signaling with heterogeneous knowledge. In *Proceedings of the Proceedings of the 2017 Winter Simulation Conference*, IEEE Press, Las Vegas, USA, 64.
[10] Natasa Miskov-Zivanov, Paolo Zuliani, Qinsi Wang, Edmund M Clarke and James R Faeder. 2016. High-level modeling and verification of cellular signaling. In *Proceedings of the 2016 IEEE International High Level Design Validation and Test Workshop (HLDVT)*, IEEE, Santa Cruz, USA, 162-169.
[11] Qinsi Wang, Natasa Miskov-Zivanov, Bing Liu, James R Faeder, Michael Lotze and Edmund M Clarke. 2016. Formal modeling and analysis of pancreatic cancer microenvironment. In *Proceedings of the International Conference on Computational Methods in Systems Biology*, Springer, Cambridge, United Kingdom, 289-305.
[12] Kai-Wen Liang, Qinsi Wang, Cheryl Telmer, Divyaa Ravichandran, Peter Spirtes and Natasa Miskov-Zivanov. 2017. Methods to expand cell signaling models using automated reading and model checking. In *Proceedings of the International Conference on Computational Methods in Systems Biology*, Springer, Darmstadt, Germany, 145-159.
[13] Khaled Sayed, Kara N Bocan and Natasa Miskov-Zivanov. 2018. Automated Extension of Cell Signaling Models with Genetic Algorithm. In *Proceedings of the 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, 5030-5033.
[14] Wayne Materi and David S Wishart. 2007. Computational systems biology in drug discovery and development: methods and applications. *Drug discovery today*, 12, 7-8 (2007), 295-303.
[15] Lily A Chylek, Leonard A Harris, Chang-Shung Tung, James R Faeder, Carlos F Lopez and William S Hlavacek. 2014. Rule-based modeling: a computational approach for studying biomolecular site dynamics in cell signaling systems. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine*, 6, 1 (2014), 13-36.
[16] Kevin Gilboy, Khaled Sayed, Niteesh Sundaram, Kara Bocan and Natasa Miskov-Zivanov. 2018. A Faster DiSH: Hardware Implementation of a Discrete Cell Signaling Network Simulator. *arXiv preprint arXiv:1811.07861* (2018).
[17] Natasa Miskov-Zivanov. 2015. Automation of biological model learning, design and analysis. In *Proceedings of the Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, ACM, Pittsburgh, USA, 327-329.
[18] Ranran Zhang, Mithun Vinod Shah, Jun Yang, Susan B Nyland, Xin Liu, Jong K Yun, Réka Albert and Thomas P Loughran. 2008. Network model of survival signaling in large granular lymphocyte leukemia. *Proceedings of the National Academy of Sciences*, 105, 42 (2008), 16308-16313.
[19] George Ferguson and James F Allen. 1998. TRIPS: An integrated intelligent problem-solving assistant. In *Proceedings of the Aaai/Iaai*, Madison, USA, 567-572.
[20] Benjamin M Gyori, John A Bachman, Kartik Subramanian, Jeremy L Muhlich, Lucian Galescu and Peter K Sorger. 2017. From word models to executable models of signaling networks using automated assembly. *Molecular systems biology*, 13, 11 (2017).
[21] Yu-Jing Huang, Marsha L Frazier, Nianxiang Zhang, Qian Liu and Chongjuan Wei. 2014. Reverse-phase protein array analysis to identify biomarker proteins in human pancreatic cancer. *Digestive diseases and sciences*, 59, 5 (2014), 968-975.
[22] *PubMed Central*. Online [https://www.ncbi.nlm.nih.gov/pmc/], 2019.
[23] *UniProt Database*. Online [https://www.uniprot.org/], 2019.
[24] Dana M Gwinn, David B Shackelford, Daniel F Egan, Maria M Mihaylova, Annabelle Mery, Debbie S Vasquez, Benjamin E Turk and Reuben J Shaw. 2008. AMPK phosphorylation of raptor mediates a metabolic checkpoint. *Molecular cell*, 30, 2 (2008), 214-226.
[25] *STRING: functional protein association networks*. Online [https://string-db.org/], 2019.
[26] Benjamin M Gyori, John A Bachman, Kartik Subramanian, Jeremy L Muhlich, Lucian Galescu and Peter K Sorger. 2017. From word models to executable models of signaling networks using automated assembly. *Molecular systems biology*, 13, 11 (2017), 954.
[27] E. Holtzapple, Telmer, C., Miskov-Zivanov, N. 2019. A Filter for Fast and Accurate Information Retrieval in Biology. *In preparation* (2019).
[28] Khaled Sayed, Cheryl A Telmer and Natasa Miskov-Zivanov. 2016. Motif modeling for cell signaling networks. In *Proceedings of the 2016 8th Cairo International Biomedical Engineering Conference (CIBEC)*, IEEE, Cairo, Egypt, 114-117.
[29] Gaoxiang Zhou, Kai-wen Liang and Natasa Miskov-Zivanov. 2018. Sensitivity Analysis of Discrete Models and Application in Biological Networks. In *Proceedings of the Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, ACM, Washington, DC, USA, 605-606.