Learning Petri Net Models of Biological Systems using ILP

Ashwin Srinivasan[1] and Michael Bain[2]

[1] Department of Computer Science
South Asian University, New Delhi, India.
[2] School of Computer Science and Engineering,
University of New South Wales, Sydney, Australia.

1 Introduction

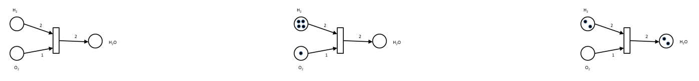
Networks are ubiquitous in Biology. They are used to represent biological relationships ranging across all levels of organisation: for example, relationships between organisms, and between an organism and its environment; the flow of energy and matter in an ecosystem; the pathway of carbon atoms through an ecosystem from producers of organic compounds to consumers that release carbon by respiration; the nitrogen cycle that links the environment to proteins and compounds that form the bodies of living things; the stimulus-response mechanisms in constituting nervous pathways; the regulation and control of endocrine glands; the events related to the division and replication of cells; and intra- and inter-cellular interactions between chemicals.

Computationally, substantial research effort has been, and continues to be invested in developing models of biological networks [5]. While much of this research has been directed at representation and reasoning, the emerging field of Systems Biology [4] has highlighted the need to extract automatically models of networks from experimental data. The requirement is for models that not only determine the underlying relationships amongst entities, but are also capable of simulating the dynamics of the system. ILP, with the ability to extract complex relations from data is a natural choice for identifying network structure, but less has been done on models that are able to generate system dynamics. The most expressive, and understandable dynamic models of biological systems identified by ILP have employed qualitative differential equations, or QDEs [9]. The representation of QDEs provides a direct and simple abstraction of quantitative ODEs. However the representation does have limitations. First, simulation can produce spurious behaviour, arising from the ambiguities inherent in the qualitative approach. Second, issues of concurrency, which are prevalent in biological systems, are not well handled. Third, there appears to be no straightforward mechanism of introducing any form of quantitative information. Fourth, there is little room for accounting for stochastic aspects inherent in the system. Most of these issues are largely absent in the long-established qualitative representation of Petri nets. Starting from a simple bipartite graph representation that is ideally suited for metabolic networks, Petri nets have been extended in a number of ways that are of interest for biological networks. This incorporates timing (timed Petri nets), concentrations (continuous Petri mets), stochasticity (stochastic Petri nets), multiple levels of organisation (hierarchical Petri nets), inhibition and activation relations (Petri nets with “test” and “repressor” arcs), and so on. Mathematically, the power of Petri nets ranges from simple qualitative producer-consumer models to that of quantitative ODEs. Computationally, the range is from above regular languages to Turing machines. A flourishing area of Petri net models for biological systems now exists [6], which has almost entirely been concerned with hand-crafted models.

In this paper we show that a known combinatorial algorithm for identifying (pure) Petri nets from data can be formulated as a search over a lattice of incidence matrices of Petri nets; and that there is a correspondence between this lattice and a lattice of definite clauses ordered by subsumption, thus allowing us to use an ILP system to learn Petri nets. This has some advantages over using a specialised Petri net learner for biological system identification. First, advances made in ILP on efficient search should allow the exploration of a larger space of models than the enumerative techniques employed in the Petri net literature. Second, we are able to use well-established network models as background knowledge to learn structured Petri net models representing compartmentalised models of biological systems. Third, the ILP setting enables us to go beyond learning simple Petri nets, to include tests on activation states. This allows us to learn not only metabolic networks but signalling networks also. We demonstrate each of these advantages using some well-known biological networks.

2 Petri Nets: an Example

Figure 1(a) shows a simple Petri net, with two kinds of nodes. Conventionally, the circular nodes are called *places* and the rectangular nodes are called *transitions*. Edges can only between a place and a transition or vice-versa (but never from one place to another, or from one transition to another), and each edge has a weight (or label: by convention, if a label is absent, then the weight is taken to be 1). A transition thus has a finite number of input places and a finite number of output places. Places can contain 0 or more tokens (usually shown as small black circles, as in Fig. 1(b)), and the dynamics of the system are described by the *firing* of transitions and the movement of tokens from one place to another. A transition is *enabled* if the number of tokens at each input place for the transition is at least equal to the weight of the arc from the place to the transition (a transition with no input places is always enabled). An enabled transition can *fire*, resulting in consuming tokens from an input place and depositing tokens in an output place: the numbers of tokens consumed and deposited being determined by the arc weights. The state of the Petri net at any point in time is the number of tokens at each place, and is called a *marking*. It is evident from Fig. 1 that this “token game” is ideally suited for chemical reactions in which reactants are consumed and products are produced.



(a) (b) (c)

Fig. 1. (a) A simple Petri net representing the reaction $2H_2 + O_2 \rightarrow 2H_2O$; (b) An "initial marking", in which molecules of hydrogen and oxygen are shown by tokens (small solid circles); (c) A "final marking", which results in two molecules of water, from the molecules of hydrogen and oxygen in (b).

3 Learning Petri Nets

A Petri net [2] is a directed bipartite graph, with two node (vertex) types, called places and transitions. Directed arcs (edges) are either from a place to a transition (consumption) or a transition to a place (production). Place nodes and arcs may be labelled. If the sets of places or transitions are empty this denotes a *degenerate* Petri net. A Petri net may be defined as a tuple $\langle P, T, M \rangle$, where P is a set of places, T is a set of transitions and the $P \times T$ matrix M is the *incidence matrix* of the graph. In this paper we will represent a matrix using square brackets and write vectors in transposed form using round brackets, e.g., $(1, 0, -1) = [1, 0, -1]^{\mathrm{T}}$.

In systems biology the incidence matrix represents the *stoichiometry* of the system, i.e., the relative quantities of all molecular species in each of the reactions in the system [7]. An entry (i, j) in this matrix denotes the net transfer of tokens between place i and transition j on a firing. For example, tokens might represent concentration levels of a molecular species in a reaction.

3.1 Pure Petri Nets

A *self-loop* in a Petri net is a pair of directed arcs, (p, t) from a place p to a transition t, and (t, p) in the reverse direction. A Petri net without self-loops is called *pure*, otherwise it is *impure*. Without loss of generality, since any impure Petri net can be converted to a pure Petri net [2], in this paper we consider only pure Petri nets.

A Lattice of Petri Nets The $P \times T$ incidence matrix M can be written using block matrix notation as $M = \left[P_1, P_2, \ldots, P_{|T|}\right]$. Each vector P_j, $1 \leq j \leq |T|$, is of size $|P|$. If M is a stoichiometric matrix then P_j is a *reaction vector* for transition t_j, written $R(t_j)$. M can be viewed as an ordered set of vectors P_j, and, by assuming each reaction vector $R(t_j)$ is uniquely identified, M can be represented as a set of reaction vectors.

In a pure Petri net, only one arc can connect a place p and transition t, either (p, t) or (t, p), but not both. The corresponding entry in the reaction vector $R(t)$ will be a negative integer if the arc is (p, t), i.e., consumption or removal of tokens from p, or a positive integer if it is (t, p), i.e., production or addition of tokens to p. If there is no such arc the entry is zero.

A standard approach for any implementable Petri net is to ensure that it is k-bounded [2]. A place p is k-bounded for an initial marking m_0 if in any marking reachable from m_0 the number of tokens does not exceed a non-negative integer bound k. A Petri net is k-bounded for an initial marking if all places are k-bounded.

Assuming that all markings are derivable from a pure k-bounded Petri net using a linear combination of reaction vectors ensures that no reaction vector component has an absolute value that exceeds k, since otherwise in some marking a place could have more than k tokens. Therefore the set of all possible $|P|$-sized reaction vectors of a pure k-bounded Petri net is finite with size $S = (2k+1)^{|P|}$.

Since the degenerate Petri net with no transitions can be represented by the empty set, the set of all pure k-bounded Petri nets is the power-set of the set of all possible reaction vectors, with size 2^S. Representing the incidence matrix of a Petri net by the set of reaction vectors corresponding to its transitions we obtain the following generality ordering between Petri nets: M_1 subsumes M_2, denoted $M_1 \succeq M_2$, if $M_1 \subseteq M_2$.

This power-set of reaction vectors and subsumption ordering forms a lattice of Petri nets, where the ordering is based on set inclusion, but where the lattice is the *inverse* of the usual subset lattice [1]. Here the bottom element is the set of all reaction vectors, and top is the empty set (the degenerate Petri net). In this lattice every pair of sets has a supremum (least upper bound) defined by their set intersection, and infimum (greatest lower bound) defined by their union. Since the lattice is finite, it is complete [1]. This lattice corresponds to the natural conception of specialising a Petri net by adding transitions.

Petri Net Reconstruction In the systems biology methodology [4] perturbation of system components results in observed state changes which need to be incorporated into a model of the system. For a given set of places P in a Petri net the states are markings and changes are represented as $|P|$-sized *difference vectors* obtained by subtracting a state from its successor, i.e., $D_{j+1} = m_{j+1} - m_j$, where the m are markings. For biological applications a difference vector D can be expressed as a linear combination of $|P|$-sized reaction vectors that denote transitions specifying the movement of tokens between places in the system.

The difficulty is that difference vectors under-determine the network to be modelled, since more than one set of reaction vectors can be consistent with a single set of difference vectors. In [3] the Petri Net Reconstruction problem is formulated as a combinatorial algorithm to find the *minimal* Petri net, or set of reaction vectors, that fit a time series data set expressed as difference vectors.

An ILP Approach The algorithm of [3] is essentially a search through the set of subsets of possible reaction vectors. Given the lattice structure established above it is easy to see that an ILP search could be used instead, based on the following representation of a Petri net as a definite clause.

Definition 1. *r*-literal. *An r-literal is a literal with three arguments: the first, N, is an identifier for a Petri net; the second, T, is an identifier for a transition node in the Petri net; and the third, R, is a list integers denoting the reaction vector for T in the incidence matrix of N.*

As is usual practice, it is convenient to regard a clause as a set of literals.

Definition 2. Petri net definite clause. *A Petri net definite clause is a set of literals. The head literal has a single argument, N, an identifier for a Petri net. The clause body is a (possibly empty) set of r-literals.*

Clearly there is a clause lattice corresponding to the Petri net lattice that enables the use of an ILP refinement operator to search it. We have developed and implemented such an approach using the ILP system Aleph [8] and report first results below. The details are left for the full version of the paper.

3.2 General Petri Nets

In [3] only simple transitions can be included, restricted to threshold logic conditions for a standard firing rule. However, by representing a Petri net as a clause in a logic program we enable the introduction of transitions in the form of arbitrary subnets encoded in background knowledge, as required for examples (b) and (c) in Figure 2. Furthermore we extend the representation of *r-literals* to enable checking of generalized pre- and post-conditions on transitions defined with respect to *arbitrary* background knowledge, thus enabling the integration of generic biological circuits, as in example (c) in Figure 2. The description of this extended clause lattice and refinement operator can only be reported here by way of the examples in Section 4 due to lack of space.

4 Applications

For reasons of space, we only show results of the kinds of networks we have been able to identify from simulated data, using the ILP system Aleph [8]. In Fig. 2, we have selected three networks that illustrate three separate aspects of learning Petri nets using an ILP system. The first, Glycolysis, shows that reasonably large networks can be identified using the ILP version of the combinatorial search in [3] (the examples shown in that paper are substantially smaller by comparison). In the second, we use one stage of the Glycolysis network as background knowledge to the ILP system to allow a more complex transition. In the third, a generic biochemical structure is used as background knowledge. New transitions are obtained when the ILP system instantiates this generic structure to specific enzyme-enabled reactions. These new transitions are used to identify the final network structure. Both the second and third networks illustrate ways of constructing networks that go beyond the basic algorithm in [3]. In the final version of the paper we will describe in detail how these networks are identified.

5 Conclusions

We have developed and implemented a novel approach to learning dynamic models of biological systems by combining Petri nets with ILP. Petri nets provide an

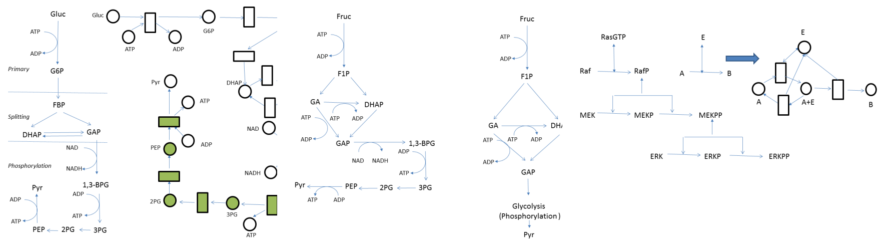


Fig. 2. Example networks identified by Aleph. (a) Glycolysis and its pure Petri net representation; (b) Fructose metabolism; and (c) the Kinase cascade in the MAPK pathway. The shaded portions in (a) correspond to the Petri net representation of the phosphorylation stage of Glycolysis, which as a "macro-transition" for (b). In (c), a generic enzymatic reaction structure is used repeatedly to establish a cascade of phosphorylation reactions (the corresponding Petri net structure of the reaction is shown on the extreme right).

effective framework for modelling and simulation, and ILP a powerful means of representing and learning with background knowledge. As far as we are aware the application of this approach has resulted in learning the largest dynamic network models of biological systems to date. Also, we believe this is the first time ILP has been used to combine a comprehensible logical representation with learning of matrix models based on linear algebra.

Acknowledgements. A.S. acknowledges the support of the School of Computer Science and Engineering and the Centre for Health Informatics at UNSW, and M.B. acknowledges the support of the Faculty of Engineering at UNSW.

References

1. B.A. Davey and H.A. Priestley. *An Introduction to Lattices and Order (Second Edition)*. Cambridge University Press, 2002.
2. R. David and H. Alla. *Discrete, Continuous, and Hybrid Petri Nets*. Springer, Berlin, Second edition, 2010.
3. M. Durzinsky, A. Wagler, and R. Weismantel. An algorithmic framework for network reconstruction. *Theoretical Computer Science*, 412:2800–2815, 2011.
4. T. Ideker, T. Galitski, and L. Hood. A new approach to decoding life: systems biology. *Ann. Review of Genomics and Human Genetics*, 2:343–372, 2001.
5. B. H. Junker and F. Schreiber. *Analysis of Biological Networks*. Wiley, NJ, 2008.
6. I. Koch, W. Reisig, and F. Schreiber, editors. *Modeling in Systems Biology: the Petri Net Approach*. Springer, Berlin, 2011.
7. B. Palsson. *Systems Biology: Properties of Reconstructed Networks*. Cambridge University Press, Cambridge, 2006.
8. A. Srinivasan. *The Aleph manual*. University of Oxford, Oxford, 2007.
9. A. Srinivasan and R. D. King. Incremental Identification of Qualitative Models of Biological Systems using Inductive Logic Programming. *Journal of Machine Learning Research*, 9:1475–1533, 2008.