Putting Humpty-Dumpty Together: Mining Causal Mechanistic Biochemical Models from Big Data

Faraz Hussain, Alvaro Velasquez, Emily Sassano, and Sumit Kumar Jha
Department of Electrical Engineering and Computer Science
University of Central Florida, Orlando FL
Email: {fhussain, velasquez, esassano, jha}@eecs.ucf.edu

***Abstract*—In traditional engineering disciplines, the construction of a system is usually preceded by a formal or informal specification of the design of the system being developed. In biochemical applications, however, a detailed specification of the system's structure and dynamics is usually unavailable. Thus, mechanistic details of biochemical systems must be mined from experimental observations. In this paper, we adopt a formal methods approach towards deriving causal mechanistic models from time-series observations of biochemical systems. The mined model captures causality among multiple biological events and also allows causal relationships between sets of events. We exploit results from trace theory and use the power of powerful constraint solvers to develop a new framework for causality identification and reasoning that captures dynamic relationships among species in biochemical reaction networks.**

I. Introduction

A fundamental challenge in computational systems biology is the algorithmic construction of mechanistic models from experimental observations. Traditionally, machine learning methods have been used to create predictive (but non-mechanistic) computational models that can reproduce the behavior of the biological experiments using *in silico* simulation. While such models are important, they do not provide an insight into the causal structure or the "engineering blueprint" of the biological system being studied.

Over the last hundred years, biologists and chemists have undertaken the painstaking task of putting together significant fragments of mechanistic models that explain causality relationships among biochemical pathways. In recent years, this process has accelerated due to the development of modern experimental methods and computational techniques that enable researchers to perturb components of a biochemical system and easily observe consequent behavioral changes [13].

By the beginning of this century, our ability to develop models of biochemical systems has reached a significant turning point because of three primary reasons:

- Modern computer-aided data acquisitions techniques, such as flow cytometry, are creating a deluge of high-dimensional data sets that cannot even be visualized directly by human experts. Hence, biochemists are forced to obtain statistical summaries from such huge data sets - often losing valuable information in the process.
- The human mind is not capable of analyzing complex models and, hence, we engineer artificial systems using modular designs with limited but well-understood interactions to gain a better understanding of these models. Biology provides a unique challenge to this modular and compositional design approach – where small well-understood modules are composed to form larger and more complex systems. While we have achieved a lot of success in understanding individual pathways using human intelligence, the asynchronous cross-talk interaction among pathways makes it unlikely that we can unravel large biochemical pathways without significant computational support.
- Commodity computing clusters can easily deliver TFLOPs of computational performance. Algorithmic discovery of mechanistic biochemical models is computationally expensive, but it is now within the grasp of these modern high-performance computing clusters. The rise of cloud-based computing and cyberinfrastructure that can be deployed across such elastic clouds will permit biologists and other end-users to employ algorithms for mining such causal mechanistic models without owning or maintaining large computing clusters.

The problem of obtaining specifications from observations of engineered systems has gained a lot of attention recently. Specification mining has been studied in different domains such as hardware design, web services, programming languages and software engineering. Most of these techniques are targeted towards a narrow class of specifications, such as decision-tree learning [11] (which produces a single output for a sequence of decisions) or PROLOG/LISP learning techniques [16] which use declarative languages. These techniques introduce a learning bias and restrict mined specifications to mostly functional properties. Our goal is to learn more expressive specifications, and in particular, dependency constraints among multiple biochemical species; such dependencies can capture sequential constraints as well as concurrency.

Unlike many manually engineered systems, biochemical systems are known to be composed of inherently concurrent asynchronous components – often working together in a shared space and exposed to noisy cross-talk. Thus, traditional methods for specification mining used in software and hardware systems do not readily extend to biochemical systems. However, the mining of causal mechanistic models is important to computational systems biology as such algorithms will aid in the development of large and complex biochemical models.

In this paper, we are interested in identifying causal relationships among multiple biochemical events. In our frame-

978-1-4673-1321-6/12/$31.00

work, one or more reactant biochemical species may be the *cause* of a biochemical reaction and, in turn, the reaction causes change in the concentration of one or more product biochemical species (besides the reactants). For many realistic systems, including those in systems biology, it is difficult to distinguish true dependencies from apparent dependencies. A primary source for this confusion is the challenge of disambiguating between the *happens-before* and the *causality* relationship. A detailed capture of all dependencies in the observed experimental traces of a biological system would also identify many such fortuitous dependencies. Thus, weeding out these incidental dependencies to capture *true* dependencies is another important aspect of this problem. For example, in a signaling pathway, some observations may correspond to the simultaneous up-regulation of two proteins without any causal link. The goal of dependency identification is to track causality dependencies and filter out incidental dependencies from a set of randomly observed traces.

Specifications that capture dependency relationships of events in a biochemical system are of immense value to researchers who want to *calibrate* a given biochemical model to achieve desired behavior. Translational systems biologists can then employ this causal structure to identify genes and proteins that may be regulated to alter clinical outcomes. Causal models can also be used to create clinical tests for identifying diseases and precursors of diseases. The goal of our work is to provide an *algorithmic framework for automating the learning of these causal dependencies*.

We discuss related work in Section II. We then describe an approach in Section III to mine dependency constraints from observed traces of biological systems which abstracts incidental dependencies and only summarizes dependencies that are likely to correspond to causality. In Section IV, we describe how we formulate the problem in order to harness the power of modern SMT solvers [8]. We then describe (Section V) how to use the set of binary causal relationships generated by the constraint solver to construct a macro-model of the biochemical system. and then describe our algorithm for learning causal mechanistic models from experimental data. In Section VI, we conclude by mentioning some limitations of our technique and directions for future work.

II. Related Work

Granger's 2003 Nobel Lecture [4] highlights one natural definition of causality that relies on temporal priority of the cause as well as the causal event altering the probability of occurrence of the effect:

- Temporal Priority: The cause occurs *before* the effect; and
- Information Content: The cause contains information about the effect that is *unique* and this information is not contained in any other variable.

Thus, Granger defines a notion of causality where X is said to cause Y if the past values of X can be used to predict the future value of Y beyond what could have been done with the past values of Y only. Granger causality has been employed in a number of interdisciplinary areas, most notably econometrics. It should be noted that Granger attributes his own thoughts on the foundational principles of causality to some of Weiner's earlier works [3].

The ideas of temporal priority and predictive information about the effect in the cause are also applicable to biochemical pathways. However, Granger's causality test is not directly applicable to the discovery of causal mechanisms in biochemical reactions for two reasons:

- A cause in a biochemical reaction can be a subset of the biochemical species and the effect can be another subset of biochemical species. Thus, the number of potential causes and effects that need to be explored can be exponential in the number of biochemical species. Further, Granger's test is implemented primarily for univariate time series data and such univariate implementations do not incorporate the notion of multiple causes influencing multiple effects.
- Individual biochemical species in a mechanistic model of a biochemical reaction pathway may not satisfy Granger's second condition of causality i.e. the information in the causal biochemical species about the effected biochemical species may not be unique.

Mining of specifications from system traces has been extensively studied in literature. The related work can be classified across three dimensions. The first dimension is the nature of learning techniques used for inferring dependency relations. These include logical induction techniques such as inductive logic programming or probabilistic techniques such as Bayesian network inference. The second dimension is the nature of the target model: temporal logic, PROLOG, probabilistic graph models etc. The third dimension is the domain in which it is applied since this problem has been independently investigated across various domains including hardware design, web-service optimization and software engineering.

We discuss the existing specification mining techniques across all the three dimensions. Inductive logic programming [12] is a mature field which deals with learning inductive logic rules from traces. Task networks [14] which are non-probabilistic graphical models have been used to capture dependencies from traces. Probabilistic graphical models have also been used to infer dependencies for the purpose of diagnosing faults in a circuit by Jha et al [7]. Learning sequential dependencies from traces in software is also well-studied in literature. Lau et al [10] propose a vector algebra based approach to infer programs from traces. Yuan et al [17] propose a diagnosis approach based on using system call traces in operating systems to infer problematic system call interactions. Inferring branching programs from examples has also been explored in the program synthesis literature [5].

Our work differs from existing literature in the way dependency or causality rules are learnt from observed traces of biological systems. We employ a novel Satisfiability Modulo Theory (SMT) based technique for studying the impact of multiple causes on multiple effects that can automatically filter out fortuitous dependencies.

III. Inferring Dependency Constraints from Traces

In this section, we discuss how observed experimental data can be used to mine dependency constraints and hence establish causality among different biochemical species.

A. Biochemical Events and Traces

We know that a biochemical reaction decreases the amount of its reactants and increases the concentration of its products. Hence, we consider two types of events – increase or decrease of concentration of biochemical species. We will use sets of these events to discover a *set of causes* (chemical reactants) leading to a *set of effects* (chemical products).

Let the set of possible events in an experimental observation be $\mathbb{E} = \{E_1, E_2, \ldots, E_n\}$. Each event E_i indicates an increase or decrease in the concentration of a biochemical species. We denote by $\mathbb{S}$ all non-empty subsets of the set of events $\mathbb{E}$. In the context of mechanistic modeling of biochemical systems, $\mathbb{S}$ indicates all possible collections of causes and effects i.e. biochemical reactants and products.

Definition 1 (Trace as a sequence of event-(sub)sets)**.** *$\langle (E_{11}, E_{12}, \ldots), \ldots, (E_{m1}, E_{m2}, \ldots E_{mj}), \ldots \rangle$ is called a trace. Here, $(E_{m1}, E_{m2}, \ldots E_{mj}) \in \mathbb{S}$ denotes the m-th subset of events in the trace and all the events E_{mj} are events in $\mathbb{E}$.* □

Thus, a trace of a biochemical system is a temporal sequence of observations of increases and decreases in concentrations of biochemical species. At any time instant, the concentration of a biochemical species may increase or decrease, thereby creating a recordable event, or the concentration may remain unchanged. In the latter case, no event is recorded in the trace.

Given a trace σ, its subtrace σ_i $(0 \leq i \leq |\sigma|)$ is the prefix of the trace up to the i-th event subset. Usually only a small number of simultaneous events happen at the exact same time, as a biochemical reactions often involves only a small number of reactants and a small number of products. For generality, we include the occurrence of simultaneous events in our framework. Let us consider a biochemical system with four biochemical entities - $C_1, C_2, C_3,$ and C_4. Let E_i $(1 \leq i \leq 4)$ denote the event that the concentration of C_i $(1 \leq i \leq 4)$ has decreased in this time step and E_i $(5 \leq i \leq 8)$ denote the event that the concentration of C_i $(1 \leq i \leq 4)$ has increased (as compared to the previous time step). A few sample traces from such a biochemical system are shown in Table I.

TABLE I: Traces from a sample biochemical system

trace 1	(E_1, E_6)	(E_2, E_5)	(E_3, E_4, E_5)	
trace 2	(E_3, E_4)	(E_5)	(E_6)	
trace 3	(E_2)	(E_5)	(E_2, E_5)	
trace 4	(E_3, E_4, E_5)	(E_2, E_1)	(E_5, E_6)	(E_2, E_5)

Note that while all observed traces of the biochemical system are independent, each trace is obtained from the same underlying biochemical mechanism. This stochasticity is a reflection of the uncertainty inherent in the interactions of the biochemical species in the system.

B. From Biochemical Traces to Parikh Maps

We now transform the sequence of sets of events describing the trace of a biochemical system into a sequence of vectors of integers. This is important as Satisfiability Modulo Theory solvers are more efficient at computing linear arithmetic constraints. For this purpose, we first define a modified Parikh vector projection of the traces. Such vectors were first used by Parikh [15] to present a technique for deciding whether a string with a given number of various literals is accepted by a context-free language.

Definition 2 (Parikh vector over events)**.** *Given the set of events $\mathbb{E} = \{E_1, E_2, \ldots, E_n\}$, the Parikh vector $p_{\mathbb{E}}$ is a mapping from $\mathbb{E}^* \rightarrow \mathbb{N}^n$*

$$p_{\mathbb{E}}(w) = (\#E_1(w), \#E_2(w), \ldots, \#E_n(w))$$

where $\#E_k(w)$ denotes the number of E_k events in a sequence of events $w \in \mathbb{E}^*$[1]*.* □

Thus, the Parikh vector summarizes the count of each event in a given sequence of events. We now lift the Parikh vector mapping to *sequences of sets of events*.

Definition 3 (Parikh vector over sets of events)**.** *Given the set of events $\mathbb{E} = \{E_1, E_2, \ldots, E_n\}$, and with $\mathbb{S} = \mathbb{P}(E) \setminus \phi$, define the function $p_{\mathbb{S}} : \mathbb{S}^* \rightarrow \mathbb{N}^n$ where:*

$$p_{\mathbb{S}}(w) = (\sum_{s \in w} \#E_1(s), \sum_{s \in w} \#E_2(s), \ldots, \sum_{s \in w} \#E_n(s))$$

where $\#E_k(s)$ denotes the number of E_k events in a subset s and $\sum_{s \in w} \#E_k(s)$ denotes the sum of the number of E_k events in all the subsets s in a sequence of subsets $w \in \mathbb{S}^*$. □

As an example, by slightly abusing notation to let $\mathbb{S}(\mathbb{E})$ represent the power-set of $\mathbb{E}$ without the empty set, for $E = \{E_1, E_2, \ldots, E_8\}$, $p_{\mathbb{S}(\mathbb{E})}(\langle (E_1, E_6), (E_2, E_5), (E_3, E_4, E_5), (E_2, E_7, E_8) \rangle) = (1, 2, 1, 1, 2, 0, 1, 1)$.

We now extend the Parikh mapping to a trace. The key idea is to consider all prefix sub-traces and apply the modified Parikh vector mapping to each prefix of the trace to construct a trace of Parikh vectors from a given event trace.

Definition 4. *The modified Parikh vector mapping p of a trace σ is defined as follows.*

$$p(\sigma) = \langle p_{\mathbb{S}}(\sigma_1)\ p_{\mathbb{S}}(\sigma_2)\ \ldots\ p_{\mathbb{S}}(\sigma_m) \ldots p_{\mathbb{S}}(\sigma_{|\sigma|}) \rangle$$

Here, σ_i is a prefix of the trace σ up to the i^{th} event subset. □

We now use the example in Table I to illustrate the Parikh vector mapping of traces. The modified Parikh vector mapping $p(\sigma)$ of the traces in Table I is presented in Table II.

IV. Satisfiability Modulo Theory (SMT) for Causality Analysis

A. Causality Constraints from Biochemical Traces

The Parikh vector mapping $p(\sigma_i)$ embeds each prefix of every trace $\sigma_i (1 \leq i \leq |\sigma|)$ into $|\mathbb{E}|-$dimensional space.

[1] Given a set E of symbols, E^* is the set of all possible finite-length strings generated from E.

TABLE II: Parikh-mapping of the traces in Table I

trace 1	(E_1, E_6) (10000100)	(E_2, E_5) (11001100)	(E_3, E_4, E_5) (11112100)	
trace 2	(E_3, E_4) (00110000)	(E_5) (00111000)	(E_6) (00111100)	
trace 3	(E_2) (01000000)	(E_5) (01001000)	(E_2, E_5) (02002000)	
trace 4	(E_3, E_4, E_5) (00111000)	(E_2, E_1) (11111000)	(E_5, E_6) (11112100)	(E_2, E_5) (12113100)

The relation between the count of events can be written down as constraints in this $|\mathbb{E}|-$dimensional space. Let the following constraint be satisfied by the elements of $p(\sigma)$, where $\texttt{cause}, \texttt{effect} \subseteq \mathbb{E}$:

$$\left(\sum_{i \in \texttt{cause}} \#E_i\right) \geq \#E_j \tag{1}$$

The above constraint implies that E_j event happens together with or after E_i events in the set of events $\texttt{cause} \subseteq \mathbb{E}$. So, each of E_i events in cause can be a *cause* for E_j.

If we further generalize our constraint template to:

$$\sum_{i \in \texttt{cause}} \#E_i \quad \geq \quad \sum_{j \in \texttt{effect}} \#E_j \tag{2}$$

The above constraint implies that the E_j events in the set of events $\texttt{effect} \subseteq \mathbb{E}$ occurs together with or after E_i event in the set of events $\texttt{cause} \subseteq \mathbb{E}$. So one or more of the E_i events in cause can cause one or more of E_j events in effect.

Using the example traces considered in Table I and the Parikh mappings in Table II, we can study the $8-$dimensional event space and the mapping of the 4 traces and 13 Parikh-maps arising from these four traces into the $8-$dimensional space.

We can make some observations about the number of events for each E_i in all the components of the Parikh-map $p(\sigma_i)$ for each trace σ_i $(1 \leq i \leq 4)$:

- The number of E_1 events is no less than the number of E_6 events, that is, $\#E_1 \geq \#E_6$. Hence, we can conjecture that E_6 is *caused* by E_1.
- The number of E_3 and E_4 events is also no less than the number of E_5 events, that is, $\#E_3 \geq \#E_5$, $\#E_4 \geq \#E_5$. So, we can conjecture from the traces that E_5 is *caused* by E_3 and E_4.
- The number of E_2 events is no less than the number of E_5 events, that is, $\#E_2 \geq \#E_5$. Hence, we can conjecture that E_5 is *caused* by E_2.
- The number of E_1 and E_5 events do not have any sustained relationship ($\geq$ *or* $\leq$) between them. So, we do not predict a *causality* relation between the events.

Generalizing from the above discussion, we observe that we can mine dependencies between events by looking at the relation between the count of events $\#E_i$ that hold on all components of the Parikh map. So, deriving constraints similar to Equation 1 and Equation 2 will yield causality dependencies between the events. This is, essentially, a formal approach to using *happens before* as an indication of *causality*. The relation between the event counts helps us discover what events always happen before other events. We formalize the finding of above dependency constraints in the rest of the section.

Definition 5 (Octahedral constraints)**.** *An octahedral constraint is a linear inequality constraint over a set of variables x_i with all coefficients as 1, -1 or 0. Each constraint is of the form $\sum_i x_i - \sum_j x_j \geq d$ for any constant $d \in \mathbb{R}$.* □

We define *dependency constraints* as octahedral constraints over the count of events with the constant d in the inequality as always 0.

Definition 6 (Dependency Constraints)**.** *A dependency constraint $DC_{\mathbb{E}}(\sigma)$ on the trace σ is the set of octahedral constraints over the set of the count $\#E_i$s of the events in $\mathbb{E}$ such that (i) the constant term is 0, and (ii) all the elements of the Parikh mapping $p(\sigma)$ of the trace σ satisfy the dependency constraint $DC_{\mathbb{E}}(\sigma)$. Thus, dependency constraints take the form*

$$\sum_i \#E_i - \sum_j \#E_j \geq 0 \tag{3}$$

□

Given a dependency constraint, we can accumulate all events $\#E_i$ with coefficient 1 as *cause* events; and all the events $\#E_j$ with coefficient -1 as *effect* events. Then, Equation 3 and Equation 2 are of the same form.

B. SMT-based Search of Causality Constraints

We note that the number of possible dependency constraints is $3^{|\mathbb{E}|}$ for $|\mathbb{E}|$ events. Enumerating each of these $3^{|\mathbb{E}|}$ constraints and testing whether they are satisfied by all the elements in $p(\sigma)$ would be exponential in the number of events, and hence impractical. We have designed a Satisfiability Modulo Theory (SMT)-based search algorithm to search the domain of possible dependency constraints that are satisfied by the Parikh mapping of the traces.

Due to the presence of only integral coefficients in the dependency constraints, the constraint satisfaction problem can be solved using bit-vector satisfiability modulo theory solvers. The constraints are reduced to bit-vector constraints since all the variables are either 0, 1 or -1, and the constants corresponding to the number of events are bounded by the product of the length of the trace and the number of events, that is, $|\mathbb{E}||\sigma|$. In certain cases, an even tighter bound may be computed for a particular trace σ, and can be used as the upper bound for the event count. Given these bounds, we can find the appropriate length bit-vector to represent the variables and constants, and reduce the problem of finding dependency constraints to a bitvector SMT solving problem [8], [2].

The overall SMT based dependency constraint mining approach is show in Algorithm 1. The algorithm begins by constructing a template of dependency constraints as shown in Equation 2 and Equation 3. The unknown coefficients are limited to the three values $-1, 0$ or 1. An SMT solver is used to solve for these coefficients such that all the elements of the Parikh-map $p(\sigma)$ satisfy these constraints. The events with

Data: Set of events $\mathbb{E}$, Parikh-map of a trace $p(\sigma)$, choice k
Result: Dependency Constraints $DC^K_{\mathbb{E}}(\sigma)$
Construct the following SMT formula:

$$\Phi = \bigwedge_{w \in \sigma} \Bigg[\sum_{i=1}^{n} c_i(\#E_i(w)) \geq 0) \wedge (\sum_{i=1}^{n} \delta(c_i == -1) \leq k) \wedge (\sum_{i=1}^{n} \delta(c_i == 1) \leq k) \Bigg] \quad (4)$$

where $c_i \in \{-1, 0, 1\}$, $E_i \in \mathbb{E}$ and δ is a condition variable which is 1 if the condition is true and 0 otherwise;
while *(*$\texttt{SMT}(\Phi) == \texttt{SAT}$*)* **do**
  $\psi = \texttt{True}$;
  $i = 1$;
  while *(*$i \leq n$*)* **do**
    if *(*$c_i == 1$*)* **then**
      add E_i as cause event;
      $\psi = \psi \wedge (c_i == 1)$;
    end
    if *(*$c_i == -1$*)* **then**
      add E_i as effect event ;
      $\psi = \psi \wedge (c_i == -1)$;
    end
    if *(*$c_i == 0$*)* **then**
      $\psi = \psi \wedge (c_i == 0)$;
    end
    $i++$;
    Report cause and effect event dependency;
  end
  Extend Φ to eliminate the constraint found above:
  $\Phi = \Phi \wedge \neg\psi$;
end

Algorithm 1: SMT-based Mining of Dependency Constraints

coefficient 1 correspond to causes and those with coefficient -1 are effects. This constraint is then blocked from the formula, and the process is repeated till all the dependency constraints have been derived.

We limit our search for dependency constraints to those having only some constant number k of coefficients as 1 and -1. We refer to these dependency constraints as $DC^k_{\mathbb{E}}(\sigma)$. Our decision to search dependency constraints with a small number of nonzero coefficients is guided by the intuition that most events are caused by a small subset of events. This significantly reduces our search space. Jha and Langmead have earlier presented experimental evidence for this conjecture by analyzing the Ecocyc and Biocyc databases [6]. The number of possible dependency constraints is hence smaller than $3^{|\mathbb{E}|}$. In practice, we use small values of k which helps prune out fortuitous complicated inequalities that involve a lot of events, and are more likely to be incidental than real. The choice of k for biochemical pathways is typically between 2 and 8.

V. Uncovering Pathways from Constraints

While Algorithm 1 produces a set of constraints indicating causality relationships among biochemical entities $V = \{V_1, V_2, \ldots V_n\}$, it establishes a set of pairwise causal relationships among biochemical entities. We model these relationships using a directed graph $G = (V, E)$. If v_i has a causal dependence on v_j, we add the edge (v_i, v_j) to the list of edges E in the directed network.

Consider a set of biochemical entities D_1, D_2, D_3, D_4, D_5 and causal relationships $(D_1, D_2), (D_1, D_3), (D_4, D_5), (D_3, D_4), (D_1, D_4), (D_3, D_5)$, (D_1, D_5) obtained using SMT-based causal analysis.

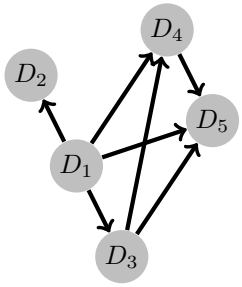

Fig. 1: A sample causal graph indicating the result of our causal analysis.

It is clear such a causality analysis produces a causality network G that does not bear any apparent similarity to a mechanistic biochemical signaling or reaction model. If (v_i, v_j) is a causal relationship and (v_j, v_k) is another causal relationship, then our algorithm will also discover (v_i, v_k) as a causal relationship. Given a causal graph G obtained by our causal analysis algorithm, we will compute a reduced graph G^* such that the causality network G can be obtained by computing the transitive closure of the reduced graph G^*. Several efficient algorithms [1], [9] exist for computing transitive reductions of directed networks.

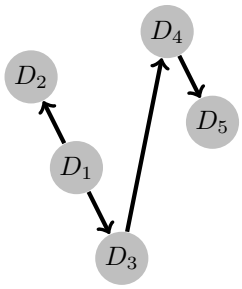

Fig. 2: A signaling mechanism obtained from the causal graph.

A single causal graph can produce multiple reduced networks (as indicated in Figure 2 and Figure 3). A search through a database of known biochemical pathways can be used to disambiguate among multiple biochemical mechanistic models.

VI. Conclusion and Future Work

In this paper, we have introduced a new methodology for causality analysis of biochemical networks that uses a combination of Parikh's map on execution traces, satisfiability modulo theory solving and transitive reduction of graphs.

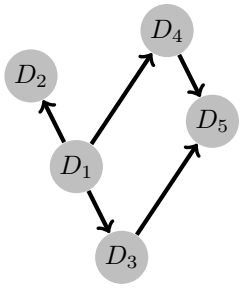

Fig. 3: Another biochemical signaling mechanism obtained from the same causal graph.

Unlike traditional notions of causality that have been developed for mathematical equation based models and observed experimental data sets, our approach treats the biochemical processes as first-class objects and directly reasons about the evolution of these processes.

Several opportunities for future work remain open. Our algorithms are deterministic and do not use ideas of probabilistic amplification; hence, they may not work well on those biochemical data sets that arise from experimental observation of stochastic biochemical systems. By combining Parikh's map on traces with probabilistic reasoning, we will be able to filter out the noise and incidental "happens before" relationships caused by the randomness inherent in many biochemical systems.

Our focus on the use of traces obtained from a biological process naturally allow the use of multiple experimental data sets of varying veracity in the mechanism identification process. In particular, our approach can be extended to account for experimental data sets at different resolutions and with different trustworthiness. Such an approach is perhaps not easily feasible in a causality analysis method that does not expose the computational nature of a biological process and reasons about its observed traces (as opposed to flat time-series data or statistical observation summaries).

We are investigating probabilistic extensions to our algorithm that can be applied to recover mechanistic models from their executions. Together with ongoing research into the discovery of parameters for probabilistic computational models, we believe that this method may be helpful in algorithmically constructing big mechanistic models from big data.

REFERENCES

[1] Alfred V. Aho, Michael R Garey, and Jeffrey D. Ullman. The transitive reduction of a directed graph. *SIAM Journal on Computing*, 1(2):131–137, 1972.

[2] Clark Barrett, Aaron Stump, Cesare Tinelli, Sascha Boehme, David Cok, David Deharbe, Bruno Dutertre, Pascal Fontaine, Vijay Ganesh, Alberto Griggio, Jim Grundy, Paul Jackson, Albert Oliveras, Sava Krsti, Michal Moskal, Leonardo De Moura, Roberto Sebastiani, To David Cok, and Jochen Hoenicke. C.: The smt-lib standard: Version 2.0. Technical report, 2010.

[3] Steven L Bressler and Anil K Seth. Wiener–granger causality: a well established methodology. *Neuroimage*, 58(2):323–329, 2011.

[4] Clive W. J. Granger. Time Series Analysis, Cointegration and Applications, 2003. [Nobel Prize Lecture].

[5] Sumit Gulwani. Synthesis from examples: Interaction models and algorithms. In *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2012. Invited talk paper.

[6] Sumit Kumar Jha and Christopher James Langmead. Poster: Synthesis of biochemical models. In *Computational Advances in Bio and Medical Sciences (ICCABS), 2011 IEEE 1st International Conference on*, pages 248–248. IEEE.

[7] Susmit Jha, Wenchao Li, and Sanjit A. Seshia. Localizing transient faults using dynamic bayesian networks. In *14th IEEE International High-Level Design Validation and Test Workshop*, 2009.

[8] Susmit Jha, Rhishikesh Limaye, and Sanjit A. Seshia. Beaver: Engineering an efficient smt solver for bit-vector arithmetic. In *21st International Conference on Computer-Aided Verification*, pages 668–674, 2009.

[9] Johannes A La Poutr and Jan van Leeuwen. Maintenance of transitive closures and transitive reductions of graphs. In *Graph-theoretic concepts in computer science*, pages 106–120. Springer.

[10] Tessa Lau, Pedro Domingos, and Daniel S. Weld. Learning programs from traces using version space algebra. In *Proceedings of the 2nd international conference on Knowledge capture*, K-CAP '03, pages 36–43, New York, NY, USA, 2003. ACM.

[11] Susan Lomax and Sunil Vadera. A survey of cost-sensitive decision tree induction algorithms. *ACM Comput. Surv.*, 45(2):16:1–16:35, March 2013.

[12] Stephen Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[13] Rami A. Namas, John Bartels, Rosemary Hoffman, Derek Barclay, Timothy R. Billiar, Ruben Zamora, and Yoram Vodovotz. Combined in silico, in vivo, and in vitro studies shed insights into the acute inflammatory response in middle-aged mice. *PLoS ONE*, 8(7):e67419, 07 2013.

[14] Negin Nejati, Pat Langley, and Tolga Konik. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, pages 665–672, New York, NY, USA, 2006. ACM.

[15] Rohit J. Parikh. On context-free languages. *J. ACM*, 13(4):570–581, October 1966.

[16] Sheela Ramanna, Lakhmi C Jain, and Robert J Howlett. *Emerging paradigms in machine learning*. Springer Publishing Company, Incorporated, 2012.

[17] Chun Yuan, Ni Lao, Ji-Rong Wen, Jiwei Li, Zheng Zhang, Yi-Min Wang, and Wei-Ying Ma. Automated known problem diagnosis with event traces. *SIGOPS Oper. Syst. Rev.*, 40(4):375–388, April 2006.